



Université de Metz



Nom : _____ Prénom : _____

Groupe : _____

T.P. de Système de Base de Données : MySQL

I.U.P. G.S.I. 3^e Année

Année Universitaire 2003/2004



Cette page est laissée blanche intentionnellement

Table des matières

1	Introduction à MySQL	7
1.1	Présentation de MySQL	7
1.2	Création de bases, gestion des droits	7
1.3	Eléments du langage de MySQL	8
1.3.1	Connexion et déconnexion du serveur	9
1.3.2	Soumettre une requête	9
1.3.3	Exemples de requêtes	12
1.3.3.1	Valeur maximale d'une colonne	12
1.3.3.2	La ligne qui contient le maximum d'une colonne	13
1.3.3.3	Maximum d'une colonne : par groupement	13
1.3.3.4	La ligne contenant le maximum d'une colonne d'un groupe	14
1.3.3.5	Utiliser des clés étrangères	15
2	TP1 : Votre première base de données	17
2.1	Créer et utiliser une base de données	17
2.1.1	Créer une table	18
2.1.2	Charger des données dans une table	19
2.1.3	Lire des informations dans une table	20
2.1.4	Utiliser plus d'une table	32
2.2	Informations sur les bases de données et tables	34
3	TP 2 : Une base de données de publications	37
3.1	Introduction	37
3.2	Création des tables	37
3.2.1	La table <code>specialites</code>	38
3.2.2	La table <code>statuts</code>	38
3.2.3	La table <code>auteurs</code>	39
3.2.4	La table <code>docs</code>	39
3.2.5	La table <code>lien_docs_auteurs</code>	40
3.3	Insertion des données	40
3.3.1	Dans la table <code>specialites</code>	40
3.3.2	Dans la table <code>statuts</code>	41
3.3.3	Dans la table <code>auteurs</code>	42
3.3.4	Dans la table <code>docs</code>	44
3.3.5	Dans la table <code>lien_docs_auteurs</code>	44
3.4	Insertion des données par fichier	45
3.4.1	Ajout d'auteurs	45
3.4.2	Ajout de publications	45
3.4.3	Ajout de liens documents auteurs	45
3.5	Les Requêtes	46

3.5.1	Requêtes simples	46
3.5.2	Requêtes de mises à jour	49
3.5.3	Requêtes de selection de lignes	50
3.5.4	Sélection de colonne, comptage et tri	51
3.5.5	Requêtes de recherche de valeurs	53
3.5.6	Requête avancée	54
4	Petit cours de HTML	57
4.1	Le principe des balises	57
4.2	La structure des pages	58
4.3	Créer des caractères spéciaux	58
4.4	Le style du texte	59
4.5	La mise en forme du texte	59
4.6	Mettre des couleurs	60
4.7	Insérer un lien hypertexte	62
4.8	Insérer une image	62
4.9	Créer un tableau	62
4.10	Créer des formulaires	64
4.10.1	Structure de base	64
4.10.2	Attributs de la balise <FORM>	64
4.10.3	Zones de saisie à une ligne	64
4.10.4	Choisir, cocher, sélectionner	65
4.10.4.1	Cases à cocher	66
4.10.4.2	Listes de choix	66
4.10.4.3	Les boutons de commande	66
4.11	Créer des cadres (ou "frame")	67
5	Petit cours de PHP et MySQL	69
5.1	Afficher une phrase ou une image	69
5.2	Afficher la date et l'heure	70
5.3	PHP dans du code HTML	70
5.4	La concaténation	72
5.5	Récupérer les valeurs d'un formulaire	73
5.6	Les structures de contrôles	75
5.7	Ecrire et lire dans un fichier texte	78
5.8	Les fonctions utilisateurs	79
5.9	Les variables d'environnement	80
5.10	Quelques fonctions utiles	80
5.11	SQL/MySQL (Create, Alter & Drop)	81
5.11.1	CREATE TABLE	81
5.11.2	ALTER TABLE	83
5.11.3	DROP TABLE	83
5.12	SQL/MySQL (Insert et Select)	83
5.13	SQL/MySQL (Delete et Update)	86
5.14	Fonctions PHP pour mySQL	87
5.15	Interroger une table MySQL	89
5.16	Alimenter une ou plusieurs tables mySQL	93
5.16.1	Alimenter une table	93
5.16.2	Alimenter deux tables et créer une liaison	95

6 TP3 : Utilisation de phpMyAdmin	97
6.1 Connexion à phpMyAdmin	97
6.2 But du TP	99
7 TP4 : Interrogation d'un base de données via le Web	101
7.1 Introduction	101
7.1.1 Connection à la base	101
7.1.2 Première requête	102
7.2 Formulaire 1 : Lister les auteurs par nom et prénom	104
7.3 Formulaire 2 : Lister les auteurs en selectionnant les champs à afficher	106
7.4 Formulaire 3 : Ajouter un utilisateur dans la base de données	107
7.5 Formulaire 4 : Requête Bibliographie	109

Cette page est laissée blanche intentionnellement

Chapitre 1

Introduction à MySQL

Cette petite présentation est constituée du chapitre 7 de la documentation française de MySQL disponible à l'adresse suivante <http://www.nexen.net>.

1.1 Présentation de MYSQL

MYSQL est un gestionnaire de base de données SQL. Il se compose d'un langage de définition de données et de droits ainsi qu'un langage de manipulation des données.

Une base de données regroupe plusieurs tables de données. Par exemple pour une application dont la base est nommée GESTION, les tables pourront s'appeler CLIENTS, FOURNISSEURS, ARTICLES et COMMANDES.

Ce chapitre est une introduction à *MySQL* qui montre comment utiliser le client *mysql* pour créer et utiliser une base de données simple. *mysql* est un client (parfois appelée "terminal" ou aussi "moniteur") qui permet à un utilisateur de se connecter à un serveur *MySQL*, de lancer quelques requêtes, et de voir les résultats. *mysql* peut aussi être lancé en mode automatique, en lui précisant un fichier qui contient les commandes à exécuter. Cette présentation de *mysql* couvre les deux aspects.

Pour avoir la liste des options disponibles sur *mysql*, il suffit d'utiliser l'option : *-help*.

```
[yann@ulyse yann]$ mysql --help
mysql Ver 11.15 Distrib 3.23.47, for pc-linux-gnu (i686)
Copyright (C) 2000 MySQL AB & MySQL Finland AB & TCX DataKonsult AB
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL license
```

```
Usage: mysql [OPTIONS] [database]
```

```
  -?, --help          Display this help and exit.
.....
[yann@ulyse yann]$
```

Ce chapitre supposera que *mysql* est installé sur votre machine, et qu'un serveur *MySQL* est accessible. Si ce n'est pas le cas, contactez votre administrateur *MySQL* (Si vous êtes l'administrateur, vous aurez certainement besoin de consulter d'autres sections de ce manuel).

Ce chapitre couvre la constitution et l'utilisation d'une base de données. Si vous êtes simplement intéressé par la lecture de bases de données déjà existantes, vous pouvez éviter les premières sections qui montrent la création d'une base de données et de tables.

Etant donné que ce chapitre n'est qu'un exemple d'introduction, de nombreux détails sont laissés de côté.

N'hésitez pas à vous reporter aux autres sections du manuel, pour toute information complémentaire.

Il est à noter que chaque instruction MYSQL peut s'étendre sur plusieurs lignes mais qu'elle doit se terminer par un point-virgule.

1.2 Création de bases, gestion des droits

C'est en principe le responsable système ("root") qui crée une base, par exemple avec la commande *mysqladmin*. Les droits peuvent ensuite être transmis par l'instruction GRANT. Par exemple, la création de la base tuteur par

le responsable système se fait avec

```
mysqladmin create essai ;
```

ensuite, ce même responsable peut exécuter en ligne

```
mysql tuteur -e "GRANT ALL PRIVILEGES ON essai.* TO gh@localhost ;"
```

et l'utilisateur gh peut alors localement faire tout ce qu'il veut avec la base. Les droits gérés par GRANT sont

```
ALL PRIVILEGES, ALTER, CREATE, DELETE, DROP, FILE, INDEX, INSERT,
PROCESS, REFERENCES, RELOAD, SELECT, SHUTDOWN, UPDATE, USAGE
```

Signalons que si l'instruction GRANT se termine par WITH GRANT OPTION ; l'utilisateur désigné peut à son tour transmettre des droits.

1.3 Eléments du langage de MYSQL

Un commentaire est une instruction qui commence par un dièse #. La première instruction d'une session ou d'un programme MYSQL doit être USE Nom_Base ;

L'instruction SELECT permet d'afficher des valeurs et d'extraire des données des bases. Par exemple :

```
SELECT VERSION() ;
```

affiche le numéro de version courante

```
SELECT COUNT(*) FROM Nom_Table ;
```

indique le nombre d'enregistrements dans la table.

L'instruction SHOW affiche de nombreux renseignements concernant les bases et les tables. Par exemple :

```
SHOW DATABASES ;
```

donne la liste de toutes les bases.

```
SHOW VARIABLES ;
```

donne la liste et la valeur de toutes les variables.

```
SHOW STATUS ;
```

décrit l'état de l'ensemble des paramètres de MYSQL.

L'instruction DESCRIBE donne des informations sur une table particulière. Par exemple :

```
USE essai ;
```

```
DESCRIBE tuteur ;
```

décrit toutes les variables de la table Tuteur pour la base essai alors que

```
DESCRIBE tuteur qt ;
```

ne décrit que le champ qt de cette table.

La création d'une table se fait avec l'instruction CREATE suivi du mot table et du nom de la table. On indique ensuite entre parenthèses le nom et le type des champs en séparant les champs par des virgules. Exemple :

```
CREATE TABLE fournisseurs ( code INT, nom CHAR ) ;
```

Les type de champs possibles sont

```
INT, TINYINT, SMALLINT, MEDIUMINT, BIGINT,
FLOAT, DOUBLE, DECIMAL,
DATE, DATETIME, TIMESTAMP, TIME, YEAR,
CHAR, VARCHAR, TEXT, TINYTEXT, MEDIUMTEXT, LONGTEXT,
ENUM, SET.
```

Un fois la table créée, il est possible de modifier la structure avec l'instruction ALTER, comme par exemple :

```
ALTER TABLE fournisseurs ADD prenom char, ADD ref int ;
```

Les spécifications possibles pour ALTER sont ADD, ALTER, CHANGE, MODIFY, DROP, RENAME.

Pour détruire une table, on utilise DROP, comme par exemple :

```
DROP TABLE fournisseurs ;
```


1.3.1 Connection et déconnection du serveur

Pour se connecter au serveur *MySQL*, il vous faut un nom d'utilisateur, et, le plus probablement, un mot de passe. Si le serveur tourne sur une autre machine, il vous faudra aussi un nom d'hôte. Contactez l'administrateur de la machine pour connaître les paramètres de connexion (i.e. le nom d'hôte, le nom d'utilisateur, et le mot de passe). Une fois que vous connaîtrez tous ces paramètres, vous pourrez vous connecter comme ceci :

```
yann@yoda:~$ mysql -h host -u user -p
```

```
Enter password: *****
```

Les ********* représentent votre mot de passe : saisissez le lorsque *mysql* affiche the **Enterpassword (Entrez votre mot de passe)** : invite.

Si tout a bien fonctionné, vous devriez voir s'afficher des informations d'introduction, suivies d'une invite de commande : *mysql*> prompt :

```
yann@yoda:~$ mysql -u mysql -p
```

```
Enter password:
```

```
Welcome to the MySQL monitor. Commands end with ; or \g.
```

```
Your MySQL connection id is 100 to server version: 3.23.49-log
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

```
mysql>
```

L'invite de commande vous indique que *mysql* est prêt à recevoir des commandes.

Certaines installations de *MySQL* permettent l'accès anonyme au serveur. Si c'est le cas de votre machine, vous devriez pouvoir vous connecter sans fournir aucune information.

```
yann@yoda:~$ mysql
```

Après s'être correctement connecté, vous pouvez vous déconnecter à tout moment, en tapant **QUIT** à l'invite de *mysql*.

```
mysql> QUIT
```

```
Bye
```

```
yann@yoda:~$
```

Vous pouvez aussi vous déconnecter en tapant les touches contrôle-D.

Par la suite, nous supposons que vous vous êtes correctement connecté au serveur.

1.3.2 Soumettre une requête

Assurez vous que vous êtes correctement connecté. Dans le cas contraire, reportez vous à la section précédente. Ce faisant, vous ne vous êtes en fait connecté à aucune base de données, mais c'est bien comme ça. A ce stade ; il est important de savoir comment envoyer une requête, avant de savoir créer une table, charger des données, et interroger la base. Cette section décrit les principes de base de saisie des commandes, en utilisant des commandes qui vous familiariseront avec le fonctionnement de *MySQL*.

Voici une commande simple qui demande au serveur la version et la date courante. Saisissez la comme ci-dessous, puis appuyez sur la touche entrée.

```
mysql> select version(), current_date();
```

```
+-----+-----+
```

```
| version() | current_date() |
```

```
+-----+-----+
```

```
| 3.23.49-log | 2002-08-01 |
```

```
+-----+-----+
```

```
1 row in set (0.06 sec)
```

```
mysql>
```



Cette première requête montre beaucoup de caractéristiques de *mysql*

Une commande consiste généralement d'une commande SQL, suivie d'un point-virgule (Il y a quelques exceptions, ou les point-virgules ne sont pas nécessaires, comme la commande *QUIT*, vue précédemment. Nous y reviendrons plus loin).

Quand une requête a été saisie, *mysql* l'envoie au serveur pour qu'il l'exécute, puis affiche le résultat, et repropose une nouvelle invite de commande : *mysql>*.

mysql> affiche la réponse du serveur sous forme de table (lignes et colonnes). La première ligne contient les titres des colonnes. Les lignes suivantes présentent les résultats de la requête. Généralement, les noms de colonnes sont les noms des colonnes des tables utilisées. Si la valeur retournée est une expression plutôt qu'une table, (comme dans l'exemple ci-dessus), *mysql* crée une colonne avec comme titre l'expression évaluée.

mysql affiche le nombre de ligne retourné, et le temps de traitement de la requête, ce qui donne une idée de la performance globale du serveur. Ces valeurs sont imprécises, car elle représente le temps passé entre l'envoi de la commande et la réception de la réponse, et ne montre pas quelle quantité de processeur a été utilisée. Cela ne permet pas de connaître la charge du serveur, ou les retards du réseau.

Par un souci de concision, la ligne "rows in set" ne sera plus affichée dans les exemples ultérieurs.

Les mots clés du langage peuvent être en majuscule ou minuscule, au choix. Les lignes suivantes sont équivalentes :

```
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
```

```
mysql> SeLeCt vErSiOn(), current_DATE;
```

Voici une autre requête qui montre que *mysql* peut être utilisé comme une simple calculatrice.

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+-----+
| SIN(PI()/4) | (4+1)*5 |
+-----+
| 0.707107    | 25      |
+-----+
```

Les commandes que nous venons de voir sont relativement courtes, et tiennent sur une seule ligne. Il est possible de saisir plusieurs commandes sur une seule ligne, il suffit de toujours les terminer par des points-virgules.

```
mysql> SELECT VERSION(); SELECT NOW();
```

```
+-----+
| version()  |
+-----+
| 3.22.20a-log |
+-----+
```

```
+-----+
| NOW()      |
+-----+
| 1999-03-19 00:15:33 |
+-----+
```

Une commande n'est pas obligatoirement sur une seule ligne : les commandes les plus longues peuvent tenir sur plusieurs lignes. Ce n'est pas un problème, car *mysql* détermine la fin de la commande grâce au point-virgule, et non pas en cherchant la fin de la ligne (en d'autres termes, *mysql* accepte n'importe quel format de colonne, mais ne les exécute que si il trouve un point-virgule à la fin de la commande).

Voici une commande simple, et multi-lignes :

```
mysql> select
-> user()
-> ,
-> current_date;
+-----+
```

```

| user()          | current_date |
+-----+-----+
| mysql@localhost | 2002-08-01  |
+-----+-----+
1 row in set (0.00 sec)

```

mysql>

Dans cet exemples, vous avez pu remarquer que l'invite passe de *mysql>* à *->* dès que la commande devient multi-lignes. C'est par ce biais que *mysql* indique qu'il n'a pas trouvé une commande complète, et qu'il attend un complément d'information. En observant bien l'invite de commande, vous saurez toujours ce que *mysql* attend de vous.

Pour annuler une commande qui est partiellement saisie, il suffit de taper '\c' (antislash-c)

```

mysql> SELECT
-> USER()
-> \c
mysql>

```

Ici, l'invite de commande reprend son aspect initial. Cela indique que *mysql* est prêt pour une nouvelle commande.

La table suivante montre les différentes formes de l'invite de commande, et sa signification :

Une commande peut s'étendre sur plusieurs lignes si, par accident, vous oubliez de terminer votre ligne par un point-virgule. Dans ce cas, *mysql* attend plus d'informations :

```

mysql> SELECT USER()
->

```

Si cela vous arrive (vous pensez avoir entré une commande, mais la seule réponse est cette désespérante invite *->*); le plus souvent *mysql* attends le point-virgule. Si vous ne comprenez pas que *mysql* attend la suite de votre commande, vous risquez d'attendre un bon moment. Il suffit alors de compléter la commande avec un point-virgule, pour valider la commande.

```

mysql> select user()
-> ;
+-----+
| user()          |
+-----+
| mysql@localhost |
+-----+
1 row in set (0.00 sec)

```

mysql>

Les formes *>* et *>* d'invite de commande apparaissent lors de la saisie de chaînes. Avec *MySQL*, vous pouvez écrire des chaînes avec les guillemets simples et doubles : *''''* ou *''''''* ; 'comme par exemple *'bonjour'* ou *"au revoir"*. *>* et *>* signifie donc que vous avez commencé à saisir une chaîne de caractères, mais que vous n'avez pas encore fini. Si vous saisissez une chaîne de plusieurs lignes, c'est une indication judicieuse, mais est-ce souvent le cas ? En général, ces deux invites de commande indiquent que vous avez oublié de refermer les guillemets :

```

mysql> SELECT * FROM my\_table WHERE nom = "Smith AND age 60; 30;
">;

```

Si vous saisissez cette commande *SELECT* , puis tapez ENTREE, il ne va rien se passer. Plutôt que de se demander " mais qu'est ce qui prend tant de temps ", il vaut mieux remarquer que l'invite a pris la forme particulière de *>* . Cela signifie que *mysql* s'attend ce que vous complétiez votre chaîne et la commande.

En effet, la chaîne *"Smith* n'a pas de deuxième guillemet.

A ce moment, que faire ? La chose la plus simple d'annuler la commande. Cependant, vous ne pouvez pas taper *\c* , car *mysql* l'interprétera comme un caractère de chaîne. A la place, il faut clore la chaîne, puis taper *\c* .

```
mysql> SELECT * FROM my_table WHERE nom = "Smith AND age 60; 30;
">"; "\c
mysql>
```

L'invite de commande redevient *mysql>*, ce qui indique que *mysql* est prêt pour une nouvelle commande.

Il est important que les invites de commande de la forme signifie '>' et '>' que vous n'avez pas terminé une chaîne, et que toutes les lignes suivantes seront ignorées par l'analyseur de *mysql* – y compris la commande **QUIT!** Cela peut être déroutant, surtout si vous ne savez pas qu'il faut absolument fournir un guillemet de fin, même pour annuler la saisie

1.3.3 Exemples de requêtes

Voici quelques exemples de requêtes classiques avec *MySQL*.

Certains des exemples utilisent la table 'shop' qui contient les prix de chaque article (numéro d'objet).

Supposons que chaque objet a un prix unique, et que le couple (item, trader) et une clé primaire pour ces lignes.

Vous pouvez créer cet exemple avec la table suivante :

```
mysql> \u test
Database changed
mysql> CREATE TABLE shop (
  -> article INT(4) UNSIGNED ZEROFILL DEFAULT '0000' NOT NULL,
  -> dealer CHAR(20) DEFAULT '' NOT NULL,
  -> price DOUBLE(16,2) DEFAULT '0.00' NOT NULL,
  -> PRIMARY KEY(article, dealer));
Query OK, 0 rows affected (0.22 sec)

mysql> INSERT INTO shop VALUES
  -> (1,'A',3.45),(1,'B',3.99),(2,'A',10.99),(3,'B',1.45),(3,'C',1.69),
  -> (3,'D',1.25),(4,'D',19.95);
Query OK, 7 rows affected (0.19 sec)
Records: 7 Duplicates: 0 Warnings: 0

mysql>
```

Les données pour l'exemple sont :

```
mysql> SELECT * FROM shop;
+-----+-----+-----+
| article | dealer | price |
+-----+-----+-----+
| 0001 | A      | 3.45 |
| 0001 | B      | 3.99 |
| 0002 | A      | 10.99 |
| 0003 | B      | 1.45 |
| 0003 | C      | 1.69 |
| 0003 | D      | 1.25 |
| 0004 | D      | 19.95 |
+-----+-----+-----+
7 rows in set (0.00 sec)
```

```
mysql>
```

1.3.3.1 Valeur maximale d'une colonne

"Quel est le plus grand numéro d'objet?"

```
mysql> SELECT MAX(article) AS article FROM shop;
+-----+
| article |
+-----+
```

```
|      4 |
+-----+
1 row in set (0.00 sec)
```

```
mysql>
```

1.3.3.2 La ligne qui contient le maximum d'une colonne

"Retrouver le prix, le vendeur et le numéro de l'objet le plus cher du magasin" En ANSI-SQL cela est très facilement fait avec un sous sélection :

```
SELECT article, dealer, price
FROM shop
WHERE price=(SELECT MAX(price) FROM shop)
```

Avec *MySQL* (et donc, sans les sous sélections), il faut le faire en deux étapes :

1. Retrouver la valeur maximale de la table, avec la commande SELECT.
2. Avec la valeur lue, créer la requête suivante :

```
mysql> SELECT article, dealer, price
-> FROM shop
-> WHERE price=19.95;
+-----+-----+-----+
| article | dealer | price |
+-----+-----+-----+
|    0004 | D      | 19.95 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql>
```

Une autre solution est de trier les objets par prix, et de lire la première ligne, avec la clause *MySQL* LIMIT :

```
mysql> SELECT article, dealer, price
-> FROM shop
-> ORDER BY price DESC
-> LIMIT 1;
+-----+-----+-----+
| article | dealer | price |
+-----+-----+-----+
|    0004 | D      | 19.95 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql>
```

Note : Avec cette méthode, on ne verra qu'un seul objet, même si il y a plusieurs objets de meme prix.

1.3.3.3 Maximum d'une colonne : par groupement

"Quel est le prix maximal d'un article?"

```
mysql> SELECT article, MAX(price) AS price
-> FROM shop
-> GROUP BY article;
+-----+-----+
| article | price |
+-----+-----+
|    0001 | 3.99 |
|    0002 | 10.99 |
|    0003 | 1.69 |
|    0004 | 19.95 |
```

```
+-----+-----+
4 rows in set (0.00 sec)
```

```
mysql>
```

1.3.3.4 La ligne contenant le maximum d'une colonne d'un groupe

"Pour chaque article, trouver le vendeur le plus cher."

En ANSI SQL on pourrait le faire avec une sous sélection, comme ceci :

```
SELECT article, dealer, price
FROM shop s1
WHERE price=(SELECT MAX(s2.price)
FROM shop s2
WHERE s1.article = s2.article)
```

Avec *MySQL* il vaut mieux le faire en deux étapes :

1. Retrouver la liste des (article,prix_maxima). La ligne contenant le maximum d'une colonne d'un groupe.
2. pour chaque article trouvé, retrouver la ligne correspondante, pour lire le nom du vendeur. price.

Cela peut se faire facilement avec une table temporaire :

```
mysql> INSERT INTO tmp SELECT article, MAX(price) FROM shop GROUP BY article;
Query OK, 4 rows affected (0.00 sec)
Records: 4 Duplicates: 0 Warnings: 0
```

```
mysql> SELECT shop.article, dealer, shop.price FROM shop, tmp
-> WHERE shop.article=tmp.article AND shop.price=tmp.price;
```

```
+-----+-----+-----+
| article | dealer | price |
+-----+-----+-----+
| 0001 | B      | 3.99 |
| 0002 | A      | 10.99 |
| 0003 | C      | 1.69 |
| 0004 | D      | 19.95 |
| 0001 | B      | 3.99 |
| 0002 | A      | 10.99 |
| 0003 | C      | 1.69 |
| 0004 | D      | 19.95 |
+-----+-----+-----+
8 rows in set (0.00 sec)
```

```
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> DROP TABLE tmp;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql>
```

Si vous n'utilisez pas de table temporaire, il vous faut verrouiller la table.

"Est ce qu'il est impossible de faire cela avec une seule requête?"

Oui, mais en utilisant une astuce qui s'appelle : "MAX-CONCAT trick" :

```
mysql> SELECT article,
-> SUBSTRING( MAX( CONCAT(LPAD(price,6,'0'),dealer) ), 7) AS dealer,
-> 0.00+LEFT( MAX( CONCAT(LPAD(price,6,'0'),dealer) ), 6) AS price
-> FROM shop
-> GROUP BY article;
+-----+-----+-----+
| article | dealer | price |
```

```

+-----+-----+-----+
| 0001 | B      | 3.99 |
| 0002 | A      | 10.99|
| 0003 | C      | 1.69 |
| 0004 | D      | 19.95|
+-----+-----+-----+
4 rows in set (0.00 sec)

```

```
mysql>
```

Le dernier exemple peut être fait de manière plus efficace, en effectuant la scission de la colonne au niveau du client ;

1.3.3.5 Utiliser des clés étrangères

Il n'y a pas besoin de clé étrangère pour joindre deux tables.

La seule chose que MySQL ne fait pas est de CHECK (vérifier) que les clés que vous utilisez existent vraiment dans la table que vous référencez, et qu'il n'efface pas de lignes dans une table avec une définition de clé étrangère. Si vous utilisez vos clés de manière habituelle, cela fonctionnera parfaitement.

```

mysql> CREATE TABLE persons (
  -> id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  -> name CHAR(60) NOT NULL,
  -> PRIMARY KEY (id)
  -> );

```

```
Query OK, 0 rows affected (0.00 sec)
```

```

mysql> CREATE TABLE shirts (
  -> id SMALLINT UNSIGNED NOT NULL AUTO_INCREMENT,
  -> style ENUM('t-shirt', 'polo', 'dress') NOT NULL,
  -> color ENUM('red', 'blue', 'orange', 'white', 'black') NOT NULL,
  -> owner SMALLINT UNSIGNED NOT NULL REFERENCES persons,
  -> PRIMARY KEY (id)
  -> );

```

```
Query OK, 0 rows affected (0.37 sec)
```

```
mysql> INSERT INTO persons VALUES (NULL, 'Antonio Paz');
```

```
Query OK, 1 row affected (0.00 sec)
```

```

mysql> INSERT INTO shirts VALUES
  -> (NULL, 'polo', 'blue', LAST_INSERT_ID()),
  -> (NULL, 'dress', 'white', LAST_INSERT_ID()),
  -> (NULL, 't-shirt', 'blue', LAST_INSERT_ID());

```

```
Query OK, 3 rows affected (0.00 sec)
```

```
Records: 3 Duplicates: 0 Warnings: 0
```

```
mysql> INSERT INTO persons VALUES (NULL, 'Lilliana Angelovska');
```

```
Query OK, 1 row affected (0.00 sec)
```

```

mysql> INSERT INTO shirts VALUES
  -> (NULL, 'dress', 'orange', LAST_INSERT_ID()),
  -> (NULL, 'polo', 'red', LAST_INSERT_ID()),
  -> (NULL, 'dress', 'blue', LAST_INSERT_ID()),
  -> (NULL, 't-shirt', 'white', LAST_INSERT_ID());

```

```
Query OK, 4 rows affected (0.00 sec)
```

```
Records: 4 Duplicates: 0 Warnings: 0
```

```
mysql> SELECT * FROM persons;
```

```

+-----+-----+-----+
| id | name                |
+-----+-----+-----+

```

```
| 1 | Antonio Paz          |
| 2 | Lilliana Angelovska |
+-----+
2 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM shirts;
+-----+-----+-----+-----+
| id | style  | color | owner |
+-----+-----+-----+-----+
| 1 | polo   | blue  | 1     |
| 2 | dress  | white | 1     |
| 3 | t-shirt| blue  | 1     |
| 4 | dress  | orange| 2     |
| 5 | polo   | red   | 2     |
| 6 | dress  | blue  | 2     |
| 7 | t-shirt| white | 2     |
+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

```
mysql> SELECT s.* FROM persons p, shirts s
-> WHERE p.name LIKE 'Lilliana%'
-> AND s.owner = p.id
-> AND s.color <> 'white';
+-----+-----+-----+-----+
| id | style | color | owner |
+-----+-----+-----+-----+
| 4 | dress | orange| 2     |
| 5 | polo  | red   | 2     |
| 6 | dress | blue  | 2     |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
mysql>
```


Chapitre 2

TP1 : Votre première base de données

Cette partie du TP est constituée d'une partie du chapitre 7 de la documentation française de MySQL disponible à l'adresse suivante <http://www.nexen.net>.

Il s'agit ici d'un TP d'initiation aux Bases de Données sous MySQL. Le but du TP est de vous faire prendre en main les commandes de base du langage SQL (Un SQL simplifié puisqu'il n'y a pas de notion de sous sélection.)

Votre travail consiste à tester toutes les commandes qui se trouve dans la suite du chapitre, et de vérifier les résultats obtenus par rapport à la solution indiquée sur le document.

2.1 Créer et utiliser une base de données

Si l'administrateur vous a créé une base de données, alors vous pouvez directement commencer à l'utiliser. Sinon, il vous faut la créer vous même :

```
mysql> CREATE DATABASE menagerie;
```

Sous Unix, les noms de base de données sont sensibles à la casse (contrairement aux mots clés SQL), donc il faudra faire référence à votre base de données sous le nom *menagerie*, et non pas *Menagerie*, *MENAGERIE* ou tout autre variante. Sous Windows, cette restriction ne s'applique pas, même si vous devez faire référence à vos bases et tables de la même manière tout au long d'une même commande).

Dans votre cas la base de données est déjà créée et vous ne possédez pas les droits nécessaires à la création de base de données.

Créer une base de données ne la sélectionne pas automatiquement. Il faut le faire explicitement. Pour faire de *menagerie* votre base courante, il faut utiliser la commande :

```
mysql> USE user_menagerie;
```

Database changed

La base n'a besoin d'être créée qu'une seule fois, mais il faudra la sélectionner à chaque fois que vous commencerez une session *mysql*. Il suffira alors d'utiliser la même commande que ci-dessus.

Alternativement, vous pouvez sélectionner une base dès la connexion, en passant le nom de la base après tous les paramètres de connexion :

```
yann@yoda:~/ $ mysql -h 193.50.119.130 -u user -p user_menagerie
```

```
Enter password: *****
```

Remarquez bien que *menagerie* n'est pas dans votre mot de passe. Si vous voulez transmettre votre mot de passe après l'option *-p*, vous devez le faire sans espace entre le mot de passe et l'option : (e.g., tel que *-pmypassword*, mais pas *-p mypassword*). Cependant, mettre votre mot de passe dans la ligne de connexion n'est pas très recommandé, car cela vous rend vulnérable à tous les mouchards qui pourraient être sur votre machine.

2.1.1 Créer une table

Créer une base de données est facile, mais, jusqu'à présent, c'est vide. La commande **SHOW TABLES** vous dira :

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

```
mysql>
```

La partie la plus difficile est le choix de la structure de votre base de données, et des tables dont vous aurez besoin, et quelles colonnes seront nécessaires.

Vous pouvez envisager de créer une table qui créera un enregistrement pour chacun de vos animaux. Cette table portera le nom de **animaux** et devrait contenir au minimum le nom de l'animal. Etant donné que le nom seul n'est pas vraiment intéressant, il faudra qu'il contienne aussi d'autres informations. Par exemple, si plusieurs personnes de votre famille ont des animaux domestiques, vous voudrez garder la liste de chaque maître. Vous voudrez peut être aussi conserver des informations basiques telles que le genre ou la race.

Et l'âge? Cela pourrait être intéressant à conserver, mais ce n'est pas une bonne chose à conserver dans une base de données. En effet, l'âge change tous les jours, et il faudrait changer constamment la base de données.

Au contraire, il est bien mieux de conserver la date de naissance. Alors, à chaque fois que vous aurez besoins de l'âge, il suffira de faire la différence entre la date du jour et la date de naissance. **MySQL** disposent de puissantes fonctions de calculs sur les dates. Enregistrer la date de naissance plutôt que l'âge a d'autres atouts :

Vous pourrez utiliser la base de données pour garder en mémoire les dates d'anniversaires de vos animaux (Si cela vous semble un peu idiot, remarquez bien que c'est exactement la même chose que de conserver la date d'anniversaire de vos clients, et de leur envoyer cette carte d'anniversaire à la spontanéité toute informatique).

Vous pourrez faire des calculs d'âge en relation avec d'autres dates. Par exemple, si vous enregistrer la date de mort, vous pourrez facilement calculer à quel âge est mort votre compagnon.

Votre imagination fertile vous permettra sûrement d'imaginer une foule d'informations utiles pour garnir la table **animaux** , mais les champs que nous venons d'identifier seront suffisant pour l'instant : le nom, le propriétaire, la race, le genre, la date de naissance et celle de mort.

Utilisez maintenant la fonction de création de table pour créer la votre :

```
mysql> CREATE TABLE animaux (nom VARCHAR(20), proprietaire VARCHAR(20),
-> espece VARCHAR(20), genre CHAR(1), naissance DATE, mort DATE);
Query OK, 0 rows affected (0.00 sec)
```

```
mysql>
```

VARCHAR est un bon choix pour le nom, le propriétaire et la race, car ces valeurs auront des longueurs variables. Les longueurs de ces colonnes n'ont pas besoin d'être toutes identiques, ni de valoir 20. Vous pouvez choisir n'importe quelle longueur entre 1 et 255, du moment que cela vous semble approprié (si vous vous trompez , vous pourrez toujours agrandir le champs avec la fonction **MySQL** : **ALTER TABLE**).

Le genre des animaux peu prendre de nombreuses formes, comme par exemple **"m"** et **"f"**, ou peut être **"male"** et **"femelle"**. Le plus simple sera d'utiliser les caractères **"m"** et **"f"**.

L'utilisation du type **DATE** pour représenter les dates de naissance **naissance** et de mort **mort** est un choix évident.

Maintenant que vous avez créer une table, , **SHOW TABLES** devrait être plus loquace :

```
mysql> SHOW TABLES;
+-----+
| Tables_in_menagerie |
+-----+
| animaux              |
+-----+
1 row in set (0.00 sec)
```

```
mysql>
```

Pour vérifier que la table a été créée comme vous le désiriez, utilisez la commande **DESCRIBE** :

```
mysql> DESCRIBE animaux;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| nom            | varchar(20)   | YES  |     | NULL    |       |
| propriétaire  | varchar(20)   | YES  |     | NULL    |       |
| espece        | varchar(20)   | YES  |     | NULL    |       |
| genre         | char(1)       | YES  |     | NULL    |       |
| naissance     | date          | YES  |     | NULL    |       |
| mort          | date          | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

Vous pouvez utiliser *DESCRIBE* à tout moment, par exemple, si vous oubliez les noms de colonnes ou leur type.

2.1.2 Charger des données dans une table

Après avoir créé votre table, il faut la remplir. La fonction *LOAD DATA* et *INSERT* remplissent cette fonction.

Supposons que les informations sur vos animaux soient décrites comme dans le tableau ci-dessous :

Remaquez bien que *MySQL* utilise un format de date de type *AAAA-MM-JJ* ; qui n'est pas le format standard.)

Etant donné que vous commencez avec une table vide, le meilleur moyen de remplir cette table est de créer un fichier texte, chaque ligne contenant les informations d'un animal, puis de le charger directement dans la table avec une seule commande.

Vous créez ainsi un fichier *animaux.txt* contenant un enregistrement par ligne, avec des valeurs séparées par des tabulations, et dans le même ordre que l'ordre dans lequel les colonnes ont été listées dans la commande *CREATE TABLE*. Pour les valeurs manquantes (comme par exemple, les genres inconnues, ou les dates de mort des animaux vivants), vous pouvez utiliser la valeur *NULL* . Vous la représenterez dans le texte avec *\N*.

```
Fluffy Harold chat f 1993-02-04 \N
Claws Gwen chat m 1994-03-17 \N
Buffy Harold chien f 1989-05-13 \N
Fang Benny chien m 1990-08-27 \N
Bowser Diane chien m 1998-08-31 1995-07-29
Chirpy Gwen oiseau f 1998-09-11 \N
Whistler Gwen oiseau \N 1997-12-09 \N
Slim Benny serpent m 1996-04-29 \N
Puffball Diane hamster f 1999-03-30 \N
```

Par exemple, l'enregistrement de l'oiseau Whistler ressemblera à ceci :

```
Whistler Gwen oiseau \N 1997-12-09 \N
```

Pour charger ce fichier '*animaux.txt*' dans la table *animaux* , utilisez la commande suivante :

```
mysql> LOAD DATA LOCAL INFILE "animaux.txt" INTO TABLE animaux;
Query OK, 9 rows affected (0.00 sec)
Records: 9 Deleted: 0 Skipped: 0 Warnings: 1

mysql>
```

Vous pourriez spécifier le type de chaque colonne et le marqueur de fin de ligne dans la commande *LOAD DATA* si vous le désiriez, mais les valeurs par défaut (tabulations et retour chariot) fonctionnent très bien ici. Pour n'ajouter qu'un seul enregistrement à la fois, la fonction *INSERT* est plus pratique : Dans sa forme la plus simple, vous fournissez les valeurs dans l'ordre des colonnes. Par exemple, si Diane reçoit un hamster du nom de Puffball, vous pourriez ajouter un nouvel enregistrement avec la commande suivante :

```
mysql> INSERT INTO animaux
-> VALUES ('Puffball','Diane','hamster','f','1999-03-30',NULL);
Query OK, 1 row affected (0.00 sec)
```

```
mysql>
```

Notez bien que les chaînes et les dates sont spécifiées avec des guillemets. De la même façon, vous pouvez insérer la valeur **NULL** directement pour représenter une valeur manquante. N'utilisez pas **\N** comme pour **LOAD DATA**.

À partir de cet exemple, vous voyez que la commande **INSERT** requiert nettement plus de frappe au clavier que la fonction **LOAD DATA**

2.1.3 Lire des informations dans une table

La commande **SELECT** sert à lire des informations d'une table. La forme générale est la suivante :

```
SELECT what_to_select
FROM which_table
WHERE conditions_to_satisfy
```

what_to_select indique ce que vous voulez afficher. Cela peut être une liste de champs, ou bien le joker ***** qui signifie "toutes les colonnes" **which_table** indique dans quelle table lire les informations. La clause **WHERE** est optionnelle. Si elle est présente, **conditions_to_satisfy** spécifie les conditions qu'une ligne doit remplir pour être retenue, et retournée.

Sélection toutes les données La forme la plus simple de **SELECT** permet d'obtenir la liste complète des ligne d'une table :

```
mysql> select * from animaux;
+-----+-----+-----+-----+-----+-----+
| nom      | proprietaire | espece | genre | naissance | mort      |
+-----+-----+-----+-----+-----+-----+
| Fluffy   | Harold       | chat   | f     | 1993-02-04 | NULL     |
| Claws    | Gwen        | chat   | m     | 1994-03-17 | NULL     |
| Buffy    | Harold       | chien  | f     | 1989-05-13 | NULL     |
| Fang     | Benny        | chien  | m     | 1990-08-27 | NULL     |
| Bowser   | Diane       | chien  | m     | 1998-08-31 | 1995-07-29 |
| Chirpy   | Gwen        | oiseau | f     | 1998-09-11 | NULL     |
| Whistler | Gwen        | oiseau | NULL  | 1997-12-09 | NULL     |
| Slim     | Benny        | serpent | m     | 1996-04-29 | NULL     |
| Puffball | Diane       | hamster | f     | 1999-03-30 | NULL     |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.00 sec)
```

```
mysql>
```

Cette forme de **SELECT** est utile pour passer en revue une table, comme par exemple, une table que vous viendriez de charger. Dans cet exemple, la table ci-dessus montre qu'il y a eu une erreur dans le fichier.

Bowser semble être né après être mort ! En consultant son dossier, vous vous apercevez que sa date correcte de naissance est 1989, et non pas 1998.

Il y a au moins deux façons de corriger cette erreur :

Editez le fichier '**animaux.txt**' pour corriger l'erreur, puis effacer la table ,et la recharger avec la **DELETE** et **LOAD DATA** :

```
mysql> DELETE FROM animaux;
Query OK, 0 rows affected (0.00 sec)

mysql> LOAD DATA LOCAL INFILE "animaux.txt" INTO TABLE animaux;
Query OK, 9 rows affected (0.00 sec)
Records: 9 Deleted: 0 Skipped: 0 Warnings: 1
```

```
mysql>
```

Cependant, en faisant cela, il vous faudra aussi insérer de nouveau la fiche de Puffball.

Ou bien, corriger seulement la fiche erronée avec une commande **UPDATE** :

```
mysql> UPDATE animaux SET naissance = "1989-08-31" WHERE nom = "Bowser";
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> SELECT * FROM animaux;
```

nom	proprietaire	espece	genre	naissance	mort
Fluffy	Harold	chat	f	1993-02-04	NULL
Claws	Gwen	chat	m	1994-03-17	NULL
Buffy	Harold	chien	f	1989-05-13	NULL
Fang	Benny	chien	m	1990-08-27	NULL
Bowser	Diane	chien	m	1989-08-31	1995-07-29
Chirpy	Gwen	oiseau	f	1998-09-11	NULL
Whistler	Gwen	oiseau	NULL	1997-12-09	NULL
Slim	Benny	serpent	m	1996-04-29	NULL
Puffball	Diane	hamster	f	1999-03-30	NULL

9 rows in set (0.00 sec)

```
mysql>
```

Dans cet exemple, on voit qu'il est facile de sélectionner toute la table. Mais généralement, ce n'est pas très pratique, surtout quand la table devient trop grande. En général, il s'agit de réponse à une question plus spécifique, pour laquelle il va falloir ajouter des contraintes sur les informations à retourner. Voyons maintenant quelques exemples de requêtes.

Sélectionner une partie des lignes Il est bien sûr possible de ne sélectionner quelques lignes dans une table. Mettons que vous souhaitiez vérifier que la nouvelle date de naissance de Bowser's a bien été prise en compte. Il suffit de sélectionner l'enregistrement de Bowser comme ceci :

```
mysql> SELECT * FROM animaux WHERE nom = "Bowser";
+-----+-----+-----+-----+-----+-----+
| nom    | proprietaire | espece | genre | naissance | mort    |
+-----+-----+-----+-----+-----+-----+
| Bowser | Diane       | chien  | m     | 1989-08-31 | 1995-07-29 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql>
```

Le résultat confirme bien que l'année de naissance est 1989, et non plus 1998.

Les comparaisons de chaîne sont généralement insensible à la casse : on aurait plus préciser le nom "bowser", "BOWSER", etc. Le résultat aurait été le même.

Vous pouvez faire des recherches sur d'autres colonnes que nom. Par exemple, si vous voulez savoir quels animaux sont nés 1998, faites un test sur la colonne naissance :

```
mysql> SELECT * FROM animaux WHERE naissance >= "1998-1-1";
+-----+-----+-----+-----+-----+-----+
| nom    | proprietaire | espece | genre | naissance | mort    |
+-----+-----+-----+-----+-----+-----+
| Chirpy | Gwen       | oiseau | f     | 1998-09-11 | NULL    |
| Puffball | Diane     | hamster | f     | 1999-03-30 | NULL    |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql>
```

Vous pouvez aussi combiner les conditions : par exemple, pour rechercher les chiennes

```
mysql> SELECT * FROM animaux WHERE espece = "chien" AND genre = "f";
+-----+-----+-----+-----+-----+-----+
| nom    | proprietaire | espece | genre | naissance | mort |
+-----+-----+-----+-----+-----+-----+
| Buffy | Harold      | chien  | f     | 1989-05-13 | NULL |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql>
```

La requête précédente utilisait l'opérateur logique AND (ET) Il y a aussi un opérateur OR (OU) :

```
mysql> SELECT * FROM animaux WHERE espece = "serpent" OR espece = "oiseau";
+-----+-----+-----+-----+-----+-----+
| nom      | proprietaire | espece | genre | naissance | mort |
+-----+-----+-----+-----+-----+-----+
| Chirpy   | Gwen        | oiseau | f     | 1998-09-11 | NULL |
| Whistler | Gwen        | oiseau | NULL  | 1997-12-09 | NULL |
| Slim     | Benny       | serpent | m     | 1996-04-29 | NULL |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
mysql>
```

AND et OR peut être utilisés dans la même requête. C'est alors une bonne idée d'utiliser des parenthèses pour préciser les regroupements :

```
mysql> SELECT * FROM animaux WHERE (espece = "chat" AND genre = "m")
-> OR (espece = "chien" AND genre = "f");
+-----+-----+-----+-----+-----+-----+
| nom    | proprietaire | espece | genre | naissance | mort |
+-----+-----+-----+-----+-----+-----+
| Claws  | Gwen        | chat   | m     | 1994-03-17 | NULL |
| Buffy  | Harold      | chien  | f     | 1989-05-13 | NULL |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql>
```

Sélectionner une colonne spécifique Il se peut que vous n'avez pas besoin de toutes les colonnes de votre table, mais juste de quelques colonnes. Il suffit alors de citer les colonnes qui vous intéressent. Par exemple, si vous ne voulez voir que les noms des animaux, avec leur date de naissance, il suffit de ne sélectionner que les colonnes nom et naissance :

```
mysql> SELECT nom, naissance FROM animaux;
+-----+-----+
| nom      | naissance |
+-----+-----+
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Buffy    | 1989-05-13 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Puffball | 1999-03-30 |
+-----+-----+
9 rows in set (0.00 sec)
```

```
mysql>
```

Pour lister les propriétaires d'animaux, utilisez la requête suivante :

```
mysql> SELECT proprietaire FROM animaux;
+-----+
| proprietaire |
+-----+
| Harold      |
| Gwen        |
| Harold      |
| Benny       |
| Diane       |
| Gwen        |
| Gwen        |
| Benny       |
| Diane       |
+-----+
9 rows in set (0.00 sec)
```

```
mysql>
```

Cependant, vous pouvez remarquer que cette requête simple affiche le champs propriétaire de chaque ligne, ce qui conduit à avoir des redondances (comme Gwen). Pour ne les voir apparaître qu'une seule fois, il faut utiliser le mot clé DISTINCT :

```
mysql> SELECT DISTINCT proprietaire FROM animaux;
+-----+
| proprietaire |
+-----+
| Harold      |
| Gwen        |
| Benny       |
| Diane       |
+-----+
4 rows in set (0.00 sec)
```

```
mysql>
```

Vous pouvez encore combiner une clause WHERE lors de la sélection de lignes et de colonnes Par exemple, pour obtenir les dates de naissances des chiens et des chats, utilisez la requête suivante :

```
mysql> SELECT nom, espece, naissance FROM animaux
-> WHERE espece = "chien" OR espece = "chat";
+-----+-----+-----+
| nom    | espece | naissance |
+-----+-----+-----+
| Fluffy | chat   | 1993-02-04 |
| Claws  | chat   | 1994-03-17 |
| Buffy  | chien  | 1989-05-13 |
| Fang   | chien  | 1990-08-27 |
| Bowser | chien  | 1989-08-31 |
+-----+-----+-----+
5 rows in set (0.01 sec)
```

```
mysql>
```

Trier les lignes Vous avez pu remarquer que les lignes précédentes ont été affichées dans un ordre aléatoire. Comme il est plus facile d'analyser une requête dont les lignes ont été triées, il vaut mieux trier ces lignes avec la clause ORDER BY.

Voici la liste des dates de naissances des animaux, classées par date :

```
mysql> SELECT nom, naissance FROM animaux ORDER BY naissance;
```

```

+-----+-----+
| nom      | naissance |
+-----+-----+
| Buffy    | 1989-05-13 |
| Bowser   | 1989-08-31 |
| Fang     | 1990-08-27 |
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Slim     | 1996-04-29 |
| Whistler | 1997-12-09 |
| Chirpy   | 1998-09-11 |
| Puffball | 1999-03-30 |
+-----+-----+
9 rows in set (0.00 sec)

```

mysql>

Pour inverser l'ordre de tri, ajoutez le mot clé DESC (descendant) après le nom de la colonne que vous classez.

```

mysql> SELECT nom, naissance FROM animaux ORDER BY naissance DESC;
+-----+-----+
| nom      | naissance |
+-----+-----+
| Puffball | 1999-03-30 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Claws    | 1994-03-17 |
| Fluffy   | 1993-02-04 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Buffy    | 1989-05-13 |
+-----+-----+
9 rows in set (0.00 sec)

```

mysql>

Vous pouvez faire des classements avec plusieurs critères de tri. Par exemple, pour trier les animaux pas espèce, puis par naissance pour chaque type d'animaux, utilisez la requête suivante :

```

mysql> SELECT nom, espece, naissance FROM animaux ORDER BY espece, naissance DESC;
+-----+-----+-----+
| nom      | espece  | naissance |
+-----+-----+-----+
| Claws    | chat    | 1994-03-17 |
| Fluffy   | chat    | 1993-02-04 |
| Fang     | chien   | 1990-08-27 |
| Bowser   | chien   | 1989-08-31 |
| Buffy    | chien   | 1989-05-13 |
| Puffball | hamster | 1999-03-30 |
| Chirpy   | oiseau  | 1998-09-11 |
| Whistler | oiseau  | 1997-12-09 |
| Slim     | serpent | 1996-04-29 |
+-----+-----+-----+
9 rows in set (0.00 sec)

```

mysql>

Notez bien que le mot clé DESC ne s'applique qu'à la colonne le précédent immédiatement (naissance) ; espece étant trié dans l'ordre ascendant.

Calculs sur les dates *MySQL* possède de puissantes fonctions pour effectuer des calculs sur les dates, comme par exemple, calculer un âge, ou extraire des parties de date.

Pour déterminer l'âge de chacun des animaux, il faut calculer la différence entre la naissance et la date courante. Puis, convertir ces deux dates en jours, et diviser le tout par 365, pour avoir le nombre d'année.

```
mysql> SELECT nom, (TO_DAYS(NOW())-TO_DAYS(naissance))/365 FROM animaux;
```

```
+-----+-----+
| nom      | (TO_DAYS(NOW())-TO_DAYS(naissance))/365 |
+-----+-----+
| Fluffy   | 9.50 |
| Claws    | 8.38 |
| Buffy    | 13.23 |
| Fang     | 11.94 |
| Bowser   | 12.93 |
| Chirpy   | 3.89 |
| Whistler | 4.65 |
| Slim     | 6.26 |
| Puffball | 3.35 |
+-----+-----+
```

```
9 rows in set (0.00 sec)
```

```
mysql>
```

Bien que cette requête fasse bien ce qu'on lui demande, il y a de la place pour quelques améliorations. En premier lieu, les résultats gagneraient à être classés. De plus, le titre de la colonne n'est pas très explicite.

Le premier problème peut être résolu avec une clause ORDER BY nom qui va classer par ordre alphabétique.

Pour régler le problème du titre, nous allons utiliser un alias.

```
mysql> SELECT nom, (TO_DAYS(NOW())-TO_DAYS(naissance))/365 AS age
-> FROM animaux ORDER BY nom;
```

```
+-----+-----+
| nom      | age  |
+-----+-----+
| Bowser   | 12.93 |
| Buffy    | 13.23 |
| Chirpy   | 3.89 |
| Claws    | 8.38 |
| Fang     | 11.94 |
| Fluffy   | 9.50 |
| Puffball | 3.35 |
| Slim     | 6.26 |
| Whistler | 4.65 |
+-----+-----+
```

```
9 rows in set (0.01 sec)
```

```
mysql>
```

Pour trier les résultats par âge plutôt que par nom, il suffit de le mettre dans la clause ORDER BY :

```
mysql> SELECT nom, (TO_DAYS(NOW())-TO_DAYS(naissance))/365 AS age
-> FROM animaux ORDER BY age;
```

```
+-----+-----+
| nom      | age  |
+-----+-----+
| Puffball | 3.35 |
| Chirpy   | 3.89 |
| Whistler | 4.65 |
| Slim     | 6.26 |
| Claws    | 8.38 |
| Fluffy   | 9.50 |
| Fang     | 11.94 |
+-----+-----+
```

```
| Bowser | 12.93 |
| Buffy | 13.23 |
+-----+-----+
9 rows in set (0.00 sec)
```

```
mysql>
```

Une requête similaire pourrait calculer l'âge de mort des animaux morts. Pour cela, vous allez déterminer les animaux morts, en testant la colonne mort à NULL. Puis, pour les valeurs non-NULL, calculez l'âge avec les colonnes mort et naissance :

```
mysql> SELECT nom, naissance, mort, (TO_DAYS(mort)-TO_DAYS(naissance))/365 AS age
-> FROM animaux WHERE mort!=0 ORDER BY age;
+-----+-----+-----+-----+
| nom | naissance | mort | age |
+-----+-----+-----+-----+
| Bowser | 1989-08-31 | 1995-07-29 | 5.91 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql>
```

```
mysql> SELECT nom, naissance, mort, (TO_DAYS(mort)-TO_DAYS(naissance))/365 AS age
-> FROM animaux WHERE mort IS NOT NULL ORDER BY age;
+-----+-----+-----+-----+
| nom | naissance | mort | age |
+-----+-----+-----+-----+
| Bowser | 1989-08-31 | 1995-07-29 | 5.91 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql>
```

La requête utilise mort IS NOT NULL plutôt que mort!= NULL car NULL est une valeur spéciale.

Cela est expliqué plus loin. Allez au paragraphe [2.1.3 Travailler avec la valeur NULL](#).

Et comment rechercher les animaux dont l'anniversaire sera le mois prochain ? Pour ce genre de calculs, year et day sont inutiles, il suffit d'extraire le mois de la colonne naissance . **MySQL** fournit plusieurs fonctions d'extraction comme par exemple YEAR(), MONTH() et DAY(). MONTH() est le plus approprié ici. Pour voir comment cela fonction, exécutez la commande suivante, qui naissance et MONTH(naissance) :

```
mysql> SELECT nom, naissance, MONTH(naissance) FROM animaux;
+-----+-----+-----+
| nom | naissance | MONTH(naissance) |
+-----+-----+-----+
| Fluffy | 1993-02-04 | 2 |
| Claws | 1994-03-17 | 3 |
| Buffy | 1989-05-13 | 5 |
| Fang | 1990-08-27 | 8 |
| Bowser | 1989-08-31 | 8 |
| Chirpy | 1998-09-11 | 9 |
| Whistler | 1997-12-09 | 12 |
| Slim | 1996-04-29 | 4 |
| Puffball | 1999-03-30 | 3 |
+-----+-----+-----+
9 rows in set (0.00 sec)
```

```
mysql>
```

Trouver les animaux dont la date de naissance est le mois prochain est facile. En supposant que nous soyons au mois d'avril. Alors, le mois est le 4, et il suffit de rechercher les animaux nés au mois de May (5), comme ceci :

```
mysql> SELECT nom, naissance FROM animaux WHERE MONTH(naissance) = 5;
```

```
+-----+-----+
| nom   | naissance |
+-----+-----+
| Buffy | 1989-05-13 |
+-----+-----+
1 row in set (0.00 sec)
```

mysql>

Il y a bien sur un cas particulier : décembre. Il ne suffit pas seulement d'ajouter 1 à numéro du mois courant et de chercher les dates d'anniversaires correspondantes, car personne ne naît au mois 13. À la place, il faut chercher les animaux qui sont nés au mois de janvier.

Vous pourriez écrire une requête qui fonctionne, quelque soit le mois courant. De cette façon, vous n'aurez pas à utiliser un numéro particulier de mois dans la requête. `DATE_ADD()` vous permettra d'ajouter une durée de temps à une date. Si vous ajoutez un mois à la date de `NOW()`, puis vous en sortez le mois avec `MONTH()`, le résultat sera bien le mois suivant.

```
mysql> SELECT nom, naissance FROM animaux
-> WHERE MONTH(naissance) = MONTH(DATE_ADD(NOW(), INTERVAL 1 MONTH));
+-----+-----+
| nom   | naissance |
+-----+-----+
| Chirpy | 1998-09-11 |
+-----+-----+
1 row in set (0.00 sec)
```

mysql>

Une autre manière de faire serait d'ajouter 1 au mois courant, puis d'utiliser la `(MOD)` pour "boucler" à la fin de l'année, et faire correspondre janvier et décembre :

```
mysql> SELECT nom, naissance FROM animaux
-> WHERE MONTH(naissance) = MOD(MONTH(NOW()),12) + 1;
+-----+-----+
| nom   | naissance |
+-----+-----+
| Chirpy | 1998-09-11 |
+-----+-----+
1 row in set (0.00 sec)
```

mysql>

Travailler avec la valeur NULL La valeur NULL peut se comporter de manière surprenante si vous l'utilisez. Conceptuellement, NULL signifie "valeur manquante" ou "valeur inconnue" et il est traité de manière légèrement différente des autres valeurs. Pour tester une valeur à NULL, vous ne pouvez pas utiliser les opérateurs de comparaison habituels, tels que `=`, `<` or `!=`. Pour vous en convaincre, essayez la requête suivante :

```
mysql> SELECT 1 = NULL, 1 != NULL, 1 < NULL, 1 > NULL;
+-----+-----+-----+-----+
| 1 = NULL | 1 != NULL | 1 < NULL | 1 > NULL |
+-----+-----+-----+-----+
| NULL | NULL | NULL | NULL |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

mysql>

Clairement, vous n'obtiendrez aucun résultat significatif de ces comparaisons. Utilisez les opérateurs `IS NULL` et `IS NOT NULL` :

```
mysql> SELECT 1 IS NULL, 1 IS NOT NULL;
```

```

+-----+-----+
| 1 IS NULL | 1 IS NOT NULL |
+-----+-----+
|          0 |          1 |
+-----+-----+
1 row in set (0.00 sec)

```

mysql>

Avec *MySQL*, 0 signifie faux et 1 signifie vrai.

Cette gestion spéciale de NULL explique pourquoi, dans la section précédente, il était nécessaire de savoir quels animaux étaient encore vivant, en utilisant mort IS NOT NULL à la place de mort != NULL.

Recherche de valeurs *MySQL* propose les méthodes de recherche standard du SQL, mais aussi les recherches à base d'expression régulière, similaire à celle utilisées dans les utilitaires Unix, tels que vi, grep et sed.

Les méthodes de recherche SQL vous permettent d'utiliser le caractère "_" pour remplacer n'importe quel caractère unique, et "%" pour remplacer n'importe quel nombre de caractères (y compris le caractère 0).

Les recherches SQL sont insensibles à la casse. Reportez vous aux exemples ci-dessous. Remarquez bien que l'on n'utilise pas = ou != mais plutôt LIKE ou NOT LIKE.

Recherche des noms commençant par "b" :

```

mysql> SELECT * FROM animaux WHERE nom LIKE "b%";
+-----+-----+-----+-----+-----+-----+
| nom      | proprietaire | espece | genre | naissance | mort      |
+-----+-----+-----+-----+-----+-----+
| Buffy    | Harold       | chien  | f      | 1989-05-13 | NULL      |
| Bowser   | Diane        | chien  | m      | 1989-08-31 | 1995-07-29 |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

mysql>

Recherche des noms finissant par "fy" :

```

mysql> SELECT * FROM animaux WHERE nom LIKE "%fy";
+-----+-----+-----+-----+-----+-----+
| nom      | proprietaire | espece | genre | naissance | mort      |
+-----+-----+-----+-----+-----+-----+
| Fluffy   | Harold       | chat   | f      | 1993-02-04 | NULL      |
| Buffy    | Harold       | chien  | f      | 1989-05-13 | NULL      |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

mysql>

Recherche des noms contenant "w" :

```

mysql> SELECT * FROM animaux WHERE nom LIKE "%w%";
+-----+-----+-----+-----+-----+-----+
| nom      | proprietaire | espece | genre | naissance | mort      |
+-----+-----+-----+-----+-----+-----+
| Claws    | Gwen        | chat   | m      | 1994-03-17 | NULL      |
| Bowser   | Diane        | chien  | m      | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen        | oiseau | NULL   | 1997-12-09 | NULL      |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

```

mysql>

Recherche des noms contenant exactement 5 caractères, utilisez le caractère "_" :

```
mysql> SELECT * FROM animaux WHERE nom LIKE "_____";
+-----+-----+-----+-----+-----+-----+
| nom    | proprietaire | espece | genre | naissance | mort |
+-----+-----+-----+-----+-----+-----+
| Claws  | Gwen         | chat   | m     | 1994-03-17 | NULL |
| Buffy  | Harold       | chien  | f     | 1989-05-13 | NULL |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql>
```

L'autre type de recherche disponible avec *MySQL* est les expressions régulières. Pour utiliser ce type de recherche, il faut ajouter les mots clé REGEXP et NOT REGEXP (ou RLIKE et NOT RLIKE, qui sont des synonymes).

Les caractéristiques des expressions régulières sont :

- "." remplace n'importe quel caractère qui n'apparaît qu'une fois.
- Une classe de caractères "[...]" remplace n'importe quel caractère qui apparaît dans les crochets. Par exemple, "[abc]" peut remplacer "a", "b" ou "c". Pour un intervalle de caractères, utilisez le tiret "." "[a-z]" remplace n'importe quelle lettre minuscule, et "[0-9]" remplace n'importe quel nombre.
- "*" remplace zéro ou plus occurrences du caractère le précédent immédiatement. Par exemple, "x*" remplace n'importe quelle nombre de "x". "[0-9]*" "" remplace n'importe quelle nombre de chiffres, et ".*" remplace n'importe quelle nombre de caractères.
- Les expressions régulières sont sensibles à la casse, mais vous pouvez utiliser une classe de caractères pour les rendre insensible à la casse. Par exemple, "[aA]" remplace n'importe quel "a", minuscule ou majuscule, et "[a-zA-Z]" remplace n'importe quelle lettre, minuscule ou majuscule.
- La recherche est positive, si elle est vérifiée à n'importe quel endroit de la valeur (en SQL, ce n'est vrai que sur la valeur entière).
- Pour contraindre une expression au début ou à la fin de la valeur, utilisez les caractères spéciaux "^" pour le début ou "\$" pour la fin.

Pour illustrer le fonctionnement des expressions régulières, les requêtes précédentes ont été réécrites en utilisant les expressions régulières.

Recherche des noms commençant par "b" : on utilise "^" pour indiquer le début de la valeur, et "[bB]" pour rechercher indifféremment, "b" minuscule ou majuscule.

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "^[bB]";
+-----+-----+-----+-----+-----+-----+
| nom    | proprietaire | espece | genre | naissance | mort |
+-----+-----+-----+-----+-----+-----+
| Buffy  | Harold       | chien  | f     | 1989-05-13 | NULL |
| Bowser | Diane        | chien  | m     | 1989-08-31 | 1995-07-29 |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql>
```

Recherche des noms finissant par "fy" : on utilise "\$" pour indiquer la fin de la valeur

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "fy$";
+-----+-----+-----+-----+-----+-----+
| nom    | proprietaire | espece | genre | naissance | mort |
+-----+-----+-----+-----+-----+-----+
| Fluffy | Harold       | chat   | f     | 1993-02-04 | NULL |
| Buffy  | Harold       | chien  | f     | 1989-05-13 | NULL |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql>
```

Recherche des noms contenant "w" :, on utilise "[wW]" pour rechercher les "w", 'minuscule ou majuscule :

```
mysql> SELECT * FROM animaux WHERE nom REGEXP "[wW]";
+-----+-----+-----+-----+-----+-----+
```

```

| nom      | proprietaire | espece | genre | naissance | mort      |
+-----+-----+-----+-----+-----+-----+
| Claws    | Gwen         | chat   | m     | 1994-03-17 | NULL     |
| Bowser   | Diane        | chien  | m     | 1989-08-31 | 1995-07-29 |
| Whistler | Gwen         | oiseau | NULL  | 1997-12-09 | NULL     |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)

```

```
mysql>
```

Etant donné qu'une expression régulière est vrai si elle est vrai sur une partie d'une valeur, il n'est pas besoin de caractères spéciaux.

Recherche des noms contenant exactement 5 caractères, utilisez "^" et "\$" pour indiquer le début et la fin de la chaîne, et 5 fois "." pour les 5 caractères.

```

mysql> SELECT * FROM animaux WHERE nom REGEXP "^.....$";
+-----+-----+-----+-----+-----+-----+
| nom      | proprietaire | espece | genre | naissance | mort      |
+-----+-----+-----+-----+-----+-----+
| Claws    | Gwen         | chat   | m     | 1994-03-17 | NULL     |
| Buffy    | Harold       | chien  | f     | 1989-05-13 | NULL     |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

```
mysql>
```

Vous auriez pu aussi utiliser l'opérateur "n" "n-fois" :

```

mysql> SELECT * FROM animaux WHERE nom REGEXP "^.{5}$";
+-----+-----+-----+-----+-----+-----+
| nom      | proprietaire | espece | genre | naissance | mort      |
+-----+-----+-----+-----+-----+-----+
| Claws    | Gwen         | chat   | m     | 1994-03-17 | NULL     |
| Buffy    | Harold       | chien  | f     | 1989-05-13 | NULL     |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

```

```
mysql>
```

Compter les lignes Les bases de données sont souvent utilisées pour répondre aux questions du type : "combien de fois une information est-elle enregistrée dans une table?". Par exemple, vous pouvez souhaiter connaître le nombre d'animaux que vous avez, ou le nombre d'animaux de chaque propriétaire, ou encore toutes sortes de statistiques sur les animaux.

Pour compter le nombre total d'animaux que vous avez, il suffit de compter le nombre de ligne dans la table animaux , puisqu'il y a un enregistrement par animal. La fonction COUNT() compte le nombre de ligne non-NULL. Votre requête ressemblera alors à :

```

mysql> SELECT COUNT(*) FROM animaux;
+-----+
| COUNT(*) |
+-----+
|          9 |
+-----+
1 row in set (0.00 sec)

```

```
mysql>
```

Précédemment, vous avez recherché les noms des propriétaires d'animaux. Vous pouvez utiliser la fonction COUNT() pour connaître le nombre d'animaux que chaque propriétaire a :

```
mysql> SELECT proprietaire, COUNT(*) FROM animaux GROUP BY proprietaire;
```

```

+-----+-----+
| proprietaire | COUNT(*) |
+-----+-----+
| Benny       |         2 |
| Diane       |         2 |
| Gwen        |         3 |
| Harold      |         2 |
+-----+-----+
4 rows in set (0.00 sec)

```

mysql>

Remarques : l'utilisation de la clause GROUP BY qui rassemble les lignes par proprietaire. Sans cette clause, vous obtenez le message d'erreur suivant :

```

mysql> SELECT proprietaire, COUNT(proprietaire) FROM animaux;
ERROR 1140: Mixing of GROUP columns (MIN(),MAX(),COUNT()...) with no GROUP columns is illegal if there
mysql>

```

COUNT() et GROUP BY sont utiles pour caractériser vos informations dans de nombreuses situations : Les exemples suivant effectuent des statistiques sur vos animaux :

Nombre d'animaux par espèce :

```

mysql> SELECT espece, COUNT(*) FROM animaux GROUP BY espece;
+-----+-----+
| espece | COUNT(*) |
+-----+-----+
| chat   |         2 |
| chien  |         3 |
| hamster |         1 |
| oiseau |         2 |
| serpent |         1 |
+-----+-----+
5 rows in set (0.00 sec)

```

mysql>

Nombre d'animaux par genre :

```

mysql> SELECT genre, COUNT(*) FROM animaux GROUP BY genre;
+-----+-----+
| genre | COUNT(*) |
+-----+-----+
| NULL  |         1 |
| f     |         4 |
| m     |         4 |
+-----+-----+
3 rows in set (0.00 sec)

```

mysql>

(Dans cette réponse, NULL indique "genre inconnu.")

Nombre d'animaux par espece et genre :

```

mysql> SELECT espece, genre, COUNT(*) FROM animaux GROUP BY espece, genre;
+-----+-----+-----+
| espece | genre | COUNT(*) |
+-----+-----+-----+
| chat   | f     |         1 |
| chat   | m     |         1 |
| chien  | f     |         1 |

```

```

| chien | m | 2 |
| hamster | f | 1 |
| oiseau | NULL | 1 |
| oiseau | f | 1 |
| serpent | m | 1 |
+-----+-----+-----+
8 rows in set (0.00 sec)

```

```
mysql>
```

Il n'y a pas besoin d'utiliser la table entière avec la fonction COUNT(). Par exemple, la requête précédente effectuée sur la population de chien et de chat devient :

```

mysql> SELECT espece, genre, COUNT(*) FROM animaux
-> WHERE espece = "chien" OR espece = "chat"
-> GROUP BY espece, genre;
+-----+-----+-----+
| espece | genre | COUNT(*) |
+-----+-----+-----+
| chat   | f     | 1         |
| chat   | m     | 1         |
| chien  | f     | 1         |
| chien  | m     | 2         |
+-----+-----+-----+
4 rows in set (0.00 sec)

```

```
mysql>
```

Ou, pour avoir le nombre d'animaux par genre, et pour les espèces connues :

```

mysql> SELECT espece, genre, COUNT(*) FROM animaux
-> WHERE genre IS NOT NULL
-> GROUP BY espece, genre;
+-----+-----+-----+
| espece | genre | COUNT(*) |
+-----+-----+-----+
| oiseau | f     | 1         |
| chat   | f     | 1         |
| chat   | m     | 1         |
| chien  | f     | 1         |
| chien  | m     | 2         |
| hamster | f    | 1         |
| serpent | m    | 1         |
+-----+-----+-----+

```

2.1.4 Utiliser plus d'une table

La table animaux contient la liste des animaux que vous avez. Vous pourriez vouloir enregistrer d'autres informations à leur sujet, telles que des événements de leur vie, comme les visites chez le vétérinaire, ou les dates des portées de petits : vous avez besoin d'une autre table. À quoi va t elle ressembler ?

- Elle va devoir contenir les noms des animaux, pour savoir à qui c'est arrivé.
- Il faut une date pour l'événement.
- Il faut un champs de description des événements
- Il faut aussi un champs de catégorie d'événement.

Avec ces indications, la requête de CREATE TABLE va ressembler à ceci :

```

mysql> CREATE TABLE event (nom VARCHAR(20), date DATE,
-> type VARCHAR(15), remark VARCHAR(255));
Query OK, 0 rows affected (0.00 sec)

```

```
mysql>
```


Comme pour la table animaux table, il est plus facile de charger les premières valeurs à partir d'un fichier, dont les champs sont délimités avec des tabulations (animaux.txt) :

```
Fluffy 1995-02-04 naissance 4 chatons, 3 femelles, 1 male
Buffy 1991-05-17 naissance 5 chiots, 2 femelles, 3 male
Buffy 1992-07-13 naissance 3 chiots, 3 femelles
```

Chargez les informations comme ceci :

```
mysql> LOAD DATA LOCAL INFILE "event.txt" INTO TABLE event;
Query OK, 3 rows affected (0.00 sec)
Records: 3 Deleted: 0 Skipped: 0 Warnings: 1
```

mysql>

```
mysql> select * from event;
```

nom	date	type	remark
Fluffy	1995-02-04	naissance	4 chatons, 3 femelles, 1 male
Buffy	1991-05-17	naissance	5 chiots, 2 femelles, 3 male
Buffy	1992-07-13	naissance	3 chiots, 3 femelles

3 rows in set (0.00 sec)

mysql>

Etant donné ce que vous avez appris avec les requêtes sur la table animaux table, vous devriez être capable d'exécuter des requêtes sur la table event ; les principes sont les mêmes. Mais la table event pourrait se révéler insuffisante pour répondre à vos questions.

Supposons que vous voulez avoir l'âge des animaux lorsqu'ils ont eu leur portée. La table event indique quand ils ont eu leur portée, mais pour calculer l'âge de la mère, il faut aussi sa date de naissance. Etant donné que cette date est stockée dans la table animaux, vous avez besoin des deux tables dans la même requête :

```
mysql> SELECT animaux.nom, (TO_DAYS(date) - TO_DAYS(naissance))/365 AS age, remark
-> FROM animaux, event
-> WHERE animaux.nom = event.nom AND type = "portée";
```

nom	age	remark
Fluffy	2.00	4 chatons, 3 femelles, 1 male
Buffy	2.01	5 chiots, 2 femelles, 3 male
Buffy	3.17	3 chiots, 3 femelles

3 rows in set (0.00 sec)

mysql>

Il faut remarquer plusieurs choses à propos de cette requête :

- La clause FROM est une liste contenant les noms des deux tables, car la requête clause va chercher des informations dans ces deux tables.
- Lorsque vous combinez des informations entre plusieurs tables, il faut spécifier comment les lignes vont correspondre. Ici, la correspondance est simple, et basée sur la colonne nom. La requête utilise une clause WHERE pour rechercher et assortir les valeurs des deux tables, avec la colonne nom.
- Etant donné que les deux tables ont une colonne nom il faut préciser la table d'appartenance de ces colonnes à chaque référence. C'est facilement faisable en ajoutant simplement le nom de la table devant le nom de la colonne.

Les regroupements sont aussi possibles sur une même table. Cela revient à comparer des valeurs d'une table avec d'autres valeurs de la même table. Par exemple, pour marier vos animaux entre eux, vous pouvez faire un regroupement de la table animaux avec elle-même pour rechercher les males et femelles de la même espèce :

```
mysql> SELECT p1.nom, p1.genre, p2.nom, p2.genre, p1.espece
-> FROM animaux AS p1, animaux AS p2
-> WHERE p1.espece = p2.espece AND p1.genre = "f" AND p2.genre = "m";
+-----+-----+-----+-----+-----+
| nom   | genre | nom   | genre | espece |
+-----+-----+-----+-----+-----+
| Fluffy | f     | Claws | m     | chat   |
| Buffy  | f     | Fang  | m     | chien  |
| Buffy  | f     | Bowser | m     | chien  |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

Dans cette requête, plusieurs alias sont définis pour pouvoir faire référence à chaque instance de la table, et à la bonne colonne.

2.2 Informations sur les bases de données et tables

Que se passe-t-il si vous oubliez le nom d'une base de données, d'une table ou la structure d'une table donnée ? *MySQL* résoud ce problème avec plusieurs commandes qui fournissent des informations sur les bases et les tables.

Vous avez déjà rencontré `SHOW DATABASES`, qui liste les bases gérées par le serveur. Pour connaître la base de données courante, utilisez `DATABASE()` :

```
mysql> SELECT DATABASE();
+-----+
| DATABASE() |
+-----+
| menagerie  |
+-----+
1 row in set (0.00 sec)

mysql>
```

Si vous n'avez pas de base sélectionnée, le résultat est vide.

Pour connaître les tables de la base de données courante, (par exemple, si vous n'êtes pas sûr du nom d'une table), utilisez la commande suivante :

```
mysql> SHOW TABLES;
+-----+
| Tables_in_menagerie |
+-----+
| animaux              |
| event                |
+-----+
2 rows in set (0.00 sec)

mysql>
```

Pour connaître les colonnes d'une table, (par exemple, si vous n'êtes pas sûr du nom d'une colonne), utilisez la commande suivante :

```
mysql> DESCRIBE animaux;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| nom            | varchar(20)  | YES  |     | NULL    |       |
| propriétaire  | varchar(20)  | YES  |     | NULL    |       |
| espece        | varchar(20)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
```

```
| genre      | char(1) | YES | | NULL | | |
| naissance | date   | YES | | NULL | | |
| mort      | date   | YES | | NULL | | |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

mysql>

Field donne le nom de la colonne, Type est le type de données, Null indique si la colonne accepte la valeur NULL ou pas, Key indique que la colonne est manual_tocée, et Default indique la valeur par défaut de la colonne.

Si vous avez des manual_toc sur une table, SHOW manual_toc FROM tbl_nom fournit la liste des informations les concernant.

Cette page est laissée blanche intentionnellement

Chapitre 3

TP 2 : Une base de données de publications

3.1 Introduction

Il s'agit de modéliser la liste des publications d'un laboratoire. Notre modèle conceptuel des données aura l'architecture suivante :

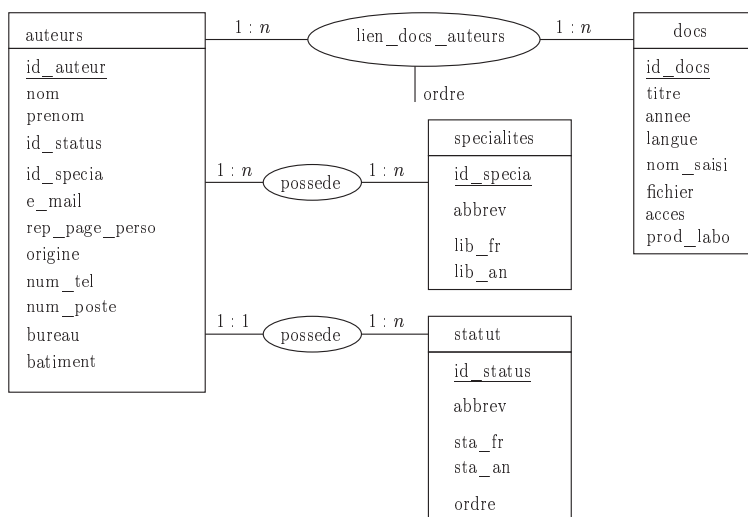


FIG. 3.1 – Modèle conceptuel de la base de données

Ceci conduit à la définition du modèle relationnelle suivant :

- auteurs (id_auteur, nom, prenom, id_status, id_specia, e_mail, rep_pages_perso, origine, num_telephone, num_poste, bureau, bâtiment) ;
- docs (id_docs, titre, annee, type, langue, fichier, acces) ;
- lien_docs_auteurs (id_docs, id_auteur, ordre) ;
- specialites (id_specia, abbrev, lib_fr, lib_an) ;
- status (id_status, abbrev, sta_fr, sta_an, ordre) ;

Note : Vous écrirez les lignes de commandes nécessaires dans les espaces blancs laissés à cet effet. Il est aussi évident que chaque commande devra être suivie d'une vérification. Par exmple lors de la création d'une table vous devez vérifier que cette dernière à bien été crée avec les champs désirées.

3.2 Création des tables

Dans un premier temps vous créerez les différentes tables nécessaires en choisissant judicieusement les champs (nom et type).

3.2.1 La table specialites

3.2.2 La table statuts

3.2.3 La table auteurs

3.2.4 La table docs

3.2.5 La table lien_docs_auteurs

3.3 Insertion des données

3.3.1 Dans la table specialites

id_specia	abbrev	lib_fr	lib_an
1	INFO	Informatique	Informatic
2	MECA	Mécanique	Mecanic
3	AUTO	Automatique	Automatic
4	PSYCHO	Psychologie	Psychology
5	ERGO	Ergonomie	Ergonomic
6	GEN	Général	General

3.3.2 Dans la table statuts

id_status	abbrev	sta_fr	sta_an	ordre
1	PR	Professeur	Professor	1
2	MCF	Maître de conférences	Assistant Professor	20
3	DR	Directeur de recherche	Research Director	5
4	CR	Chargé de recherche	Researcher	15
5	TH	Doctorant	Doctoral Student	55
6	ETU	Etudiant	Student	65
7	DEA	Etudiant en DEA	Student	60
8	IR	Ingénieur de recherche	Research Engineer	45
9	ASS	Assistant	Assistant	25
10	Autre	Autre	Other	80
11	MON	Moniteur	Monitor	52
12	PAST	PAST	Assistant Professor	40
13	ATER	ATER	Doctoral Student	30
14	SEC	Secrétaire	Secretary	70
15	DOC	Docteur	Doctor	35
16	MC-HDR	Maître de conférences HDR	Assistant Professor HDR	10
17	PRAG	PRAG	Assistant Professor	23
18	AI	Assistant ingénieur	Assistant Engineer	51

3.3.3 Dans la table auteurs

id_auteur	nom	prenom	id_status	id_specia
-----------	-----	--------	-----------	-----------

1	MORERE	Yann	2	3
2	DURANT	Alain	1	3
3	DUPONT	Guy	16	3
4	DUSCHMOLL	Olivier	2	3
5	DUMOULIN	Stéphane	2	1
6	DUPIERRE	Odile	2	3
7	DUTILLEUL	Abdallah	2	3
8	DUSAPIN	Pierre	2	3

e_mail	rep_pages_perso	origine	num_telephone
morere@imaginaire.org	http://ymoreere.multimania.com	Inconnu	555 555 554
durant@imaginaire.org	NULL	Inconnu	555 555 556
dupont@imaginaire.org	NULL	Inconnu	555 555 555
duscmoll@imaginaire.org	NULL	Inconnu	555 555 553
dumoulin@imaginaire.org	http://www.imaginaire.org/~dumoulin	Inconnu	555 555 552
dupierre@imaginaire.org	NULL	Inconnu	555 555 557
dutilleul@imaginaire.org	NULL	Inconnu	555 555 559
dusapin@imaginaire.org	NULL	Inconnu	555 555 560

num_poste	bureau	batiment
55 54	4	Imag2
55 56	1	Imag2
55 55	2	Imag2
55 53	9	Imag2
55 52	8	Imag2
55 57	6	Imag2
55 59	12	Imag2
55 60	11	Imag2

3.3.4 Dans la table docs

id_docs	titre	annee	type	langue	fichier	acces
1	Mise en place d'un serveur WEB	2002	article	française		0
2	La culture du blé sous marin	2012	article	française		0

3.3.5 Dans la table lien_docs_auteurs

id_docs	id_auteur	ordre
1	1	2
1	2	1
2	5	1
2	6	3
2	7	1

3.4 Insertion des données par fichier

3.4.1 Ajout d'auteurs

Créer le fichier texte `auteurs.txt` contenant les données nécessaires afin d'ajouter les auteurs suivants :

```

+-----+-----+-----+-----+-----+
| POULAIN   | Amélie   |      2 |      3 | poulain@imaginaire.org |
| LUKE      | Lucky    |     16 |      1 | luke@imaginaire.org   |
| VAILLANT  | Michel   |      2 |      3 | vaillant@imaginaire.org |
| HOCHET    | Ric      |      1 |      1 | hochet@imaginaire.org |
+-----+-----+-----+-----+-----+
| http://www.poulain.com           | Inconnu | 555 555 669 | 56 69 | 4 | Imag2 |
| http://www.luckyluke.com         | Inconnu | 555 555 668 | 56 70 | 4 | Imag2 |
| http://www.michelvaillant.com    | Inconnu | 555 555 669 | 56 71 | 4 | Imag2 |
| http://www.ricochet.com          | Inconnu | 555 555 669 | 56 72 | 4 | Imag2 |
+-----+-----+-----+-----+-----+

```

3.4.2 Ajout de publications

Ajoutez les publications suivantes par l'intermédiaire d'un fichier texte `docs.txt`.

```

+-----+-----+-----+-----+-----+
|      3 | Les Bases de Données : Mise en Oeuvre | 2002 | article | française |      | 0 |
|      4 | Le langage C/C++                       | 2001 | article | française |      | 0 |
|      5 | Le traitement du signal                 | 2000 | revue   | française |      | 0 |
|      6 | A quick introduction to MySQL           | 2002 | article | anglaise |      | 0 |
+-----+-----+-----+-----+-----+

```

3.4.3 Ajout de liens documents auteurs

Ajoutez dans la table `lien_docs_auteurs` les références suivantes afin de relier les auteurs aux articles qu'ils ont écrits comme décrits dans le tableau suivant ;

nom	prenom	ordre	titre
POULAIN	Amélie	1	Les Bases de Données : Mise en Oeuvre
VAILLANT	Michel	2	Les Bases de Données : Mise en Oeuvre
HOCHET	Ric	3	Les Bases de Données : Mise en Oeuvre
LUKE	Lucky	1	Le langage C/C++
DUMOULIN	Stéphane	2	Le langage C/C++
MORERE	Yann	3	Le langage C/C++
DUSAPIN	Pierre	4	Le langage C/C++
VAILLANT	Michel	1	Le traitement du signal
POULAIN	Amélie	2	Le traitement du signal
HOCHET	Ric	3	Le traitement du signal
DUPIERRE	Odile	1	A quick introduction to MySQL
DUSCHMOLL	Olivier	2	A quick introduction to MySQL
DUTILLEUL	Abdallah	3	A quick introduction to MySQL

3.5 Les Requêtes

3.5.1 Requêtes simples

Donnez tous les articles de la base ainsi que tous leurs champs Réponse :

Donnez tous les informations disponibles sur les auteurs Réponse :

Donnez les noms et prénoms de tous les auteurs d'articles Réponse :

Donnez toutes les informations disponibles sur M. VAILLANT Réponse :

Donnez le numéro de téléphone et le site web de Mlle POULAIN Réponse :

Donnez les noms et prénoms de tous les auteurs d'articles triés par nom dans l'ordre alphabétique puis à l'inverse de l'ordre alphabétique Réponse :

Comptez le nombre d'auteurs, puis de publications Réponse :

3.5.2 Requêtes de mises à jour

Listez dans un premier temps, les auteurs qui n'ont pas de site web, et mettez à jours ces champs (avec des adresses de site que vouschoisirrez Réponse :

3.5.3 Requêtes de selection de lignes

Listez tous les auteurs qui sont dans les bureaux supérieurs à 10 Réponse :

Listez tous les auteurs qui sont dans les bureaux 4, 8 ou 9 Réponse :

Listez tous les auteurs qui sont Maître de Conférences et qui sont automaticiens Pour information Maître de Conférences est équivalent à `id_status = 2` et la spécialité automatique est équivalent à `id_specia=3`. Réponse :

3.5.4 Sélection de colonne, comptage et tri

Listez tous les documents (titre et langue) en les classant par ordre décroissant d'année Réponse :

Listez les auteurs par leurs noms et prénoms, en les classant par, nom puis prénom et bureau
Réponse :

Donnez le nombre total d'auteurs puis de documents Réponse :

Donnez tous les auteurs regroupés par statuts Réponse :

3.5.5 Requêtes de recherche de valeurs

Recherche des noms d'auteurs commençant par "D" Réponse :

Recherche des noms d'auteurs se terminant pas "re" Réponse :

Recherche les noms des auteurs qui ont 6 caractères Réponse :

3.5.6 Requête avancée

Donnez les noms et prénoms des auteurs qui ont écrit le document "Le langage C/C++" Réponse :

Donnez les noms, prénoms statuts des auteurs classés par intitulé de statut Réponse :

Donnez les noms, prénoms, statuts et spécialités des auteurs classés par intitulé de spécialités

Réponse :

Donnez le nombre d'auteurs par publication Réponse :

Cette page est laissée blanche intentionnellement

Chapitre 4

Petit cours de HTML

Ce petit cours est tiré d'une page web disponible à l'adresse suivante <http://www.linternaute.com/communiq/pperso/debutant/html.shtml>.

Derrière ces quatre lettres barbares, se cache le langage informatique vedette du Web.

Pour être tout à fait honnête, vous n'êtes pas obligé de connaître l'HTML (HyperText Markup Language) en français "langage de balisage hypertexte", pour créer votre site perso. Si vous utilisez un éditeur WYSIWYG (What You See Is What You Get, en français "ce que vous voyez -à l'écran- est ce que vous obtenez -en ligne"), le logiciel traduit automatiquement vos désirs en langage html (voir la liste des différents éditeurs sur JDNet Téléchargement). Néanmoins, connaître les bases du html peut s'avérer très utile lorsque votre logiciel d'édition, forcément imparfait, ne vous "obéit pas au doigt et à l'oeil" et laisse par exemple un espace indésirable entre deux pavés de texte, un alignement peu heureux dans un tableau ou encore une couleur de lien qui s'accorde mal avec la couleur de fond de votre page.

L'HTML, qu'est-ce que c'est ? L'HTML n'est pas un langage de programmation, il s'apparente plutôt à un outil de mise en forme des pages web (texte et images). C'est ce langage qui est utilisé pour créer toutes les pages web que vous voyez lorsque vous surfez avec Internet Explorer ou Netscape Communicator.

Si vous ne savez à quoi ressemble "la bête", prenez n'importe quelle page web et regardez son "code source" dans Explorer ou Communicator : il suffit pour cela d'aller dans le menu "Affichage" et de cliquer sur "Source" ou "Source de la page". Si c'est la première fois que vous faites cette manipulation, vous risquez d'être un peu affolé par ces centaines et centaines de lignes de codes qui ressemblent à du javanais. Mais une fois assimilé les bases du HTML, tout devient parfaitement limpide ou presque.

Apprendre l'HTML, à quoi ça sert ? Lorsque l'on comprend ce qui se passe "sous le capot" de son site, on se sent un peu moins bête. Et on peut agir directement dans le code pour améliorer la qualité de ses pages. Ajoutez à cela que certains de ces éditeurs wysiwyg n'ont pas de version française, il vaut donc mieux connaître les mots anglais usuels utilisés dans les menus pour savoir s'en servir. Enfin, nul besoin d'avoir fait une grande école ou d'être un informaticien chevronné pour comprendre les bases du html et l'utiliser, il suffit simplement de connaître trois mots d'anglais et de disposer de quelques heures dans son emploi du temps.

Le HTML en dix leçons

Pour votre initiation, L'Internaute vous propose 10 leçons simples, appuyées par des exemples qui vous permettront de connaître les bases. Vous pourrez vous exercer en tapant le code HTML dans un "bloc-notes" ou un "wordpad". Pour voir le résultat de votre travail, vous enregistrerez votre document au format HTML (soit, "nomdudocument.html"). Fermez-le. Puis, ouvrez-le pour voir la page telle qu'elle apparaît dans un navigateur web.

4.1 Le principe des balises

L'HTML repose entièrement sur un système de **balises** (ou "tags" en anglais). Ce sont ces balises qui permettent d'enclencher des actions : par exemple, mettre un texte ou des mots en gras, créer un paragraphe, un tableau...

Votre page HTML se présente ainsi comme une succession de balises dites "ouvrantes" `< >` et de balises "fermantes" caractérisées par un slash `</ >`. Ces balises délimitent la portée de l'action enclenchée.

Certaines balises comportent des **attributs** (qu'on appelle aussi paramètres ou propriétés) qui permettent par exemple de choisir la couleur de fond de la page et des liens hypertextes, de choisir la taille, la police et la couleur des caractères... Ces attributs sont placés à l'intérieur de la balise.

Exemple :

```
<html>
<body>
Ceci est le texte de mon site.
</body>
</html>
```

Remarque : Attention, en cas d'instructions multiples (un texte qui serait en gras et en italique par exemple), vous ne pouvez pas croiser les différentes balises, il faut respecter un ordre très précis : la première ouverte <html> est la dernière fermée, la seconde ouverte <body> est l'avant-dernière fermée...

4.2 La structure des pages

Votre page web se décompose en deux parties, la **tête** ("head") et le **corps** ("body") : la première permet d'inclure du texte non visible lorsque la page est en ligne. Ces informations, qui doivent être inscrites sur chacune de vos pages web, servent principalement à référencer votre site auprès des moteurs de recherche et donc à lui assurer une bonne visibilité au milieu des millions de pages web existantes. Notez qu'aujourd'hui, les balises "description" et "mots-clés" ne sont plus prises en compte par les moteurs de recherche dans l'indexation des sites mais continuent en pratique à être renseignées (pour plus de détails, voir notre dossier sur le référencement).

La "tête" est comprise entre les balises <HEAD> et </HEAD>

Elle comporte les éléments suivants :

- le **titre** ("title") de la page compris entre les balises <TITLE> et </TITLE>
- la **description** du contenu de la page dans la balise <META NAME="DESCRIPTION" CONTENT="mettez à cet endroit le sujet de votre page">
- les **mots-clés** ("keywords") dans la balise <META NAME="KEYWORDS" CONTENT="mettez à cet endroit les mots qui définissent de quoi traite votre page">

Le "corps" de votre page web, autrement dit ce qui sera visible dans les navigateurs et donc lu par les internautes, est compris entre les balises <BODY> et </BODY>. C'est entre ces deux balises que vous allez inclure du texte, des images, des liens ("links"), des tableaux ("tables"), des cadres ("frames")...

Exemple

La page d'accueil de votre site ressemblera à ça :

```
<HTML>
<HEAD>
<TITLE>nom de mon site</TITLE>
<META NAME="DESCRIPTION" CONTENT="ce que ma page contient">
<META NAME="KEYWORDS" CONTENT="mots qui définissent de quoi traite ma page">
</HEAD>
<BODY>
Ceci est la page d'accueil de mon site. Vous y trouverez des informations sur...
</BODY>
</HTML>
```

Remarque : Seul le texte en noir sera visible par les internautes lorsqu'ils se connecteront sur votre page. Remarquez que nous avons défini la "tête" en bleu et les balises du "corps" en rouge, pour faire ressortir la structure d'une page HTML type.

4.3 Créer des caractères spéciaux

Le HTML étant un langage mis au point par des américains, un certain nombre de subtilités de la langue française lui échappent et demandent un traitement spécial : en clair, les caractères accentués, les cédilles ou les trémas ne peuvent être tapés directement sur le clavier.

En langage HTML, les accents sont donc traités par des **entités** commençant par & et se terminant par un point virgule. Vous trouverez ci-dessous les caractères spéciaux les plus courants :

- é = é ; (acute = aigu)
- è = è ;

- à = à
- ê = ê (circumflex=circonflexe)
- ÿ = ï (humlaut en allemand = tréma)
- ç = ç (cedilla=cédille)

Bon à savoir : si l'on veut introduire un espace insécable pour ne pas séparer deux mots ou deux chiffres (par exemple, 250 000), il faudra utiliser le caractère ; (non breaking space).

Exemple

le texte Voilà l'été et ses 200 000 kilomètres de bouchons. **s'écrit :**

```
<html>
<body>
Voil&agrave; ; l'&eacute; ;t&eacute; ; et ses 200&nbsp;;000 kilom&egrave; ;tres de bouchons.
</body>
</html>
```

4.4 Le style du texte

Afin d'améliorer la lisibilité de vos textes ou simplement d'accroître la richesse de votre mise en page, vous pouvez donner un style particulier aux mots. Voici les balises les plus courantes :

- **gras** : mot(s) compris entre la balise ouvrante et la balise fermante (bold).
- *italique* : <i> et </i> (italic).
- souligné : <u> et </u> (underline).
- indice : _{et} (subscript)
- exposant : ^{et} (superscript)

Exemple :

Le texte Ceci est une *introduction* au **html**. Vous y trouverez les principales balises utilisées pour créer des pages web. **s'écrit :**

```
<html>
<body>
Ceci est une <b><i>introduction</i></b> au <b>html</b>. Vous y trouverez les principales <u>balises</u>
utilis&eacute; ;es pour cr&eacute; ;er des pages web.
</body>
</html>
```

4.5 La mise en forme du texte

Vos textes gagneront naturellement en lisibilité si vous faites un effort de mise en page. Voici les principales balises qui vous aideront à mettre en forme votre texte :

- **aller à la ligne** :
 (break)
- **créer un paragraphe** : <p> et </p>
- **créer un filet**/une ligne horizontale : <hr>
- **niveaux de titre** de 1 (le plus grand) à 6 (le plus petit) : <h1> et </h1>, <h2> et </h2>...
- **liste non ordonnée** (c'est à dire des puces au début de chaque ligne) : et (unordered list) avec à l'intérieur de ces balises les et pour chaque élément de la liste.
- **liste ordonnée** (1, 2, 3...) : et (ordered list) avec à l'intérieur de ces balises les et pour chaque élément de la liste.

Exemple

L'Internaute Caractéristiques du site

- Attrayant
- Sympathique
- Complet

s'écrit :

```
<html>
<body>
<h1>L'Internaute</h1>
<br>
caract&eacute; ;ristiques du site
<br><br>
<ul>
```

```

<li>attrayant</li>
<li>sympathique</li>
<li>complet</li>
</ul>
</body>
</html>

```

Les attributs (paramètres) qui viennent préciser la balise **<BODY>** (concerne l'ensemble de la page) :

- couleur de fond de la page : **BGCOLOR** (BackGround),
- couleur du texte : **TEXT**,
- couleur des liens non visités : **LINK** (lien),
- couleur des liens activés : **ALINK**,
- couleur des liens visités : **VLINK**.

L'attribut (paramètre) qui détermine le positionnement des paragraphes dans la page (calé à gauche ou à droite, centré) :

- alignement : **ALIGN** + choix entre **LEFT** (gauche), **CENTER** (centre) et **RIGHT** (droit).

Les attributs (paramètres) qui viennent préciser les paramètres du texte utilisent la balise **** :

- couleur : **COLOR**,
- taille : **SIZE**

- police (type de caractère) : **FACE**.

Exemple

(pour les libellés des couleurs et savoir insérer des liens, voir les pages suivantes)

```

<html>
<body bgcolor="#000000" text="#ff0000" alink="#ffffff" vlink="#ffff00">
<font color="#ffffff" size="3" face="arial">
<ul>
<li>le html, qu'est-ce que c'est?</li>
<li>apprendre le html à quoi &ccedil;a sert?</li>
<li>le principe des balises</li>
</ul>
vous &ecirc;tes sur le site de <a href="http ://www.linternaute.com" target="blank">L'Internaute</a>
<br><br>
Pour envoyer vos remarques,
<a href="mailto :guernalec@benchmark.fr">clickez ici</a>
</font>
</body>
</html> Voir l'exemple dans un navigateur.

```

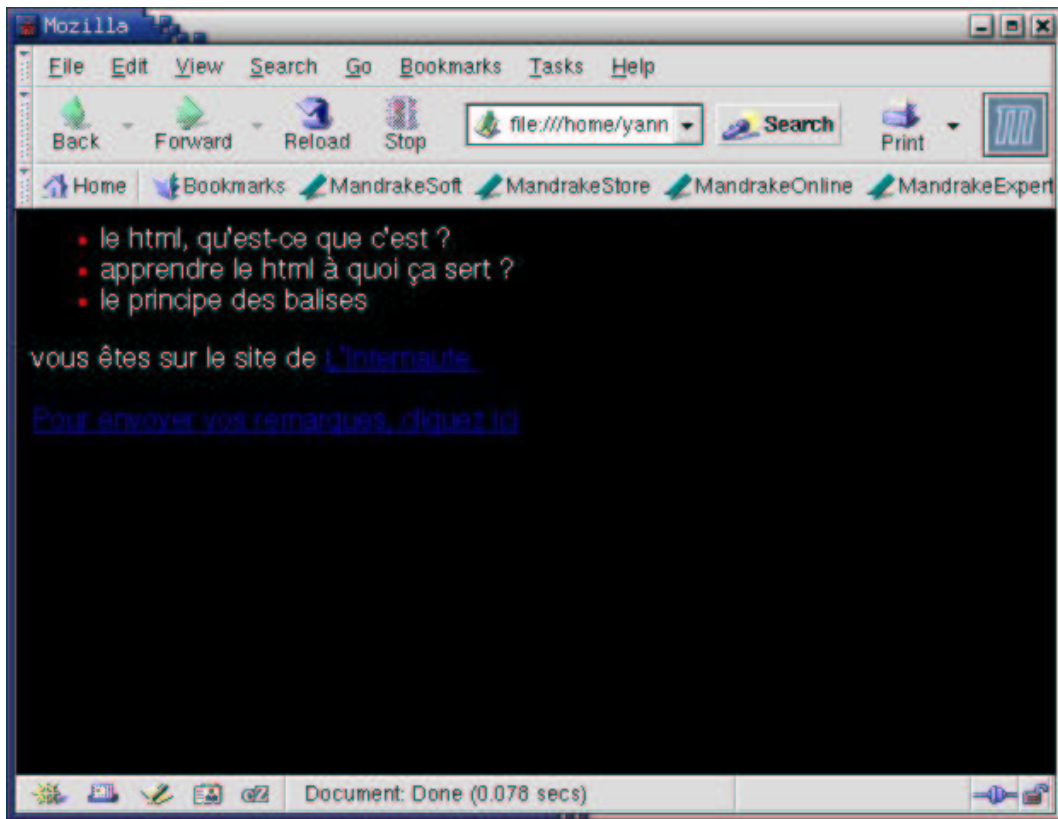
4.6 Mettre des couleurs

Si vous utilisez les noms de couleurs usuels comme "white" (blanc), "black" (noir), "red" (rouge), "blue" (bleu), cette solution est peu fiable et vous réservera des surprises plutôt désagréables. Pour plus de rigueur, il convient de se conformer au codage des couleurs en RGB (Red Green Blue). Par exemple, la couleur de fond de votre page s'écrit comme ceci :

```
bgcolor="#rrggbb"
```

Chaque composante de la couleur que vous désirez est définie par une valeur numérique comprise entre 0 et 255, et exprimée en hexadécimales, c'est à dire à l'aide des chiffres 0 - 9 et des lettres a - f : 00 correspond à l'absence de couleur et ff à la luminosité maximale.

	Red	Green	Blue
Red	ff	00	00
Green	00	ff	00
Bblue	00	00	ff
Black	00	00	00
White	ff	ff	ff

**Exemple**

L'Internaute est en couleurs

S'écrit :

```
<html>
<body>
<font color="#0066FF">
L'internaute
</font>
<font color="#66FF33">
est en
</font>
<font color="#FF66FF">
couleurs
</font>
</body>
</html>
```

La page <http://www.lynda.com/hexh.html> contient un tableau assez complet des couleurs disponibles.

4.7 Insérer un lien hypertexte

Les liens, traditionnellement en bleu et soulignés, permettent de naviguer dans et à l'extérieur de votre site et d'envoyer directement un mail à une personne avec sa messagerie. Pour les créer en langage HTML, utilisez la balise : `<a>` et `` pour anchor (ancree) avec l'attribut href (hypertext reference).

L'attribut "target" (cible en français) d'un lien hypertexte permet de préciser où la page appelée va afficher. Parmi les différentes possibilités, nous avons retenu les deux plus courantes :

1. la page s'affiche en ouvrant une nouvelle fenêtre du navigateur, vous écrirez : `target="blank"` (vierge),
2. la nouvelle page reste dans la même page, vous écrirez : `target="self"` (même).

Exemples :

- L'Internaute s'écrit :

```
<a href="http://www.linternaute.com" target="blank">
```

L'Internaute

```
</a>
```

- Envoyez un mail à Florence Guernalec s'écrit :

Envoyez un mail à

```
<a href="mailto:guernalec@benchmark.fr">Florence Guernalec</a>
```

(pour "mail to", "adresser par mail à")

4.8 Insérer une image

Utilisez la balise `` pour "IMaGe" avec l'attribut `src` (source de l'image), soit son adresse ou URL (Uniform Resource Locator). Dans le cas des sites personnels les plus simples, les images se trouvent dans le même dossier que vos pages HTML, l'appel d'une image s'écrira alors de la façon suivante :

```

```

Les attributs d'une image :

- largeur : `WIDTH`,
- hauteur : `HEIGHT`,
- espace horizontal qui sépare l'image du texte : `HSPACE`,
- espace vertical qui sépare l'image du texte : `VSPACE`,
- affichage alternatif (texte qui s'affiche sur votre navigateur avant que l'image ne se télécharge) : `ALT`.

Exemple :

L'image ci-dessous s'appelle "monimage.jpg"



S'écrit :

```
<html>
```

```
<body>
```

```

```

```
</body>
```

```
</html>
```

N.B : notez que sur le web, les images sont en général au format GIF (.gif) ou au format JPEG (.jpg). Mais on préférera le format PNG libre (Portable Network Graphic) au format Gif qui est propriétaire.

4.9 Créer un tableau

Un tableau est compris entre les balises : `<table>` et `</table>`

- Chaque ligne (ou rang) du tableau est comprise entre la balise <tr> et </tr> ("table row"),
- Chaque cellule est comprise entre la balise <td> et </td> ("table data") ou <th> et </th> (table header) si on désire mettre en forme le texte contenu dans la cellule..

Les attributs d'un tableau :

- bordure du tableau : BORDER (exprimée en pixels),
- largeur : WIDTH,
- hauteur : HEIGHT,
- couleur de fond : BGCOLOR.

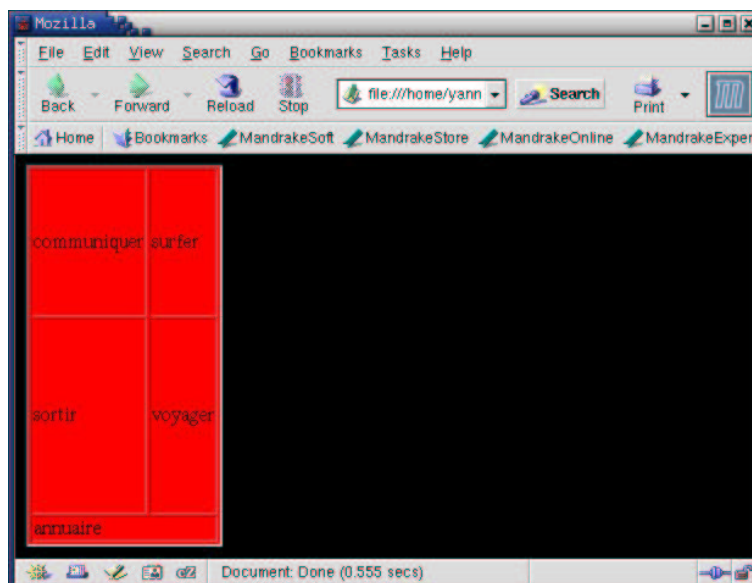
Les attributs des cellules :

- largeur : WIDTH,
- hauteur : HEIGHT,
- couleur de fond : BGCOLOR,
- alignement horizontal : ALIGN,
- alignement vertical : VALIGN,
- marge séparant le bord des cellules de leur contenu : CELSPADDING,
- nombre de pixels séparant les cellules : CELSPACING,
- étendre une cellule sur plusieurs lignes : ROWSPAN,
- étendre une cellule sur plusieurs colonnes : COLSPAN.

Exemple

```
<html>
<body bgcolor="#000000">
<table bgcolor=" #ff0000" border="2">
<tr width="120" height="120">
<td>communiquer</td>
<td>surfer</td>
</tr>
<tr width="120" height="160">
<td cellpadding="15" cellspacing=10>sortir</td>
<td cellpadding="15" cellspacing=10>voyager</td>
</tr>
<tr>
<td colspan="2">annuaire</td>
</tr>
</table>
</body>
</html>
```

Voici le résultat :



4.10 Créer des formulaires

L'utilisation des formulaires est une clé de l'interactivité grandissante entre le webmaster et les visiteurs.

Je vois deux avantages aux formulaires :

- Côté webmaster : les données ainsi recueillies (par le biais de zones (ou champs) de saisie, de listes déroulantes, de boutons de commande, de zones de texte à plusieurs lignes) sont toutes de même type et donc peuvent être analysées statistiquement le cas échéant (attention à la législation sur la collecte de telles données).
- Côté visiteur : en quelques clics un formulaire bien construit est rempli et retourné au webmaster ce qui est beaucoup plus facile et incitatif que de cliquer sur un lien e-mail pour lui envoyer un message de feedback.

4.10.1 Structure de base

`<FORM>` `</FORM>` indique au navigateur le début et la fin d'un formulaire. Deux éléments essentiels sont ensuite précisés à l'intérieur du tag d'ouverture :

1. L'adresse de destination du formulaire ("`http://www.truc.com/`" ou "`mailto:machin@bidule.net`") : `<FORM ACTION="URL">`. Le but de ce cours étant une initiation, on retiendra la seule possibilité de recevoir par mail le contenu du formulaire rempli.
2. La méthode de transmission du formulaire : `<FORM METHOD="METHODE">` où METHODE est soit GET soit POST. Je conseillerais dans le cadre d'une récupération par mail, de s'en tenir à `post`.

4.10.2 Attributs de la balise <FORM>

Outre les deux attributs mentionnés ci-dessus, on peut aussi choisir de préciser :

- Le type d'encodage du formulaire : `<FORM ENCTYPE="TYPE">`. On choisira de fixer le type d'encodage à `text/plain` pour ce qui nous concerne.
- Le nom du formulaire : `<FORM NAME="Nom">` (sert ensuite le cas échéant dans des scripts).

Voici un exemple de structure externe de formulaire :

```
<FORM ACTION="mailto:toto@truc.fr?subject=Réponse au cours 8" METHOD="post" Name="feedback">
</FORM>
```

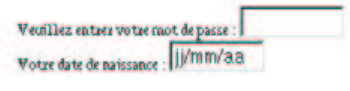
A présent, pour qu'un utilisateur puisse entrer des informations structurées dans le formulaire, il va falloir en définir le contenu...

4.10.3 Zones de saisie à une ligne

Elles sont créées par la balise `<INPUT>` qu'on ne ferme pas, tout comme c'était le cas pour la balise d'insertion d'image. A l'intérieur de cette balise, un certain nombre d'éléments sont à préciser :

Attribut	Fonction
<code><NAME="nom"></code>	Chaque zone de saisie doit avoir un nom unique (et de préférence significatif) dans le formulaire. Ce nom doit bien être mis entre guillemets et ne doit contenir ni accents ni espace. Il sert lors de la récupération du contenu du formulaire par mail.
<code><SIZE="valeur"></code>	Détermine la longueur (nombre de caractères) de la zone affichée à l'écran
<code><MAXLENGTH="valeur"></code>	Détermine la longueur interne de la zone, soit le nombre de caractères qu'elle peut réellement contenir (illimité par défaut). Le cas échéant, une barre de défilement s'ajoutera si la longueur interne est supérieure à la longueur affichée.
<code><TYPE="text int float date url password"></code>	En principe, on peut tout écrire dans une zone input. Mais on peut aussi choisir de spécifier le type de saisie autorisée : texte, nombre entier, nombres décimaux, date, adresse internet, mot de passe (caractères non visibles et remplacés par des astérisques). Nous reviendrons plus loin sur des cas particuliers de valeur pour TYPE comme radio, checkbox, button, submit, reset et file.
<code><VALUE="valeur"></code>	Sert à afficher une valeur par défaut de la zone de saisie.
<code><MIN="valeur"></code>	Fixe la valeur minimale autorisée dans une zone destinée à la saisie numérique.
<code><MAX="valeur"></code>	Fixe la valeur maximale autorisée dans une zone destinée à la saisie numérique.

Voici un exemple d'application :

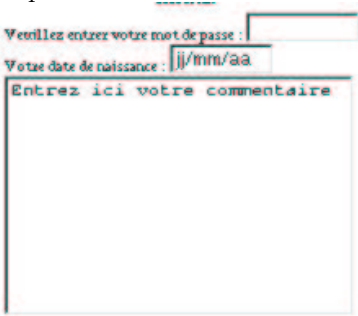
Code source	Résultat
<pre><FORM ACTION="mailto :phoebz@ifrance.com ?subject= Essai formulaire leçon 8" METHOD="POST" EN- TYPE="text/plain"> <P align="left"> Veuillez entrer votre mot de passe : <IN- PUT TYPE="PASSWORD" NAME="pass" SIZE="9">
 Votre date de naissance : <INPUT TYPE="DATE" NAME="Naissance" SIZE="8" MAXLENGHT="8" VA- LUE="jj/mm/aa">
 </FORM></pre>	<p>Top of Form 1</p>  <p>Bottom of Form 1</p>

On voit rapidement que cet exemple n'est pas fonctionnel car rien ne permet de déclencher l'envoi du formulaire rempli à l'adresse mail indiquée, ce que nous verrons plus loin.

Zones de saisie multilignes

Ces zones servent par exemple à entrer des commentaires et autres remarques d'une certaine longueur. Elles sont créées par la balise `<TEXTAREA>` `</TEXTAREA>`. Cette zone de saisie doit elle aussi être nommée, ce qui se fait de la même façon qu'une zone monoligne, par `<TEXTAREA NAME="nom">`. On doit également en définir la taille, c'est ce à quoi servent `<TEXTAREA ROWS="valeur" COLS="valeur">`.

Voici un exemple d'application :

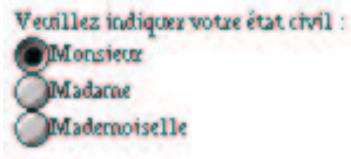
Code source	Résultat
<pre><FORM ACTION="mailto :phoebz@ifrance.com ? subject= Essai formulaire leçon 8" ME- THOD="POST" ENTYPE="text/plain"> <P align="left"> Veuillez entrer votre mot de passe : <INPUT TYPE="PASSWORD" NAME="pass" SIZE="9">
 Votre date de naissance : <INPUT TYPE="DATE" NAME="Naissance" SIZE="8" MAXLENGHT="8" VALUE="jj/mm/aa">
 <TEXTAREA NAME="Comment" COLS="30" ROWS="10"> Entrez ici votre commen- taire</TEXTAREA>
 </FORM></pre>	<p>Top of Form 2</p>  <p>Entrez ici votre commentaire</p> <p>Bottom of Form 2</p>

4.10.4 Choisir, cocher, sélectionner

On propose un certain nombre de petit "boutons", chacun ayant un nom, parmi lesquels on ne peut en choisir qu'un. C'est ici qu'on utilise `<INPUT TYPE="radio">`.

On va ensuite utiliser `<INPUT TYPE="radio" NAME="Nom">` pour nommer le groupe de zone d'options et créer autant de zones de même nom que d'options (ça paraît complexe comme ça, mais un p'tit exemple et hop ça sera limpide). Tous les boutons d'un même groupe ont le même nom, il va donc falloir définir un nom interne pour différencier les différents boutons (je rappelle qu'on ne peut en choisir qu'un) ce qui se fait avec `<INPUT TYPE="radio" NAME="Nom" VALUE="valeur">`.

Concrètement, voici un petit exemple :

Code source	Résultat
<pre><FORM ACTION="mailto :phoebz@ifrance.com? subject= Essai formulaire leçon 8" METHOD="POST" EN- TYPE="text/plain"> <P align="left"> Veuillez indiquer votre état civil :
 <INPUT TYPE="Radio" NAME="EC" Value="Monsieur" CHECKED> Monsieur
 <INPUT TYPE="Radio" NAME="EC" Value="Madame"> Madame
 <INPUT TYPE="Radio" NAME="EC" Value="Mademoiselle">Mademoiselle </FORM></pre>	<p>Top of Form 3</p>  <p>Bottom of Form 3</p>

Tous les boutons correspondant au choix de l'état-civil sont nommé EC mais ont chacun un nom interne différent : lors du retour du formulaire le mail contiendra EC = Valeur interne du bouton sélectionné. L'attribut CHECKED permet de présélectionner un bouton (à utiliser bien évidemment pour un seul d'entre eux!). Le nom de la zone et le nom interne ne doivent comporter ni espace, ni accent, ce qui évidemment n'est pas le cas pour la "légende" (texte visible à l'écran).

4.10.4.1 Cases à cocher

Elles se présentent comme les boutons d'options comme un groupe de boutons avec une légende, mais avec cette différence que le visiteur peut en cocher plusieurs. Le fonctionnement est semblable à celui vu pour les boutons d'options avec cette fois la syntaxe suivante : `<INPUT TYPE="CHECKBOX" NAME="Nom de la zone" VALUE="Nom interne">`.

4.10.4.2 Listes de choix

Elles permettent elles aussi à l'utilisateur de choisir une option parmi d'autres. On les crée à l'aide de la balise `<SELECT NAME="Nom de la liste" SIZE="5" MULTIPLE> </SELECT>` : notez bien qu'il n'y a toujours ni espace ni accent dans le nom, que SIZE indique le nombre d'éléments qui seront visibles sans le recours à une barre de navigation et que MULTIPLE autorise expressément les choix multiples (utilisation de la touche MAJ et de la touche CTRL). Il convient ensuite de spécifier les différents éléments de cette liste : `<OPTION>Elément de la liste</OPTION>`.

N.B. : SELECTED permet de présélectionner un élément.

4.10.4.3 Les boutons de commande

Une fois le formulaire créé, il faut penser à l'envoyer ; donner au visiteur la possibilité de tout reprendre à zéro ("ouin, j'm'ai trompé!") peut aussi être une bonne idée...

La syntaxe pour le bouton d'envoi de commande d'envoi de formulaire est la suivante : `<INPUT TYPE="SUBMIT" VALUE="Nom">`, le nom étant le texte qui figurera sur le bouton.

La syntaxe pour le bouton de "remise à zéro" du formulaire est la suivante : `<INPUT TYPE="RESET" VALUE="Nom">`, le nom étant le texte qui figurera sur le bouton.

Allez, un petit formulaire pour la route (enfin, juste pour avoir de quoi bien regarder le code source qui suit!) ? Et voici le code HTML correspondant :

```
<form action="mailto:smz@libertysurf.fr" enctype="text/plain" method="post">
<table width="100%" bgcolor="#ffffff">
<TR><TD bgcolor="#99cccc" align="center" colspan="2"><font size="2">Merci de
bien vouloir remplir ce formulaire</font></td></tr>
<TR><TD colspan="2" align="left"><font size="2">
<FIELDSET><LEGEND ALIGN="top">Informations personnelles</LEGEND><br>
Nom ou Nick : <Input name="Nom" type="TEXT" size="20" maxlength="20"><BR>
Ville - Pays : <Input name="Lieu" type="TEXT" size="60" maxlength="60"
value="Grenoble - France"><BR>
Sexe : <Input type="radio" name="sexe" value="masculin" checked> Masculin -
<Input type="radio" name="sexe" value="feminin"> F&eacute;minin - <Input
```

```

type="radio" name="sexe" value="indetermine"> Indéterminé;<BR>
Age : <SELECT
NAME="age"><option>0-15</option><option>15-25</option><option>25-35</option><opt
ion>35 et plus</option></select><br><br>
</FIELDSET>
</font></TD></TR>
<TR>
<TD bgcolor="#ffffcc" width="40%" height="100"><font size="1"><FIELDSET><LEGEND
align="top">Votre avis m'intéresse</LEGEND><BR>
Vous pensez que cette leçon :<BR>
<input type="checkbox" name="lesson" value="clair" checked> Est très claire <BR>
<input type="checkbox" name="lesson" value="ok"> Est utile <BR>
<input type="checkbox" name="lesson" value="revision"> Doit être complétée et améliorée<br><br>
<TABLE><TR><TD align="left" valign="middle"><font size="1">Le graphisme en est
:</font></td><TD align="center"><font size="1"><SELECT name="graphisme"
size="3"><option>Sympa</option><option>Moyen</option><option>À
revoir</option></select></font></td></tr></table>
</FIELDSET></font></td>
<TD bgcolor="#3399cc" valign="top"><font size="1"><FIELDSET><LEGEND
align="top">Des suggestions ?</LEGEND><TEXTAREA name="commentaire" cols="50"
rows="9">Tapez ici toute remarque que vous jugerez
utile</textarea></FIELDSET></font></td>
</tr>
<TR><TD colspan="2" align="center" bgcolor="#ffcc99"><input type="submit"
value="Envoyer">&nbsp;&nbsp;&nbsp;<input type="reset" value="Effacer"></td></tr>
</TABLE>
</form></p><BR><BR>

```

4.11 Créer des cadres (ou "frame")

Créer des "frame", ces cadres indépendants ayant chacun leur propre barre de défilement (ou non), réunis sur une même page, revient à construire une page web à partir de plusieurs pages. Sur cette page principale comprise entre la balise <FRAMESET> et </FRAMESET>, on appelle les pages comme on appelle une image avec la balise <FRAME SRC> en définissant leur taille (COL) et leur emplacement.

Les attributs des cadres :

- bordure : FRAMEBORDER (1=yes, 0=no),
- marge horizontale à l'intérieur du cadre : MARGINWIDTH,
- marge verticale à l'intérieur du cadre : MARGINHEIGHT,
- taille des cadres inchangé : NORESIZE,

– présence d'un ascenseur (ou barre de défilement) : `SCROLLING` (auto, yes, no).

Exemple

Supposons que vous avez deux pages ("`page1.html`" et "`page2.html`") et que vous vouliez les intégrer dans une seule page web avec la "`page1.html`" à gauche de l'écran et la "`page2.html`" à droite, ayant respectivement pour taille 250 et 500, et la première ayant un ascenseur, le code HTML s'écrira :

```
<html>
<body>
<frameset col= "250, 500">
<frame src= "page1.html" name="gauche" frameborder="no" noresize scrolling="yes" marginwidth="0"
marginheight="0">
<frame src="page2.html" name="droite" frameborder="no" noresize scrolling="no" marginwidth="0"
marginheight="0">
</frameset>
</body>
</html>
```

Voilà. Vous n'en savez certainement pas assez pour construire le plus beau site du monde mais suffisamment pour écrire à la main une page simple ou "entrer dans le code" et modifier des paramètres écrits par votre éditeur.

Chapitre 5

Petit cours de PHP et MySQL

Ce chapitre est tiré et adapté du site web <http://www.phpdebutant.com>.

5.1 Afficher une phrase ou une image

Pour créer un fichier PHP, ouvrez votre éditeur de texte favoris. Le code PHP doit être insérer entre les balises `<? et ?>`.

En réalité voici les balises possibles : `<? ?>` ou `<?php ?>` ou `<% %>`, un conseil prenez l'habitude de toujours utiliser les mêmes, les premières sont les plus utilisées.

La première chose à savoir c'est qu'une syntaxe se termine **TOUJOURS** par un point-virgule voir ci-dessous, si vous l'oubliez vous verrez apparaître une `PARSE ERROR` lors de l'exécution de votre fichier.

Code PHP	Donne comme résultat à l'écran
<code><?></code>	
<code>print ("Bonjour le monde! ");</code>	Bonjour le monde!
<code>?></code>	

C'est la fonction `Print()` que nous utiliserons pour afficher du texte et des images à l'écran. Ici on voit bien que la phrase n'est pas du tout formatée , voici donc comment l'on peut inclure les balises HTML dans PHP (ci-dessous).

Code PHP	Donne comme résultat à l'écran
<code><?></code>	
<code>print(" Bonjour le monde!</code>	Bonjour le monde!
<code>");</code>	
<code>?></code>	

Voilà nous avons ajouté la balise `font` comme en HTML pour formater le texte, la grosse différence **IMPOR- TANTE**, étant l'ajout d'un antislash avant chaque caractère spécial. Exemple : une quote ' devient \ ' et une double-quote " devient \" , il s'agit tout simplement de dire à PHP qu'il ne s'agit pas d'un élément de la syntaxe mais juste d'une apostrophe ou guillemet dans un texte. Affichons maintenant une image en plus du texte.

Code PHP	Donne comme résultat à l'écran
<code><?></code>	
<code>print("<center></code>	Bonjour le monde!
<code>Bonjour le monde!
 ");</code>	
<code>print("</center> ");</code>	
<code>?></code>	

La seule remarque importante dans ce cas est qu'il faut toujours penser à mettre `border="0"` dans votre balise image (`<img...>`) sans quoi vous aurez une belle bordure autour de l'image .. beurk :). Notez aussi la balise `
` qui sert à passer la ligne du dessous.

Différences entre les navigateurs

À ce stade il est déjà important de faire la distinction en les navigateurs Internet Explorer (IE) et Netscape

Navigator (NS), Opera, Mozilla et Lynx :-)) dans un premier temps je vous conseille d'installer les 2 premiers si vous êtes sous windows et les 3 derniers si vous êtes sous Linux.

Pourquoi tant de haine ? TLà où Netscape respecte la syntaxe exacte du HTML 4, IE lui est plus nonchalant :out simplement parce que tous les navigateurs ne respectent pas scrupuleusement les normes du W3C World Wide Web Consortium). **NB** : Notez que la fonction `Echo()` est semblable à la fonction `Print()`, vous verrez plus loin dans quel contexte je préfère utiliser `Echo()`. Si vous avez eu l'occasion de bosser sous MS-DOS vous serez tentés par `Echo()` alors que `Print()` vient d'UNIX.

Pour résumer : voici les 2 méthodes conseillées pour afficher du texte, des images (n'oubliez pas que ce sont les double quotes qui délimitent la chaîne de caractères :

```
- print (" le texte ");
- echo " le texte " ;
```

5.2 Afficher la date et l'heure

Avec PHP, il est fort simple d'afficher la date du jour mais aussi de savoir quel jour nous serons dans 432 jours et réciproquement dans le passé. Voyons tout d'abord une date simple, nous allons en profiter pour utiliser notre première variable (les variables commencent toujours par le signe dollar \$).

Code PHP

(ne copier/coller pas ce code dans votre éditeur, Ce qui donne à l'écran retapez-le ou gare aux erreurs...)

```
<?
$date = date("d-m-Y");
$heure = date("H :i");
Print("Nous sommes le $date et il est $heure");
?>
```

Nous sommes le 14-09-2000 et il est 15 :10

C'est donc la fonction `date()` qui permet d'obtenir l'heure locale du serveur, mais attention l'heure locale est fonction de la situation géographique du serveur en lui-même. En effet un serveur situé au canada vous donnera l'heure du canada, en ce qui nous concerne les serveurs de Free.fr sont en France donc l'heure locale sera l'heure Française :).

Dans le code ci-dessus nous générons la variable `$date` en lui donnant la valeur de ce que retourne la fonction `date("d-m-Y")` en l'occurrence : **14-09-2000**. Les paramètres contenus entre les parenthèses `d-m-Y` peuvent être placés dans l'ordre que vous désirez, ainsi la date au format US sera écrite ainsi : `date("Y-m-d")`, il existe beaucoup de paramètres (extrait de la doc. en Français de **Nexen.net**), je vous conseille de les tester pour vous rendre compte de ce que chaque paramètre retourne comme résultat :

- **a** - "am" (matin) ou "pm" (après-midi)
- **A** - "AM" (matin) ou "PM" (après-midi)
- **d** - Jour du mois, sur deux chiffres (éventuellement avec un zéros) : "01" à "31"
- **D** - Jour de la semaine, en trois lettres (et en anglais) : par exemple "Fri" (pour Vendredi)
- **F** - Mois, textuel, version longue ; en anglais, i.e. "January" (pour Janvier)
- **h** - Heure, au format 12h, "01" à "12"
- **H** - heure, au format 24h, "00" à "23"
- **g** - Heure, au format 12h sans les zéros initiaux, "1" à "12"
- **G** - Heure, au format 24h sans les zéros initiaux, "0" à "23"
- **i** - Minutes ; "00" à "59"
- **j** - Jour du mois sans les zéros initiaux : "1" à "31"
- **l** - ('L' minuscule) - Jour de la semaine, textuel, version longue ; en anglais, i.e. "Friday" (pour Vendredi)
- **L** - Booléen pour savoir si l'année est bissextile ("1") ou pas ("0")
- **m** - - Mois ; i.e. "01" à "12"
- **n** - Mois sans les zéros initiaux ; i.e. "1" à "12"
- **M** - Mois, en trois lettres (et en anglais) : par exemple "Jan" (pour Janvier)
- **s** - Secondes ; i.e. "00" à "59"
- **S** - Suffixe ordinal d'un nom

5.3 PHP dans du code HTML

Attention : A partir du moment où vous placez du code PHP dans un fichier `*.htm` ou `*.html`, vous devrez renommer ce fichier en `*.php` ou `*.php3` ou encore `*.phtml`, bien que le plus utilisé soit `*.php3`. Si vous ne faites

pas cette manipulation, le code apparaîtra en toutes lettres dans le navigateur sans être exécuté par le serveur (n'ayant pas reconnu l'extension associée à php).

Comme je vous le disais en introduction, l'un des avantages du PHP c'est qu'il s'intègre facilement dans du code HTML classique. C'est d'ailleurs pour cela (en partie) qu'il connaît un fort succès sur les homepages persos. En effet chacun peut à sa guise inclure quelques parties en PHP sans avoir à casser le site entièrement.

Le code PHP/HTML

(ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)

```
<html>
<body>
<font size="2" face="Arial">Le texte en
HTML</font>
<?
// le code PHP ———
$heure = date("H\hi");
print("<font size=\"2\" face=\"Arial\"> et celui en PHP.</font>");
?>
Le texte en HTML et celui en PHP.
Il est 18h16.
<!-- retour au code HTML -->
<br><font size="2" face="Arial">Il est <? echo
$heure; ?>.</font>
</body>
</html>
```

L'exemple ci-dessus démontre bien cette facilité à mélanger les deux langages. Notez que la seconde fois j'ai utilisé la fonction `echo` et non pas `print()`, c'est le plus souvent dans ce cas que je l'utilise pour une lecture plus simple du code dans les autres cas ce sera `print()`, mais bien sûr vous êtes libre sur ce coup là :).

A noter : En PHP si vous souhaitez ajouter des commentaires il suffit de faire suivre deux Slashes // puis le commentaire de votre choix. Je mets l'accent sur les commentaires surtout lorsque que vous aurez des dizaines de lignes de code, ils vous seront utiles pour vous y retrouver 6 mois plus tard, donc n'hésitez pas à en mettre, même sur des choses qui vous paraissent logiques sur l'instant. Si vous souhaitez mettre plusieurs lignes en commentaire, vous pouvez également utiliser le "slash étoile" puis "étoile slash" à la fin comme ceci : /* le commentaire */.

Quelques mots au passage sur la fonction `include()` : Si le code de votre page HTML est long, il est parfois fouillis de rajouter des lignes de codes PHP en plus, la méthode que j'utilise beaucoup pour bien séparer le code HTML du code PHP est la fonction `include()`.

Le code HTML/PHP

(ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)

```
<html>
<body>
<font size="2" face="Arial">Le texte en
HTML</font>
<?
include("toto.inc.php3"); // on appelle le fichier
?>
Le texte en HTML et celui en PHP. Il est 18h16.
</body>
</html>
```

Le code PHP de toto.inc.php3

```
<?
$heure = date("H\hi");
print("<center><font size=\"2\" face=\"Arial\"> et celui en PHP. Il est
$heure.</font></center>");
?>
```

Et voilà le tour est joué, le code PHP est maintenant dans un fichier bien séparé mais est exécuté à l'appel du fichier principal. Vous aurez noté que les fichiers qui sont inclus portent l'extension `*.inc.php3`, ceci pour une meilleure lisibilité. Ainsi, en effet vous savez tout de suite si le fichier est exécuté directement ou bien s'il est

uniquement appelé dans un ou plusieurs autres fichiers.

5.4 La concaténation

Le [point](#) est utilisé pour concaténer des chaînes, variables etc. Prenons l'exemple d'une phrase ou un texte doit être collé au bout d'une variable, (voyez ci-dessous), pour que php sache que le nom de la variable s'arrête à un endroit précis, nous utiliserons le [point](#).

Le code PHP	Donne comme résultat à l'écran
<pre><? \$date = gmdate("H\hi"); print("Il est \$date"."gmt."); ?></pre>	Il est 19h05gmt.

Vous le voyez pour éviter que PHP pense que la variable porte le nom `$dategmt`, il faut refermer la double quote, mettre un point puis la réouvrir pour mettre le restant du texte (gmt). Notez également que le second point est lui placé entre les double quotes, donc, sera interprété comme du texte simple et non pas comme une demande de concaténation.

Nous allons maintenant voir la différence entre du texte entre ' ' dites simples quotes et du textes entre double quotes " "

Le code PHP	Donne comme résultat à l'écran
<pre><? \$nom = "Martin"; echo "Mon nom est \$nom"; ?></pre>	Mon nom est Martin
<pre><? \$nom = "Martin"; // affichage avec des simple quote echo 'Mon nom est \$nom'; ?></pre>	Mon nom est \$nom
<pre><? \$nom = "Martin"; // affichage avec des simple quote echo 'Mon nom est '.\$nom; ?></pre>	Mon nom est Martin

Vous l'aurez compris, PHP n'interprète pas ce qui se trouve entre simple quote, ainsi, ce n'est pas la valeur de `$nom` qui est affiché, mais `$nom`. Il faut donc utiliser l'opérateur de concatenation (le `.`) pour avoir l'affichage voulu.

Pensez aussi que si vous voulez afficher un ' dans un texte entre deux ' ', alors il faudra faire :

```
echo 'Aujourd\'hui';
```

Ainsi le `\` (backslash) indique à php qu'il ne faut pas considérer le ' du milieu comme celui qui délimite la fin de la chaîne de caractère, mais juste un caractère comme un autre. Il en va de même pour afficher un " entre deux ". Ci-dessous vous allez voir qu'il est possible de concaténer directement une fonction et une chaîne de caractères.

Le code PHP	Donne comme résultat à l'écran
<pre><? print('Nous somme le ' . gmdate('d-m-Y') . '...'); ?></pre>	Nous somme le 15-09-2000...

Nous avons réduit le code d'une ligne, ce qui n'est pas négligeable pour les gros développements, par contre j'admet que ceci est moins lisible pour quelqu'un qui débute totalement, c'est à vous de choisir.

Dans certains exercices futurs, nous verrons comment appeler une page en passant quelques variables, dans ce cas la concaténation nous servira je propose donc de regarder le tableau ci-dessous :

Ce qu'il faut éviter de faire :

```
<?
$fichier = "fichier.php3 ?var=$var&data=$data";
?>
```

Ce qu'il est conseillé de faire :

```
<?
$fichier = 'fichier.php3 ?var='.$var.'&data='.$data;
?>
```

En d'autres termes, chaque fois que vous collez du texte et une variable (ou fonction), n'oubliez pas de mettre le **point**. Je ne dis pas que la première méthode ne fonctionne pas, mais elle n'est pas orthodoxe, et autant prendre les bonnes habitudes tout de suite :).

5.5 Récupérer les valeurs d'un formulaire

Quand l'un de vos visiteurs entre les informations dans un formulaire, celle-ci sont récupérées sous forme de variables et c'est tout simplement le **name** de chaque champ qui devient la variable qui contient ce qu'a entré le visiteur dans le champ, oops :). Allez, un exemple me paraît plus simple, ci-dessous le **name="nom"** devient **\$nom** et **name="prenom"** devient **\$prenom**, il ne reste plus qu'à faire un **print()** des variables et le tour est joué! (**attention n'utilisez jamais d'accents ni d'espaces dans les names**).

Le code HTML du formulaire

(ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)

```
<html><body>
<form method="post" action="verif.php3">
Nom : <input type="text" name="nom"
size="12"><br>
Prénom : <input type="text" name="prenom"
size="12">
<input type="submit" value="OK">
</form></body></html>
```

Le code PHP de verif.php3

(ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)

```
<?
print("<center>Bonjour
$nom</center>");
$prenom
```

Donne comme résultat à l'écran

Top of Form 1

Bottom of Form 1

Donne comme résultat à l'écran après envoi "OK"

Bonjour Thaal Rasha

Il va bien sûr maintenant falloir contrôler les informations que rentre le visiteur pour éviter au maximum les erreurs. La première fonction que nous utiliserons est **empty()**, qui permet de contrôler si un champs est vide. Ensuite nous allons contrôler que **\$url** commence bien par **http ://** à l'aide des deux fonctions **strtolower()** et **substr()**.

Le code HTML du formulaire

(ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)

```
<html><body>
<form method="post" action="verif.php3">
Titre : <input type="text" name="titre"
size="12"><br>
URL : <input type="text" name="url" size="12"
value="http ://">
<input type="submit" value="OK">
</form></body></html>
```

Le code PHP de verif.php3

(ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)

Donne comme résultat à l'écran

Top of Form 2

Bottom of Form 2

Donne comme résultat à l'écran après envoi "OK"

```

<?
if(empty($titre))

print("<center>Le      '<b>Titre</b>'      est
vide!</center>");
exit();

// vérification du début de l'url
$verif_url = strtolower($url);
$verif_url = substr("$verif_url", 0, 7);
// on verifie les 7 premiers caractères
if ($verif_url!="http ://")
{
print("L'URL      doit      commencée      par
<b>http ://</b>");
exit();
}
else
{
print("$titre : <a href=\"\$url\">$url</a>");
}
?>

```

Erreur n°1 : Le 'Titre' est vide!
Erreur n°2 : L'URL doit commencer par http ://
Si pas d'erreur : Titre : URL

Avec cet exemple nous commençons à attaquer les conditions, c'est un aspect primordial dans tous les langages. La première vérification porte sur le champ 'titre', la fonction `empty()` permet de contrôler si celui-ci est vide ou non. Ce qui nous donne :

- `if(empty($titre)) 'erreur';` : Si la variable `$titre` est vide alors j'affiche le message : 'Le titre est vide' (placé entre accolades) et j'arrête l'exécution du reste du code avec la commande `exit()`.
- Par contre si la variable n'est pas vide, l'exécution ne prend pas en compte ce qui se trouve entre accolades et continue.

La seconde vérification est plus fine puisqu'il s'agit de vérifier que les 7 premiers caractères qui ont été entrés par le visiteur sont bien `http ://`. Pour commencer nous utilisons la fonction `strtolower()` qui permet de transformer tous les caractères en minuscules (ex. `HTTP ://www.MONsite.CoM` devient `http ://www.monsite.com`). Puis à l'aide de la fonction `substr()`, nous sélectionnons les 7 premiers caractères (*0 est toujours le premier caractère d'une chaîne - le second chiffre ' 7 ' étant le nombre de caractères à sélectionner*), puis nous les comparons à ce que nous avons dans notre condition `if` :

- `if ($verif_url!="http ://") 'erreur';` : Si les 7 premiers caractères sont différents (signe : `!=`) de `http ://`, alors on exécute ce qui se trouve entre accolades (*en l'occurrence on affiche un message d'erreur*), puis nous arrêtons le reste du code avec la commande `exit()`.
- Par contre si le résultat est correct, PHP ignore ce qui se trouve entre accolades et exécute le reste du code. Vous pourrez faire autant de tests que vous voudrez sur les champs, mais ne soyez pas trop draconien car les visiteurs n'aiment pas trop que l'on empiète sur leur liberté :). Les contrôles les plus fréquents s'effectuent sur les URL et email pour savoir si l'email comporte bien un "@" et un `point`.

Le code HTML du formulaire

(ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)

```

<html><body>
<form method="post" action="verif.php3">
Votre email : <input type="text" name="email"
size="20">
<input type="submit" value="OK">
</form></body></html>

```

Le code PHP de verif.php3

(ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)

Donne comme résultat à l'écran

Top of Form 3



Bottom of Form 3

Donne comme résultat à l'écran après envoi "OK"

```

<?
$point = strpos($email,".");
$aroba = strpos($email,"@");

if($point=="")
{
echo "Votre email doit comporter un
<b>point</b>";
}
elseif($aroba=="")
{
echo "Votre email doit comporter un
<b>'@'</b>";
}
else
{
echo "Votre email est : '<a
href=\"mailto:.$email.\"\"><b>$email</b></a>";
}
?>

```

Erreur n°1 : Votre email doit comporter un **point** !
Erreur n°2 : Votre email doit comporter un '@' ! *Si pas d'erreur* : Votre email est : **email@email.com**

Comme son nom l'indique, la fonction `strpos()` retourne la position d'un caractère dans une chaîne si celui-ci existe, autrement `strpos()` retourne "rien". C'est ce que nous utilisons pour savoir si les `point` et `@` sont bien présents dans l'email.

Exemple : Si `strpos()` retourne "10" cela veut dire que le premier caractère recherché est placé juste après les 10 premiers caractères donc en **11e position** dans la chaîne, puisque vous devez toujours vous rappeler que php commence à compter à **0** et non pas **1**.

5.6 Les structures de contrôles

Pour commencer voici un petit tableau bien utile sur les instructions les plus utilisées

<code>if</code>	Si
<code>else</code>	Autrement
<code>elseif</code>	Autrement Si
<code>switch</code>	selon
<code>while</code>	Chaque fois que (<i>boucle</i>)
<code>for</code>	Tant que (<i>boucle</i>)
<code>==</code>	Strictement égal à
<code>!=</code>	Différent de
<code><</code>	Plus petit que
<code>></code>	Plus grand que
<code><=</code>	Plus petit ou égal
<code>>=</code>	Plus grand ou égal
<code>and</code> ou <code>&&</code>	Et
<code>or</code> ou <code> </code>	Ou

Pour illustrer les `if`, `else` et `elseif`, voici un exemple très simple à lire, nous définissons une variable à la valeur ' 512 ' puis nous allons tester si celle-ci est comprise entre 0 et 499 puis entre 500 et 999 et enfin supérieure à 999, ce qui nous donne :

Le code PHP

Ce qui donne à l'écran

```

<?
$toto = 512;

// on enchaîne les contrôles ci-dessous —
if($toto>=0 && $toto<500) //1er
{
echo $toto.' est compris entre 0 et 499';
}
elseif($toto>=500 && $toto<1000) //2eme
{
echo $toto.' est compris entre 500 et 999';
}
else //3eme
{
echo $toto.' est plus grand que 999';
}
?>

```

- 1er contrôle :** Si (512 est plus grand ou égal à 0 et que 512 est plus petit que 500) on affiche le 1er message ;
- 2e contrôle :** Si (512 est plus grand ou égal à 500 et que 512 est plus petit que 1000) on affiche le 2e message ;
- 3e contrôle :** Dans tous les autres cas on affiche le 3e message ;

Je vous invite à faire plusieurs tests en changeant à chaque fois la valeur de **\$toto** pour vérifier que les messages respectifs s'affichent bien quand les conditions sont remplies.

Voici un autre exemple pour mieux comprendre, et qui nous permettra de voir la structure switch par la suite.

Le code PHP

```

<?
$medor = 'chien';

// on enchaîne les contrôles ci-dessous —
if($medor == 'girafe')
{
echo 'Medor est un girafe!';
}
elseif($medor == 'elephant')
{
echo 'Medor est un éléphant';
}
elseif($medor == 'souris')
{
echo 'Medor est une souris';
}
elseif($medor == 'chien')
{
echo 'Medor est un chien';
}
elseif($medor == 'chat')
{
echo 'Medor est un chat';
}
else
{
echo 'Peut être un hippopotame? Qui sait ...';
}
?>

```

Ce qui donne à l'écran

Medor est un chien

Cependant vous vous apercevez que cette structure est un peu lourde, et pas forcément facile à lire. Utilisez un **switch** permet de résoudre ce problème. Le code suivant est exactement le même que le précédent, mais écrit avec un **switch** :

Le code PHP

```
<?
$medor = 'chien';

switch($medor)
{
case 'girafe' :
echo 'Medor est un girafe!';
break;
case 'elephant' :
echo 'Medor est un éléphant';
break;
case 'souris' :
echo 'Medor est une souris';
break;
case 'chien' :
echo 'Medor est un chien';
break;
case 'chat' :
echo 'Medor est un chat';
break;
default :
echo 'Peut être un hippopotame? Qui sait ...';
}
?>
```

Ce qui donne à l'écran

Medor est un chien

Notez bien l'utilisation de **break** ;. Cela permet de ne pas passer aux case suivant, sans quoi tout les messages s'afficherait.

Passons maintenant à la fameuse boucle **while**, je dis "fameuse" car elle est sujette à de petites galères quand on l'utilise pour la première fois. Nous allons reprendre notre variable **\$toto** à laquelle nous allons donner la valeur **6**, puis à l'aide de la boucle nous allons nous mettre dans le cas où nous ne connaissons pas la valeur de **\$toto** et allons la rechercher. Ce qui donne :

Le code PHP

```
<?
$toto = 6;
$i = 0;

//-----[DEBUT BOUCLE]-----
while($i != $toto)
{
echo 'Toto est différent de '.$i.'<br>';
$i++; // $i++ est équivalent à ($i+1)
}
//-----[FIN BOUCLE]-----

echo 'Toto est égal à '.$i;
?>
```

Ce qui donne à l'écran

Toto est différent de 0
Toto est différent de 1
Toto est différent de 2
Toto est différent de 3
Toto est différent de 4
Toto est différent de 5
Toto est égal à 6

Par convention la variable **\$i** fait toujours office de compteur dans une boucle elle a toujours la valeur "0" au début, vous notez que cette valeur prend **+1** à la fin de la boucle (**\$i++**) et chaque fois que la condition a été respectée on retourne à **WHILE** et l'on fait un nouveau test, etc.,

ce qui donne en français :

1. **\$i = 0** , on teste si **0 est différent de toto = Oui**, on affiche le message (echo) puis on ajoute **1** à **\$i** (**\$i++**) et on retourne à **while**.
2. **\$i = 1** , on teste si **1 est différent de toto = Oui**, on affiche le message (echo) puis on ajoute **1** à **\$i** (**\$i++**) et on retourne à **while**.
3. **\$i = 2** , on teste si **2 est différent de toto = Oui**, on affiche le message (echo) puis on ajoute **1** à **\$i** (**\$i++**) et on retourne à **while**.

4. etc...
5. `$i = 6` , on teste si **6 est différent de toto = Non**, on sort de la boucle (accolades), et on poursuit le code.
En l'occurrence on affiche le message (Toto est égal à 6).

Nous allons maintenant voir les boucles `for`. Elles permettent de réaliser la même chose que les boucles `while`, encore une fois c'est la syntaxe qui change. Au lieu de déclarer le compteur avant le début de la boucle (`$i=0;`) et à chaque fin de tour d'incrémenter le compteur (`$i++`), on le fait directement dans la déclaration de la boucle. Voici le même code que précédemment avec une boucle `for` :

Le code PHP	Ce qui donne à l'écran
<pre><? \$toto = 6; //-----[DEBUT BOUCLE]----- for(\$i=0; \$i!= \$toto; \$i++) { echo 'Toto est différent de '.\$i.'
'; } //-----[FIN BOUCLE]----- echo 'Toto est égal à '.\$i; ?></pre>	<pre>Toto est différent de 0 Toto est différent de 1 Toto est différent de 2 Toto est différent de 3 Toto est différent de 4 Toto est différent de 5 Toto est égal à 6</pre>

Notez : Il est vraiment très important de maîtriser les boucles car c'est là un élément systématiquement utilisé pour afficher des résultats venant d'une base de données et nous l'utiliserons beaucoup.

5.7 Ecrire et lire dans un fichier texte

Nous allons voir ici comment l'on peut écrire et lire depuis un fichier se trouvant sur un serveur FTP, le vôtre en l'occurrence. Pour commencer vous créez un fichier de type `*.txt` (vous pouvez mettre l'extension que vous voulez, voire pas du tout) et nous placerons le fichier dans le même répertoire que le script PHP.

Contenu du fichier <code>data.txt</code>	Donne comme résultat à l'écran
<pre>1523 Le code PHP <code>data.php3</code> (ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...) <? \$fp = fopen("data.txt","r"); // (1) \$donnees = fgets(\$fp,255); // (2) fclose(\$fp); // (3) // Affichage du résultat ----- echo "Le fichier contient : \$donnees"; ?></pre>	<pre>Le fichier contient : 1523</pre>

Vous le voyez il est relativement simple de lire ce qui se trouve dans un fichier :

1. On ouvre le fichier "`data.txt`" en lecture seule "`r`" avec la fonction `fopen()`.
2. La lecture s'effectue avec la fonction `fgets()` et on spécifie le nombre de caractères (ici `255` soit la première ligne).
3. Ensuite il ne reste plus qu'à refermer le fichier texte c'est la fonction `fclose()`.
4. Enfin on affiche le résultat , c'est la variable `$donnees` qui contient "`1523`".

Revenons à la première ligne. La commande "`r`" indique que l'on ouvre le fichier uniquement en lecture seule. Nous allons voir ci-dessous que pour l'ouvrir en *lecture/écriture*, il suffit de mettre "`r+`". Concernant la seconde fonction `fgets()`, on spécifie le nombre de caractères que l'on veut lire dans le fichier (`255`). Dans notre cas nous aurions très bien pu mettre (`$fp,4`) ; puisque `1523` ne comporte que `4` caractères = logique. Ceci dit, le fait de mettre systématiquement `255` n'engendre pas de problème dans notre cas, sachez tout de même que `255` est le nombre maximum de caractères par ligne, le `256e` passera à la ligne automatiquement à l'affichage.

Et devant vous yeux englués de bonheur :), voici le code php qui va vous permettre de réaliser un compteur de page. Notez qu'ici le fichier texte s'appelle `compteur.txt`.

Le code PHP de `compteur.php3`

(ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)

```
<?
$fp = fopen("compteur.txt","r+"); // 1.On ouvre le fichier en lecture/écriture
$nbvisites = fgets($fp,11); // 2.On récupère le nombre dans le fichier
$nbvisites++; // 3.On incrémente le nombre de visites (+1)
fseek($fp,0); // 4.On se place en début de fichier
fputs($fp,$nbvisites); // 5.On écrit dans le fichier le nouveau nb
fclose($fp); // 6.On ferme le fichier
print("$nbvisites visiteurs"); // 7.On affiche le compteur à l'écran
?>
```

Il vous suffit de placer ce code dans la page un `index.php3` de votre site. Ici la fonction `fseek()` permet de se replacer où l'on veut en l'occurrence "0", donc au début, ensuite avec `fputs()` on écrit dans le fichier à l'endroit spécifié.

Pour finir avec les fichiers, n'oubliez pas que votre fichier texte qui se trouve sur votre FTP doit avoir avoir tous les droits (`chmod 777`) pour que le script puisse écrire dedans.

A savoir : PHP permet également de créer et effacer des fichiers sur un serveur distant (FTP), je vous conseille de lire la documentation NEXEN ci-dessous pour en savoir encore plus.

5.8 Les fonctions utilisateurs

PHP propose de nombreuses fonctions, mais un autre avantage est de pouvoir créer les siennes à l'aide de `function()`. Ceci est vraiment très utile pour ne pas avoir à retaper des parties de code en entier.

Le code PHP de `fonction.php3`

(ne copier/coller pas ce code dans votre éditeur, retapez-le ou gare aux erreurs...)

```
<?
function Arial($size,$color,$texte)

print("<font face=Arial size=".$size." color=".$color.">".$texte."</font>");
```

```
?>
```

Le code PHP de `index.php3`

```
<?
Require("fonction.php3"); // on appelle la fonction
// affichage _____
Arial("2","red","Ici le texte ...");
Arial("3","#0F74A3","Le second texte ...");
?>
```

On appelle `index.php3` dans le navigateur, ce qui donne à l'écran

Ici le texte ...Le second texte ...

Reprenons depuis le début. Vous pouvez donner le nom que vous voulez à vos fonctions, ensuite entre parenthèses ce sont les arguments que vous entrez lors de l'utilisation de cette fonction. Ces arguments correspondent bien sûr à celles qui se trouvent entre accolades. Ici, il s'agit juste de donner la taille "\$size", la couleur "\$color" et le contenu du texte "\$texte". L'utilisation de vos fonctions se font de la même manière que les fonctions intégrées de PHP. Dans `Index.php3`, vous voyez que l'on appelle la fonction `Arial()` et qu'on lui donne les valeurs que l'on veut, pratique non ?

Pratique : Je vous conseille de créer un fichier `fonction.php3` qui va contenir toutes vos fonctions de texte, tableau, etc. Il suffit juste ensuite de mettre un `require()` en entête de chacun de vos autres fichiers php pour pouvoir utiliser toutes les fonctions.

Je pense que vous voyez de suite l'intérêt de créer ses propres fonctions. Cela permet une bien meilleure lecture du code et un gain de temps à la programmation.

Dans un premier temps nous en resterons là, mais nous reviendrons à des fonctions plus complètes au cours des exercices futurs.



5.9 Les variables d'environnement

Ici ça n'est pas vraiment un exercice que je vous propose, même si vous allez pouvoir tester ces variables mais plutôt des informations que je vous livre. PHP propose toute une série de variables qui sont déjà implantées dans le langage sans que vous ayez à les créer, on les appelle les variables d'environnement.

Ces variables s'écrivent toujours en **MAJUSCULES** et bien sûr précédées du signe dollar \$, vous pouvez les utiliser n'importe où dans vos scripts comme ci-dessous :

Code PHP	Ce qui donne à l'écran
<? print("Votre adresse IP est : \$REMOTE_ADDR"); ?>	Votre adresse IP est : 201.65.8.56

Voilà, rien de plus simple pour connaître l'adresse IP d'un visiteur :), il s'agit de la variable d'environnement **\$REMOTE_ADDR**.

Voici ci-dessous la liste exhaustive des variables d'environnement existantes :

Variabiles	Description	Résultat à l'écran (Free.fr)
\$DOCUMENT_ROOT	Racine du serveur	/var/www/php.proxad.net
\$HTTP_ACCEPT_LANGUAGE	Langage accepté par le navigateur	fr
\$HTTP_HOST	Nom de domaine du serveur	proxyphp3.free.fr
\$HTTP_USER_AGENT	Type de navigateur	Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)
\$PATH_INFO	Chemin web du script	/d2expert.free.fr/phpdebutant/fichier.php3
\$PATH_TRANSLATED	Chemin complet du script	/var/www/free.fr/3/d/2/e/x/d2expert/phpdebuta
\$REQUEST_URI	Chemin du script	/d2expert.free.fr/phpdebutant/fichier.php3
\$REMOTE_ADDR	Adresse IP du client	195.132.7.201
\$REMOTE_PORT	Port de la requête HTTP	45039
\$QUERY_STRING	Liste des paramètres passés au script	?var=23&data=ok
\$SERVER_ADDR	Adresse IP du serveur	212.27.32.44
\$SERVER_ADMIN	Adresse de l'administrateur du serveur	email@email.com
\$SERVER_NAME	Nom local du serveur	php.proxad.net
\$SERVER_SIGNATURE	Type de serveur	?
\$REQUEST_METHOD	Méthode d'appel du script	GET

Pour finir (les puristes vont peut-être m'en vouloir d'en parler ici), je voulais m'attarder quelques instants sur ce qui est une fonction et non pas une variable d'environnement. Je veux parler de **phpinfo()**. À l'aide de celle-ci et en 10 secondes vous aller pouvoir connaître la configuration et la version exacte de PHP qu'utilise le serveur où vous êtes hébergé. Faites un petit fichier comme ci-dessous ou **cliquez ici** pour tout savoir du serveur FREE.

Code PHP	Ce qui donne à l'écran
<? phpinfo(); ?>	Cela serait trop long :)

5.10 Quelques fonctions utiles

Fonction	Description	Code PHP	Rés.
addslashes()	Ajoute des anti-slashes devant les caractères spéciaux	\$res = addslashes ("L'a");	L\`a
stripslashes()	Retire les anti-slashes devant les caractères spéciaux.	\$res = stripslashes ("L\`a");	L'a
dechex()	Retourne la valeur Hexadécimale d'un nombre (ici 2548).	\$res = dechex ("2548");	9f4
ceil()	Retourne le nombre entier supérieur ici (12,1).	\$res = ceil ("12.1"); textcolorblue*	13

<code>chunk_split()</code>	Permet de scinder une chaîne en plusieurs morceaux.	<code>\$res = chunk_split("DGDFEF","2","-");</code>	DG- DF- EF-
<code>htmlentities()</code>	Remplace les caractères par leur équivalent HTML (si ils existent).	<code>\$res = htmlentities("&");</code>	&
<code>strstr()</code>	Recherche le premier caractère 'p' dans la chaîne et affiche le reste de la chaîne y compris le 'p'.	<code>\$res = strstr("webmaster@phpdebutant.com", "p");</code>	phpdebutant.com
<code>strlen()</code>	Retourne la longueur de la chaîne	<code>\$res = strlen("lachedecaracteres");</code>	20 la chaîne de carac- teres LA CHAINE
<code>strtolower()</code>	Passe tous les caractères en minuscules.	<code>\$res = strtolower("LA CHAINE DE caRAc- tERes");</code>	LA CHAINE DE CA- RAC- TERES
<code>strtoupper()</code>	Passe tous les caractères en MAJUSCULES.	<code>\$res = strtoupper("LA CHAINE DE caRAc- tERes");</code>	LA CHAINE DE CA- RAC- TERES
<code>str_replace()</code>	Remplace un caractère par un autre dans une chaîne. Tiens compte de la casse.	<code>\$res = str_replace("a","o","Lalala");</code>	Lololo
<code>trim()</code>	Efface les espaces blancs (\n, \r, etc) au début et à la fin d'une chaîne (pas au milieu).	<code>\$res = trim(" Salut le monde ");</code>	Salut le monde Salut
<code>ucfirst()</code>	Met la première lettre de chaque chaîne en Majuscule.	<code>\$res = ucfirst("salut le monde. ca va?");</code>	le monde. ca va? Salut
<code>ucwords()</code>	Met la première lettre de chaque mot d'une chaîne en Majuscule.	<code>\$res = ucwords("salut le monde");</code>	Le Monde
<code>strpos()</code>	Recherche la position du premier caractères trouvé. Retourne le nombre de caractères placés avant lui (ici 4).	<code>\$res = strpos("abcdef","e");</code>	4
<code>ereg()</code>	Recherche si une chaîne de caractère est contenue dans une autre (ex. recherche si "ABCDE" contient "BCD").	<code>if(ereg("BCD","ABCDEF")) echo "oui"; else echo "non";</code>	oui

* La virgule sous PHP est représentée par un point ".", ainsi **12,1 s'écrit : 12.1 !**

Notez : Dans la colonne "Code PHP" du tableau ci-dessus, j'ai volontairement omis de mettre un : `echo $res;` pour afficher le résultat, mais bien sûr vous devrez le rajouter à chaque fois pour que le résultat apparaisse à l'écran.

Pour avoir la liste exhaustive de toutes les fonctions de PHP4, rendez-vous une fois de plus chez <http://www.nexen.net> :).

5.11 SQL/MySQL (Create, Alter & Drop)

5.11.1 CREATE TABLE

Le langage SQL (Structured Query Langage) permet d'interroger une base de données qui supporte ce langage, les plus connues sont MS Access, mSQL, MySQL et PostgreSQL. Comme vous le savez nous utiliserons MySQL (**attention mSQL n'est pas MySQL**), nuance importante.

Comme je vous le disais en introduction nous allons travailler chez [Free.Fr](http://www.free.fr), vous pouvez donc vous rendre à <http://sql.free.fr/phpMyAdmin/> pour accéder à votre base de données MySQL via l'interface [phpMyAdmin](http://www.phpmyadmin.net). Notez que 99% des hébergeurs proposent phpMyAdmin pour administrer votre base de données (cela facilite grandement la vie).

La première chose que nous allons devoir faire c'est de créer une table, c'est la commande **CREATE TABLE**, voyez la syntaxe ci-dessous qui permet de créer `clients_tbl`. Il est important de savoir comment l'on créé une table en SQL avant de passer par l'interface phpMyAdmin pour le faire :

Syntaxe SQL pour créer la table : `clients_tbl`

```
CREATE TABLE clients_tbl (id INT not null AUTO_INCREMENT, prenom VARCHAR (50) not null, nom VARCHAR (50) not null, ne_le DATE not null, ville VARCHAR (90) not null, enfants INT not null, PRIMARY KEY (id))
```

Commentons la table en la visualisant sous [phpMyAdmin](#) pour MySQL

Champ	Type	Null	Defaut	Extra	Commentaire
id	int(11)	Non	0	auto_increment	INT pour Integer (nombre entier) : C'est l'id qui va nous permettre de classer nos enregistrements, l'auto-increment se charge d'affecter un nouveau numéro aux nouveaux enregistrements qui s'ajoutent dans la table.
prenom	varchar(50)	Non			Varchar pour Chaîne de caractères : Nous l'utilisons quand nous souhaitons que le champs puisse recevoir aussi bien des Chaîne mélangeant texte, nombre, etc ...
nom	varchar(50)	Non			Idem que PRENOM , la valeur placée entre parenthèses définit le nombre maximum de caractères que le champs accepte, ici 50. Si vous essayez d'insérer une chaîne de 55 caractères, les 5 derniers seront coupés.
ne_le	date	Non	0000-00-00		Date : Permet de stocker des dates, mais attention TOUJOURS au format US soit : Année/mois/jour, si vous envoyez un format français J/M/A vous obtiendrez une date fausse dans la base.
ville	varchar(90)	Non			Idem que PRENOM , ici nous avons spécifier 90 caractères car certaines villes comportent beaucoup de caractères (avec 90 nous sommes sûr)
enfants	int(11)	Non	0		INT (nombre entier) : Dans ce champs n'arriveront que des nombres entiers donc le INT voir SMALL INT et de rigueur.

Ci-dessous la table : `clients_tbl` une fois créée

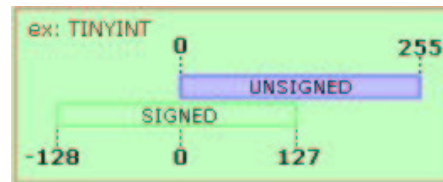
```
id prenom nom ne_le ville enfants
```

Bien sûr pour le moment cette table ne comporte aucun enregistrement ...

Dès cet instant vous devez consulter [la documentation officielle de MySQL](#) pour connaître tous les types de champs que vous pouvez utiliser dans une table, voici quelques exemples :

- **TINYINT** : Entier de 0 à 255 (unsigned)
- **SMALLINT** : Entier de 0 à 65535 (unsigned)
- **MEDIUMINT** : Entier de 0 à 16777215 (unsigned)
- **INT** : Entier de 0 à 4294967295 (unsigned)
- **BIGINT** : Entier de 0 à 18446744073709551615 (unsigned)
- **DECIMAL** : Un nombre à virgule flottante
- **DATE** : Une date, va de '1000-01-01' à '9999-12-31'
- **DATETIME** : Date et Heure, va de '1000-01-01 00 :00 :00' à '9999-12-31 23 :59 :59'
- **TIMESTAMP** : Date et Heure exprimée en secondes depuis le 1er janvier 1970. Va de '1970-01-01 00 :00 :00' à quelque part, durant l'année 2037
- **TIME** : Une mesure de l'heure, va de '-838 :59 :59' à '838 :59 :59'
- **YEAR** : Une année, va de 1901 à 2155
- **CHAR** : Chaîne de caractère de taille fixe, va de 1 à 255 caractères
- **VARCHAR** : Chaîne de caractère de taille variable, va de 1 à 255 caractères
- **TINYTEXT** ou **TINYBLOB** : Un objet BLOB ou TEXT, longueur maximale de 255
- **TEXT** ou **BLOB** : Un objet BLOB ou TEXT, longueur maximale de 65535
- **MEDIUMTEXT** ou **MEDIUMBLOB** : Un objet BLOB ou TEXT, longueur maximale de 16777215

– **LONGTEXT** ou **LOB** : Un objet BLOB ou TEXT, longueur maximale de 4294967295
A noter : Concernant la notion de **UNSIGNED** est importante pour les Integer (INT, SMALLINT, etc.) je vous propose un petit schéma simple pour vous expliquer la différence entre un Integer UNSIGNED ou pas :



Le **TINYINT** couvre de 0 à 255 quand il est **UNSIGNED**, mais il couvrira de -128 à 127 (soit toujours 255) si il ne l'est pas, Ceci est valable pour tous les Integers (nombres entiers).

5.11.2 ALTER TABLE

Une fois que votre table est créée vous pourrez bien sûr la modifier en utilisant **ALTER TABLE**, voyez l'exemple ci-dessous pour ajouter un champs :

Syntaxe SQL pour ajouter le champ 'tel' à la table : `clients_tbl`

ALTER TABLE `clients_tbl` **ADD** tel INT not null

Ci-dessous la table `clients_tbl` une fois modifiée

id	prenom	nom	ne_le	ville	enfants	tel
----	--------	-----	-------	-------	---------	-----

Voilà ci-dessus notre table a été modifié grâce à **ALTER TABLE** puis **ADD** (comme ajouter). Maintenant supprimons le champs tel de la table :

Syntaxe SQL pour supprimer le champ 'tel' de la table : `clients_tbl`

ALTER TABLE `clients_tbl` **DROP** tel

Ci-dessous la table `clients_tbl` une fois modifiée

id	prenom	nom	ne_le	ville	enfants
----	--------	-----	-------	-------	---------

Voilà ci-dessus notre table a été modifié grâce à **ALTER TABLE** puis **DROP**. Cette commande **DROP** va également nous être utile pour supprimer une table complète.

5.11.3 DROP TABLE

Il s'agit de la commande qui permet de supprimer une table complète, attention en supprimant une table vous perdez tout ce qu'elle contenait, donc à utiliser avec prudence !

Syntaxe SQL pour supprimer la table : `clients_tbl`

DROP TABLE `clients_tbl`

Notre table `clients_tbl` a été complètement effacée avec la commande **DROP TABLE**, dans notre cas elle ne contenait aucunes informations mais dans le cas contraire, cela signifie la perte de toutes les données de la table.

Pour finir : Nous avons vu ici la majeure partie des choses les plus importantes à connaître en ce qui concerne la création et la modification des tables. Dans les prochains exercices nous verrons comment traiter les données d'une table avec les commandes **SELECT**, **INSERT**, **UPDATE** et **DELETE**.

5.12 SQL/MySQL (Insert et Select)

Pour commencer nous allons re-crée la table `client_tbl` de l'exercice précédent.

Syntaxe SQL pour créer la table : `clients_tbl`

```
CREATE TABLE clients_tbl (id INT not null AUTO_INCREMENT, prenom VARCHAR (50) not null , nom VARCHAR (50) not null , ne_le DATE not null , ville VARCHAR (90) not null , enfants INT not null , PRIMARY KEY (id))
```

Ci-dessous la table : `clients_tbl` une fois créée

id	prenom	nom	ne_le	ville	enfants
----	--------	-----	-------	-------	---------

Bien sûr pour le moment cette table ne comporte aucun enregistrement ...

La commande `INSERT INTO`

Cette commande permet d'insérer des enregistrements dans une table en l'occurrence `clients_tbl`.

– `INSERT INTO clients_tbl(id,prenom,nom,ne_le,ville,enfants) VALUES(?, 'Patrick', 'Martin', '1965/10/08', 'Bordeaux', '2')`
ou bien cette autre requête qui aura le même résultat :

– `INSERT INTO clients_tbl VALUES(?, 'Patrick', 'Martin', '1965/10/08', 'Bordeaux', '2')`

Ci-dessous la table : `clients_tbl` avec le nouvel enregistrement

id	prenom	nom	ne_le	ville	enfants
1	Patrick	Martin	1965/10/08	Bordeaux	2

Dans le premier exemple nous avons spécifier les noms des champs (entre parenthèses) juste après le nom de la table. Dans ce cas il n'est pas obligatoire de le faire si dans les valeurs vous spécifiez une valeur par champs, mysql affectera les valeurs dans l'ordre données.

Par contre si vous ne donnez que deux valeurs, il sera important de toujours spécifier les champs dans lesquels elle doivent être insérées, exemple :

– `INSERT INTO clients_tbl(id,nom) Values(?, 'Dupond')`

– ou encore :

– `INSERT INTO clients_tbl(nom,enfants) Values('Dupond', '2')`

Notez : La valeur du champs `id` est vide, je vous rappelle que ce champs est en `auto-increment` dans notre base, un nombre automatique sera donc attribué à chaque nouvel enregistrement, dans ce cas la valeur dans la requête peut rester vide!

Important : Si vous voulez n'insérer que quelques-unes des valeurs d'un enregistrement , vous devrez avoir spécifier lors de la création de la Table que les autres champs peuvent rester vides! (NULL).

La commande `SELECT`

Nous allons travailler à partir de la table ci-dessous qui comporte 5 enregistrements :

Ci-dessous le contenu final de la table : `clients_tbl`

id	prenom	nom	ne_le	ville	enfants
1	Patrick	Martin	1965/10/08	Bordeaux	2
2	Julien	Lebreton	1964/02/21	Paris	2
3	Marc	Richard	1958/04/15	Lille	4
4	Francis	Perrin	1982/12/05	Paris	0
5	Daniel	Bacon	1974/07/13	Reims	1

Ceci est une représentation en mode console et non pas sous phpMyAdmin

Admettons que nous voulions affiché uniquement les personnes qui n'ont que 2 enfants, la requête SQL sera :

`SELECT * FROM clients_tbl WHERE enfants='2'`

Soit en Français :

SELECT

Je Sélectionne

*	Tous les champs
FROM clients_tbl	Depuis la table client_tbl
WHERE enfants='2'	Quand le champs enfants est égal à 2

Ce qui donne comme résultat :

1	Patrick	Martin	1965/10/08	Bordeaux	2
2	Julien	Lebreton	1964/02/21	Paris	2

Voici la même requête, mais cette fois nous n'allons demander l'affichage que des noms et prénoms :

SELECT nom,prenom FROM clients_tbl WHERE enfants='2'

Soit en Français :

SELECT	Je Sélectionne
nom,prenom	Les champs nom et prenom
FROM clients_tbl	Depuis la table client_tbl
WHERE enfants='2'	Quand le champs enfants est égal à 2

Ce qui donne comme résultat :

Patrick Martin

Julien Lebreton

Reprenons la première requête et ajoutons d'autres conditions (**WHERE**), à savoir que nous allons demander l'affichage des personnes qui ont 1 ou 2 enfants et qui habitent la ville de Paris :

SELECT * FROM clients_tbl WHERE enfants='0' OR enfants='2' AND ville='Paris'

– Vous devez faire la différence entre **OR** et **AND**, nous pouvons l'analyser ainsi :

Si le nombre d'enfant est 0 et que la ville est Paris , c'est OK.

Et si le nombre d'enfant est 2 et que la ville est Paris , c'est OK.

Il faut que les 2 conditions soient respectées pour que l'enregistrement soit affiché !

Voici les opérateurs possibles :

+	Addition
-	Soustraction
*	Multiplication
/	Division
<	Plus petit que
<=	Plus petit ou égal à
=	Égal à
!= ou <>	N'est pas égal à
>=	Plus grand ou égal à
and	ET
or	OU
not	Négation

Pour finir voici quelques autres exemples de SELECT :

SELECT * FROM clients_tbl WHERE ne_le < "1978-01-01"

SELECT * FROM clients_tbl WHERE enfants != '0'

SELECT * FROM clients_tbl WHERE nom LIKE 'le'

```
SELECT * FROM clients_tbl WHERE nom LIKE '%ri%'
```

Il serait bien trop long de lister tous les exemples possibles, mais n'hésitez pas à consulter **la documentation en Français de Nexen.net sur MySQL** à l'adresse <http://www.nexen.net>.

5.13 SQL/MySQL (Delete et Update)

Ici nous continuons de travailler la table `client_tbl` créée précédemment.

La commande `UPDATE`

Cette commande permet de modifier les valeurs d'un enregistrement déjà présent dans la table :

– `UPDATE clients_tbl SET prenom='Jacques' WHERE id='1'`

Cette commande ne pose vraiment pas de problème particulier, décortiquons la syntaxe :

<code>UPDATE clients_tbl</code>	Mise à jour de la table Clients_tbl
<code>SET prenom='Jacques'</code>	Modifier le champs prenom pour la valeur Jacques
<code>WHERE id='1'</code>	Quand le champs id est égal à 1

Ci-dessous l'enregistrement de la table : `clients_tbl` une fois modifié.

id	prenom	nom	ne_le	ville	enfants
1	Jacques	Martin	1965/10/08	Bordeaux	2

Bien sûr nous pouvons changer plusieurs valeurs d'un même enregistrement dans la même requête :

– `UPDATE clients_tbl SET prenom='Jean-Pierre', nom='Papin', ville='Marseille', enfants='3' WHERE id='1'`

Vous le voyez, il suffit de séparer les "**champs/valeurs**" par une virgule, ce qui donne comme résultat dans la table :

Ci-dessous l'enregistrement de la table : `clients_tbl` une fois modifié.

id	prenom	nom	ne_le	ville	enfants
1	Jean-Pierre	Papin	1965/10/08	Marseille	3

La commande `DELETE`

Bon vous l'aurez sans doute compris cette commande sert à supprimer un ou plusieurs enregistrements d'une table ainsi :

– `DELETE FROM clients_tbl WHERE id='1'`

Là non plus la commande ne pose vraiment pas de problème particulier , décortiquons la syntaxe :

<code>DELETE FROM clients_tbl</code>	Effacer de la table Clients_tbl
<code>WHERE id='1'</code>	Quand l'id de l'enregistrement est égal à 1

Notez : Pour finir notez que les opérateurs peuvent s'appliquer également dans le cadre d'un `UPDATE` ou d'un `DELETE`. En fait ils peuvent s'appliquer à n'importe quelle requête SQL.

Conclusion : Je n'ai abordé dans les 3 exercices sur SQL, que des syntaxes simples volontairement pour ne pas trop vous embrouiller. Nous aurons l'opportunité dans les futurs exercices de voir par exemple des requêtes `UPDATE` qui mettent à jour plusieurs tables en même temps, mais ne brûlons pas les étapes;) ... Ceci dit si vous vous sentez à l'aise n'hésitez pas !

5.14 Fonctions PHP pour MySQL

Les fonctions PHP pour MySQL commence toujours par " **MYSQL_** ", en voici ci-dessous la liste exhaustive. En règle générale je n'utilise pas plus de 6 ou 7 fonctions, c'est d'ailleurs sur celles-ci que je mettrai l'accent plus bas dans cet exercice, ceci dit, je vous conseille tout de même les tester toutes.

Fonctions	Descriptions
<code>mysql_affected_rows()</code>	Retourne le nombre de rangée affectées par la dernière requête faite sur la base de données.
<code>mysql_close()</code>	Ferme la connexion à une base de données.
<code>mysql_connect()</code>	Établit une connexion vers la base de données spécifiée dans les arguments. Si cette base se trouve sur un port différent, faites suivre le nom de la machine par (:) puis le numéro du port (ex. : 8080). Cette connexion est automatiquement fermée à la fin de l'exécution du script sauf si une fonction mysql_close() intervient auparavant.
<code>mysql_create_db()</code>	Permet de créer une nouvelle base de données.
<code>mysql_data_seek()</code>	Déplace le pointeur interne de la rangée du résultat vers la rangée spécifiée.
<code>mysql_db_query()</code>	Utilisez cette fonction avec mysql_fetch_row() pour passer à la rangée spécifiée et récupérer les données.
<code>mysql_drop_db()</code>	Permet d'exécuter une requête SQL sur une ou plusieurs tables d'une base de données.
<code>mysql_errno()</code>	Permet de supprimer une base de données. Dans ce cas toutes les données sont perdues.
<code>mysql_error()</code>	Retourne le numéro de l'erreur générée lors de la dernière action sur la base de donnée.
<code>mysql_fetch_array()</code>	Retourne la description textuelle d'une erreur générée par la dernière action sur une base de données.
<code>mysql_fetch_field()</code>	Retourne un tableau qui représente tous les champs d'une rangée dans le résultat. Chaque appel récupère la prochaine rangée et ce jusqu'à ce qu'il n'y en ait plus. Chaque valeur de champ est stockée de deux façons : elle est indexée par un offset qui commence par '0', et indexée par le nom du champ.
<code>mysql_fetch_lengths()</code>	Récupère l'information attachée à un champs du résultat. Ces champs sont numérotés à partir de zéro.
<code>mysql_fetch_object()</code>	Retourne un tableau d'une longueur spécifiée pour chaque champ de résultat. Cette fonction est semblable à mysql_fetch_array() et à mysql_fetch_row() . Mais à la place, elle retourne un objet. Chaque champ du résultat correspond à une propriété dans l'objet renvoyé. Chaque appel à mysql_fetch_object() retourne la rangée suivante, ou False s'il ne reste plus de rangée.
<code>mysql_fetch_row()</code>	Retourne un tableau qui représente tous les champs d'une rangée du résultat. Chaque appel récupère la prochaine rangée et ce jusqu'à ce qu'il n'y en ait plus. C'est la méthode la plus rapide pour obtenir des résultats à partir d'une requête.
<code>mysql_field_flags()</code>	Permet d'obtenir une description des options rattachées au champs spécifié.
<code>mysql_field_len()</code>	Retourne la longueur maximale du champ spécifié.
<code>mysql_field_name()</code>	Retourne le nom d'une colonne. L'argument champ correspond à un offset numéroté à partir de zéro.
<code>mysql_field_seek()</code>	Déplace le pointeur interne du champ vers le champs spécifié. Le prochain appel vers mysql_field_seek() retournera l'information de ce champs.
<code>mysql_field_table()</code>	Retourne le nom de la table pour le champ spécifié.
<code>mysql_field_type()</code>	Retourne le type d'un champ particulier dans le résultat obtenu.
<code>mysql_free_result()</code>	Libère la mémoire associée au résultat spécifié. Elle n'est toutefois pas strictement nécessaire, car cette mémoire est automatiquement vidée lorsqu'un script termine son exécution.
<code>mysql_insert_id()</code>	Après l'insertion d'un nouvel enregistrement avec un champ auto_increment , la fonction mysql_insert_id() retourne l' ID qui vient d'être affecté au nouvel enregistrement.
<code>mysql_list_dbs()</code>	Interroge le serveur pour obtenir une liste de bases de données. Elle retourne un pointeur de résultat qui pourra être exploité avec mysql_fetch_row() et d'autres fonctions similaires.
<code>mysql_list_fields()</code>	Retourne un pointeur de résultat correspondant à une requête sur une liste de champs pour la table spécifiée. Ce pointeur pourra être exploité par toutes les fonctions qui recherchent des rangées à partir d'un résultat. Notez que l'argument lien reste optionnel.



<code>mysql_list_tables()</code>	Retourne le pointeur de résultat d'une liste de tables pour la base de données spécifiée. Ce pointeur pourra être exploité par toutes les fonctions qui recherchent des rangées à partir d'un résultat. Notez que l'argument lien reste optionnel.
<code>mysql_num_fields()</code>	Retourne le nombre de champs dans un résultat.
<code>mysql_num_rows()</code>	Retourne le nombre de rangées dans un résultat.
<code>mysql_pconnect()</code>	Cette fonction opère de la même manière que <code>mysql_connect()</code> , sauf que la connexion ne se referme pas à la fin de l'exécution du script sauf si un <code>mysql_close()</code> se trouve en fin de script.
<code>mysql_query()</code>	Permet d'exécuter une requête SQL sur une ou plusieurs tables d'une base de données. Si la requête exécute une instruction : INSERT , DELETE ou UPDATE , une valeur booléenne sera retournée (0 ou 1). Dans le cas d'une requête de type SELECT , vous obtiendrez un identifiant de résultat. Retourne la valeur du champ spécifié dans la rangée concernée. L'argument champ peut être un numéro et, dans ce cas, il sera considéré comme un champ offset. Il peut également désigner le nom de la colonne, avec éventuellement celui de la table. Enfin, il peut également renvoyer à un alias.
<code>mysql_result()</code>	Retourne la valeur du champ spécifié dans la rangée concernée. L'argument champ peut être un numéro et, dans ce cas, il sera considéré comme un champ offset. Il peut également désigner le nom de la colonne, avec éventuellement celui de la table. Enfin, il peut également renvoyer à un alias.
<code>mysql_select_db()</code>	Sélectionne la base de données par défaut.

Fonctions : `mysql_connect()`, `_select_db()`, `_query()`, `_numrows()`, `_close()`

Ces fonctions sont le minimum que vous utiliserez à chaque fois que vous interrogerez une base de données MySQL, voyez le code ci-dessous (nous avons repris notre table `clients_tbl`).

Code PHP

```
<?
$db = mysql_connect('sql.free.fr', 'login', 'password'); // 1
mysql_select_db('nom_de_la_base',$db); // 2
$req = mysql_query('SELECT * FROM clients_tbl'); // 3
$res = mysql_numrows($req); // 4

echo 'Il y a '.$res.' enregistrement(s) dans la table Clients.'; // 5

mysql_close($db); // 6
?>
```

Donne à l'écran

Il y a 5 enregistrement(s) dans la table Clients.

Explication du code ci-dessus :

- On se connecte à la base de données :
 - Host** : Dans notre cas le HOST est "sql.free.fr", vous pouvez également utiliser "localhost" par défaut.
 - Login** : Ensuite vous devez mettre le "login" pour accéder à la base (chez les hébergeurs gratuits c'est souvent le même login que l'accès FTP).
 - Password** : Et pour finir le "mot de passe", là aussi il s'agit très souvent du même password que l'accès FTP.
- On sélectionne la base de données, en effet je vous rappelle que MySQL est un [serveur de bases de données](#), donc il peut contenir plusieurs bases. Bien sûr dans votre cas si vous êtes chez un hébergeur gratuit, vous n'avez en général droit qu'à une seule base, mais MySQL ne le sait pas ;), il faut donc lui spécifier sur quelle base vous souhaitez vous connecter.

Notez : Chez Free.fr et Nexen le nom de la base est souvent le même que le login de connexion.
- La fonction `mysql_query()` permet de passer une requête SQL vers la base de données, c'est évidemment l'un des attraits intéressants de PHP (notez que nous initialisons au passage la variable `$req` qui contient la requête).
- La fonction `mysql_numrows()` permet de compter le nombre d'enregistrements que retourne la requête "`$req`" dans notre cas : 5 puisqu'il s'agit d'un simple "select" sur la table sans aucune condition, nous initialisons donc une variable `$res` qui contient : 5.

5. Il ne reste plus qu'à afficher le nombre de résultat avec un `echo()` de `$res`.
6. Et pour finir on referme la connexion - ouverte avec `mysql_connect` - avec la fonction `Mysql_close()`. Cette fonction n'est pas vraiment obligatoire avec un `Mysql_connect()`, car par défaut la connexion sera coupée automatiquement à la fin de l'exécution du script.

5.15 Interroger une table MySQL

Maintenant que nous nous sommes connectés à la base de données, nous allons interroger une table pour en extraire les résultats puis les ranger dans un ordre précis. Créons d'abord une table comme ci-dessous.

Requête SQL : CREATE TABLE famille.tbl (id int(11) NOT NULL auto_increment, nom varchar(255) NOT NULL, prenom varchar(255) NOT NULL, statut varchar(255) NOT NULL, date date DEFAULT '0000-00-00' NOT NULL, PRIMARY KEY (id), KEY id (id), UNIQUE id_2 (id));
 INSERT INTO famille.tbl VALUES("", 'Dupond', 'Grégoire', 'Grand-père', '1932-05-17');
 INSERT INTO famille.tbl VALUES("", 'Dupond', 'Germaine', 'Grand-mère', '1939-02-15');
 INSERT INTO famille.tbl VALUES("", 'Dupond', 'Gérard', 'Père', '1959-12-22');
 INSERT INTO famille.tbl VALUES("", 'Dupond', 'Marie', 'Mère', '1961-03-02');
 INSERT INTO famille.tbl VALUES("", 'Dupond', 'Julien', 'Fils', '1985-05-17');
 INSERT INTO famille.tbl VALUES("", 'Dupond', 'Manon', 'Fille', '1990-11-29');

Structure de la table (PhpMyAdmin) :

Champ	Type	Null	Defaut	Extra
id	int(11)	Non	0	auto_increment
nom	varchar(255)	Non		
prenom	varchar(255)	Non		
statut	tinyint(255)	Non	0	
date	date	Non	0000-00-00	

Contenu de la table "famille.tbl"

id	nom	prenom	statut	date
1	Dupond	Grégoire	Grand-père	1932-05-17
2	Dupond	Germaine	Grand-mère	1939-02-15
3	Dupond	Gérard	Père	1959-12-22
4	Dupond	Marie	Mère	1961-03-02
5	Dupond	Julien	Fils	1985-05-17
6	Dupond	Manon	Fille	1990-11-29

Affichages des résultats tels qu'ils sont dans la table sans condition.

Code PHP



```

<?
// on se connecte à MySQL
$db = mysql_connect('localhost', 'login', 'password');

// on sélectionne la base
mysql_select_db('nom_de_la_base',$db);

// on créer la requete SQL et on l'envoie
$sql = 'SELECT nom,prenom,statut,date FROM famille_tbl';

// on envoie la requete
$req = mysql_query($sql) or die('Erreur SQL !<br>'.$sql.'<br>'.mysql_error());

// on fait une boucle qui va faire un tour pour chaque enregistrements
while($data = mysql_fetch_array($req))
{
// on affiche les informations de l'enregistrements en cours
echo '<b>'.$data['nom'].' '.$data['prenom'].'</b> ('.$data['statut'].')';
echo ' <i>date de naissance : '.$data['date'].'</i><br>';
}

// on ferme la connexion à mysql
mysql_close();
?>

```

Donne à l'écran

Dupond Grégoire (Grand-père), *date de naissance : 1932-05-17*

Dupond Germaine (Grand-mère), *date de naissance : 1939-02-15*

Dupond Gérard (Père), *date de naissance : 1959-12-22*

Dupond Marie (Mère), *date de naissance : 1961-03-02*

Dupond Julien (Fils), *date de naissance : 1985-05-17*

Dupond Manon (Fille), *date de naissance : 1990-11-29*

Explication : Voici donc notre première boucle sur une table, champagne! :). Plus sérieusement, vous vous apercevez que les résultats qui s'affichent sont exactement dans le même ordre que la table et pour cause, nous n'avons pas spécifié de condition dans notre requête (2), donc dans ce cas, la requête scanne la table de haut en bas.

Remarquez que la fonction `mysql_fetch_array()` renvoie un tableau dont les clés sont les noms des champs sélectionnés. On a aussi rajouté après `mysql_query()` ceci : `or die('Erreur SQL !
'.$sql.'
'.mysql_error());`. Cela veut dire qu'en cas d'erreur dans la requete vers mysql, ce qui arrive fréquemment, php va afficher un message indiquant l'erreur renvoyé par mysql (grâce à `mysql_error()`) ce qui fournit une aide précieuse pour comprendre le problème.

Vous notez également que les dates de naissances sont au format US, ceci est normal puisque nous avons défini un type DATE dans notre table, nous verrons plus bas comment convertir les dates US au format FR.

Nous allons maintenant faire plusieurs testes en ne changeant uniquement que la requête SQL (2). **Le reste du code ne change pas.**

Affichages des résultats par ordre alphabétique de prénom.

Le Code PHP de la requête

```

<?
// Gardez le code ci-dessus, changez juste la requête SQL!
$sql = 'SELECT nom,prenom,statut,date FROM famille_tbl ORDER BY prenom';

// L'opérateur ORDER BY permet de classer soit alphabétiquement
// soit numériquement suivant le type du champ.

// Si l'on souhaite classé en décroissant (ex. de Z à A), nous
// y ajouterons DESC soit : ORDER BY prenom DESC
?>

```

Donne à l'écran

Dupond Gérard (Père), *date de naissance : 1959-12-22*
Dupond Germaine (Grand-mère), *date de naissance : 1939-02-15*
Dupond Grégoire (Grand-père), *date de naissance : 1932-05-17*
Dupond Julien (Fils), *date de naissance : 1985-05-17*
Dupond Manon (Fille), *date de naissance : 1990-11-29*
Dupond Marie (Mère), *date de naissance : 1961-03-02*

Affichages des résultats par comparaison de date.**Le Code PHP de la requête**

```

<?
// Gardez le code ci-dessus, changez juste la requête!
$sql = 'SELECT nom,prenom,statut FROM famille_tbl WHERE date<1960-01-01';

// L'avantage d'avoir un type DATE dans notre base de données, c'est que
// nous pouvons comparer des dates dans la requête SQL.
// Ici nous ne souhaitons afficher que les membres de la famille qui sont
// nés avant le 1er janvier 1960, soit : WHERE date<1960-01-01

```

```

?>

```

Donne à l'écran

Dupond Grégoire (Grand-père), *date de naissance : 1932-05-17*
Dupond Germaine (Grand-mère), *date de naissance : 1939-02-15*
Dupond Gérard (Père), *date de naissance : 1959-12-22*

Affichages des résultats avec le commande LIKE.

La commande LIKE en SQL permet de fouiller le contenu de chaque champ. Je m'explique, je recherche tous les enregistrements dont le champ "**prenom**" commence par la lettre "**G**", voyons la syntaxe ci-dessous.

Le Code PHP de la requête

```

<?
// Gardez le code ci-dessus, changez juste la requête!
$sql = "SELECT nom,prenom,statut,date FROM famille_tbl WHERE prenom LIKE 'M%'";

// Le signe pourcentage "%" placé après le M, indique que la lettre M peut
// être suivie d'autres caractères, mais pas précédée!
// Notez aussi que LIKE n'est pas sensible à la casse, cela veut dire que
// la requête cherchera aussi bien des M majuscules que minuscules.

```

```

?>

```

Donne à l'écran

Dupond Marie (Mère), *date de naissance : 1961-03-02*
Dupond Manon (Fille), *date de naissance : 1990-11-29*

Maintenant voyons la même chose mais cette fois nous allons chercher la syllabe "**ma**" dans le champ "**prenom**", qu'elle soit placée au début ou au milieu d'autres caractères.

Le Code PHP de la requête

```

<?
// Gardez le code ci-dessus, changez juste la requête!
$sql = "SELECT * FROM famille_tbl WHERE prenom LIKE '%MA%'";

// Le signe pourcentage "%" placé avant et après MA indique que la syllabe
// peut-être précédée ou suivie de caractères.
// Une fois de plus notez que LIKE n'est pas sensible à la casse, la requête
// cherchera aussi bien des MA majuscules que des ma en minuscules.

?>

```

Donne à l'écran

Dupond Germaine (Grand-mère), *date de naissance : 1939-02-15*

Dupond Marie (Mère), *date de naissance : 1961-03-02*

Dupond Manon (Fille), *date de naissance : 1990-11-29*

Bien sûr, il est possible de mettre plusieurs conditions dans la même requête, exemple :

– **SELECT * FROM famille_tbl WHERE prenom LIKE '%MA%' AND date<'1960-01-01' ORDER BY prenom**
 Pour terminer voici comment vous allez pouvoir convertir la date du format US au format FR, une fois que vous avez récupéré l'information depuis la table.

Le Code PHP

```

<?
// on se connecte à MySQL
$db = mysql_connect('localhost', 'login', 'password');

// on sélectionne la base
mysql_select_db('nom_de_la_base',$db);

// on crée la requete SQL et on l'envoie
$sql = 'SELECT nom,prenom,statut,date FROM famille_tbl';

// on envoie la requete
$req = mysql_query($sql) or die('Erreur SQL !<br>'.$sql.'<br>'.mysql_error());

// on fait une boucle qui va faire un tour pour chaque enregistrements
while($data = mysql_fetch_array($req))
{
$a = substr($data['date'], 0, 4); // conversion
$m = substr($data['date'], 5, 2); // de la date
$j = substr($data['date'], 8, 2); // au format
$date = $j.'-'. $m.'-'. $a; // Français

// on affiche les informations de l'enregistrements en cours
echo '<b>'.$data['nom'].' '.$data['prenom'].'</b> ('.$data['statut'].)';
echo ' <i>date de naissance : '.$date.'</i><br>';
}

// on ferme la connexion à mysql
mysql_close();
?>

```

Donne à l'écran

Dupond Grégoire (Grand-père), *date de naissance : 17-05-1932*

Dupond Germaine (Grand-mère), *date de naissance : 15-02-1939*

Dupond Gérard (Père), *date de naissance : 22-12-1959*

Dupond Marie (Mère), *date de naissance : 02-03-1961*

Dupond Julien (Fils), *date de naissance : 17-05-1985*

Dupond Manon (Fille), *date de naissance : 29-11-1990*

5.16 Alimenter une ou plusieurs tables mySQL

Dans cet exercice nous allons voir comment alimenter une ou plusieurs tables avec les données qui proviennent d'un même formulaire.

5.16.1 Alimenter une table

Pour commencer vous allez créer la table `infos.tbl` dans phpMyAdmin comme suit :

requête SQL :

```
CREATE TABLE infos.tbl (id INT (11) not null AUTO_INCREMENT, nom VARCHAR (35) not null , prenom VARCHAR (35) not null , email VARCHAR (70) not null , icq INT (11) null , titre VARCHAR (70) not null , url VARCHAR (255) not null , PRIMARY KEY (id), INDEX (id), UNIQUE (id))
```

Ensuite nous allons utiliser le formulaire ci dessous qui va alimenter la table :

Code HTML

```
<html>
<form method="POST" action="add.php3">
<center>
<input type="text" name="nom" size="20" value="nom" maxlength="35"> <input type="text" name="prenom" size="20" value="prenom" maxlength="35"><br>
<input type="text" name="email" size="20" value="email" maxlength="70"> <input type="text" name="icq" size="20" value="icq" maxlength="11"><br>
<input type="text" name="titre" size="20" value="titre du site" maxlength="70"> <input type="text" name="url" size="20" value="url du site" maxlength="255"><br>
<input type="submit" value="Envoyer" name="envoyer">
</center>
</form>
</html>
```

Donne à l'écran

Top of Form 1

The screenshot shows a web form with the following fields from top to bottom: 'nom', 'prenom', 'email', 'icq', 'titre du site', and 'url du site'. Below these fields is a submit button labeled 'Envoyer'.

Bottom of Form 1

Vous noterez les `maxlength` dans chacun des champs, ceci permet de brider le nombre de caractères maximum que le visiteur peut entrer dans le champ, bien sûr ici le `maxlength` correspond au nombre de caractères spécifié dans la création de la table `infos.tbl`. Cela a un intérêt, celui d'être sûr que le visiteur ne tapera pas plus de caractères que prévu.

Voyons maintenant le script PHP en lui-même, celui-ci sera contenu dans le fichier `add.php3` auquel fait référence le **POST** du formulaire :

Comme vous le savez maintenant la variable de chacun des champs reprend le `name` précédé du signe dollar (`$`), dans notre cas voici les 6 variables :

`$nom` , `$prenom` , `$email` , `$icq` , `$titre` et `$url`.

Code PHP de `add.php3`



```

<?
// On commence par vérifier si les champs sont vides
if(empty($nom) OR empty($prenom) OR empty($email) OR empty($titre) OR empty($url))
{
echo '<font color="red">Attention, seul le champs <b>ICQ</b> peut rester vide!</font>';
}

// Aucun champ n'est vide, on peut enregistrer dans la table
else
{
$db = mysql_connect('localhost', 'login', 'password'); // connexion à la base
mysql_select_db('nom_de_la_base',$db); // sélection de la base

// on ecrit la requete sql
$sql = "INSERT INTO infos_tbl VALUES('','$nom','$prenom','$email','$icq','$titre','$url')";

// on insère les informations du formulaire dans la table
mysql_query($sql) or die('Erreur SQL !.$sql.<br>'.mysql_error());

// on affiche le résultat pour le visiteur
echo 'Vos infos on été ajoutées.';

mysql_close(); // on ferme la connexion
}
?>

```

Explication :

La requête `INSERT INTO` permet donc l'insertion des champs du formulaire dans la table. Dans notre cas le premier champs reste vide car il s'agit de l'id (identifiant) qui s'incrémente automatiquement à chaque nouvelle requête `INSERT`.

Notez que dans le cas ou vous ne voudriez insérer que les champs Nom et Prénom dans la table vous devriez spécifier dans la requête le nom de chaque champs, comme suit :

```
$sql = "INSERT INTO infos_tbl(nom,prenom) VALUES('$nom','$prenom')";
```

Mais attention dans ce cas les autres champs de la table devront avoir l'attribut **NULL** et non pas **NOT NULL**, Null indique au champ qu'il pourra rester vide.

Bien sûr, l'idéal et de tester si l'URL existe dans la table pour éviter les doublons, si celle-ci existe déjà on indique au visiteur qu'il ne peut pas valider son formulaire. J'ai pris l'exemple ici de l'URL mais il va sans dire que vous pouvez vérifier les champs de votre choix, cela dépend des doublons que vous ne souhaitez pas avoir dans votre table. Voici le contenu.

Code PHP de "add.php3" avec la vérification de doublon sur l'URL

```

<?
// On commence par vérifier si les champs sont vides
if(empty($nom) OR empty($prenom) OR empty($email) OR empty($titre) OR empty($url))
{
echo '<font color="red">Attention, seul le champs <b>ICQ</b> peut rester vide!</font>';
}
// Aucun champ n'est vide, on peut enregistrer dans la table
else
{
$db = mysql_connect('localhost', 'login', 'password'); // connexion à la base
mysql_select_db('nom_de_la_base',$db); // sélection de la base

// on regarde si l'url existe déjà
$sql = "SELECT id FROM infos_tbl WHERE url='$url'";
$req = mysql_query($sql) or die('Erreur SQL!'.$sql.'<br>'.mysql_error());

// on compte le nombre de résultat
$res = mysql_numrows($req);

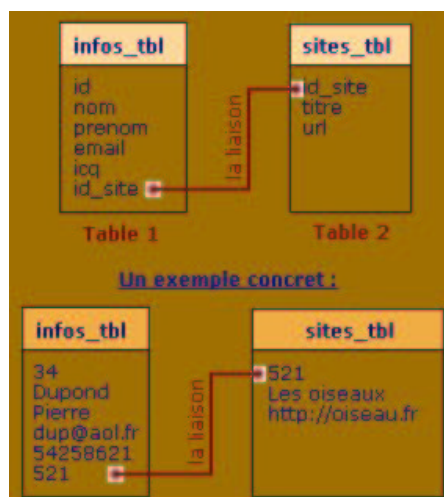
if($res!=0) // l'url existe déjà, on affiche un message d'erreur
{
echo '<font color="red">Désolé, mais cette URL existe déjà dans notre base.</font>';
}
else // L'url n'existe pas, on insère les informations du formulaire dans la table
{
$sql = "INSERT INTO infos_tbl VALUES('$nom','$prenom','$email','$icq','$titre','$url')";
mysql_query($sql) or die('Erreur SQL!'.$sql.'<br>'.mysql_error());

// on affiche le résultat pour le visiteur
echo 'Vos infos on été ajoutées.';
}
mysql_close(); // on ferme la connexion
}
?>

```

5.16.2 Alimenter deux tables et créer une liaison

Voyons maintenant comment ces mêmes informations peuvent être enregistrées dans deux tables différentes en gardant une liaison entre les deux. Le principe est en fait tout bête, admettons que nous ayons une première table qui va nous servir à stocker les coordonnées du visiteur (**\$nom**, **\$prenom**, **\$email**, **\$icq**) et une seconde dans laquelle ne seront sauvegardées que les informations du site (**\$titre** et **\$url**). Voyez ce petit schéma qui va vous éclairer sur la manière de créer une liaison entre 2 tables :



Examinons la méthode à employer :

1. On commence par insérer **\$titre** et **\$url** dans la table 2 (**sites_tbl**).
2. Une fois l'insertion effectuée, on utilise la fonction PHP **mysql_insert_id()** pour connaître l'**id_site** qui a été affecté à notre nouvel enregistrement.
3. On insère le reste du formulaire dans la table 1 (**infos_tbl**), soit : **\$nom** , **\$prenom**, **\$email**, **\$icq** et **\$id_site**.

Nous avons maintenant un liaison entre la table 1 et 2 via l'id_site. Ci-dessous le code PHP de cette manipulation :

Code PHP avec création de la liaison entre les 2 tables

```
<?
$db = mysql_connect('localhost', 'login', 'password'); // connexion à la base
mysql_select_db('nom_de_la_base',$db); // sélection de la base

// on regarde dans la table SITES_TBL si l'url existe déjà
$sql = "SELECT id FROM sites_tbl WHERE url='$url'";
$req = mysql_query($sql) or die('Erreur SQL !'.$sql.'<br>'.mysql_error());
$res= mysql_numrows($req);

if($res!=0) // l'url existe déjà, on affiche un message d'erreur
{
echo '<font color="red">Désolé, mais cette URL existe déjà dans notre base.</font>';
}

else // L'url n'existe pas, on insère d'abord les infos dans SITES_TBL
{
$sql = "INSERT INTO sites_tbl VALUES('$titre','$url')";
mysql_query($sql) or die('Erreur SQL !'.$sql.'<br>'.mysql_error());

// on récupère l'id_site qui vient d'être généré
$id_site = mysql_insert_id();

// ci-dessous on insère les infos dans INFOS_TBL
$sql = "INSERT INTO infos_tbl VALUES('$nom','$prenom','$email','$icq','$id_site')";
mysql_query($sql) or die('Erreur SQL !'.$sql.'<br>'.mysql_error());
}

mysql_close($db); // on ferme la connexion
?>
```

Important : L'id_site de la table 2 (**sites_tbl**) doit être en **auto-incrément automatique** au même titre que l'id de la table 1 (**infos_tbl**). Je ne vous donne "volontairement" pas la requête SQL pour créer les deux tables, et ce pour vous faire travaillé un peu tout de même;).

Conclusion :

Vous pouvez créer autant de liaison que vous le voulez entre vos tables, cela à pour but de les alléger ce qui permet une plus grande rapidité lorsque vous souhaitez les interroger.

Chapitre 6

TP3 : Utilisation de phpMyAdmin

phpMyAdmin est ensemble de scripts PHP qui permet d'administrer une base de données MySQL par l'intermédiaire d'un navigateur WEB, et donc, à distance, à travers Internet. Vous pourrez trouver des informations sur phpMyAdmin à l'adresse suivante <http://phpmyadmin.sourceforge.net/> ou <http://phpwizard.net/projects/phpMyAdmin/>.

phpMyAdmin peut gérer un server MySQL complet (il faut alors être super utilisateur **root**) mais aussi une simple base de données. Pour cela il est nécessaire de configurer convenablement un utilisateur MySQL qui puisse lire et écrire sur la base de données désirées.

phpMyAdmin est prévu pour réaliser les actions suivantes :

- create and drop databases
- create, copy, drop and alter tables
- delete, edit and add fields
- execute any SQL-statement, even batch-queries
- manage keys on fields
- load text files into tables
- create and read dumps of tables
- export and import CSV data
- support single- and multi-user configuration
- communicate in more than 20 different languages

6.1 Connexion à phpMyAdmin

Afin de se connecter pour la gestion de la base de données, nous devons utiliser un navigateur Web. Exécutez pour cela, **mozilla** ou **opera**.

Dans la zone d'adresse, vous devez alors saisir l'adresse suivante : <http://localhost/phpMyAdmin>.

Vous obtenez alors la fenêtre de connexion suivante :

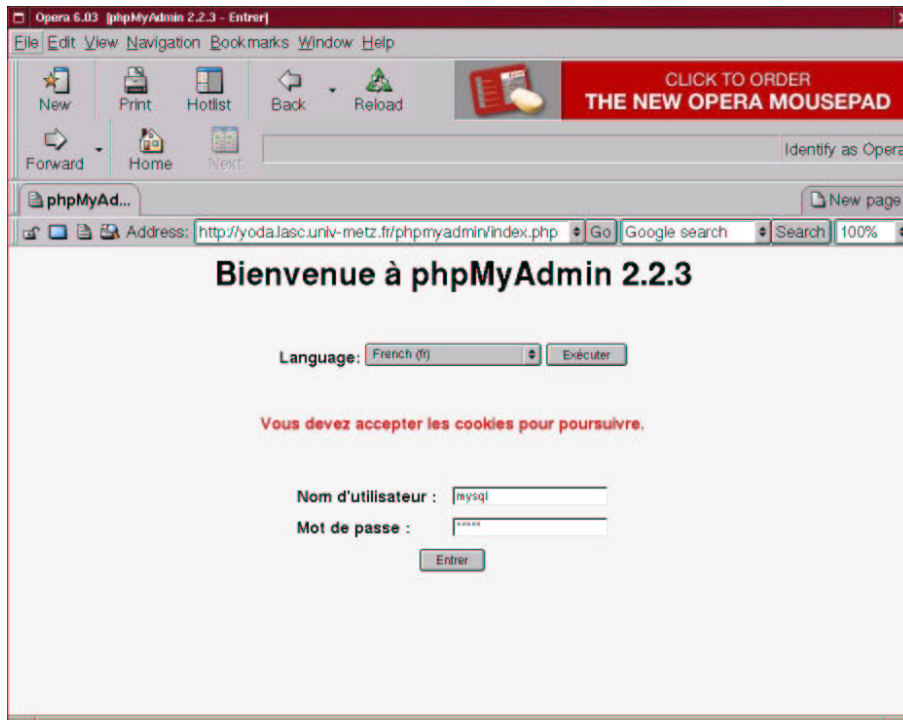


FIG. 6.1 – Interface de connexion à phpMyAdmin version cookie

Entrez le login mysql et le mot de passe mysql. Vous obtenez la fenêtre suivante :

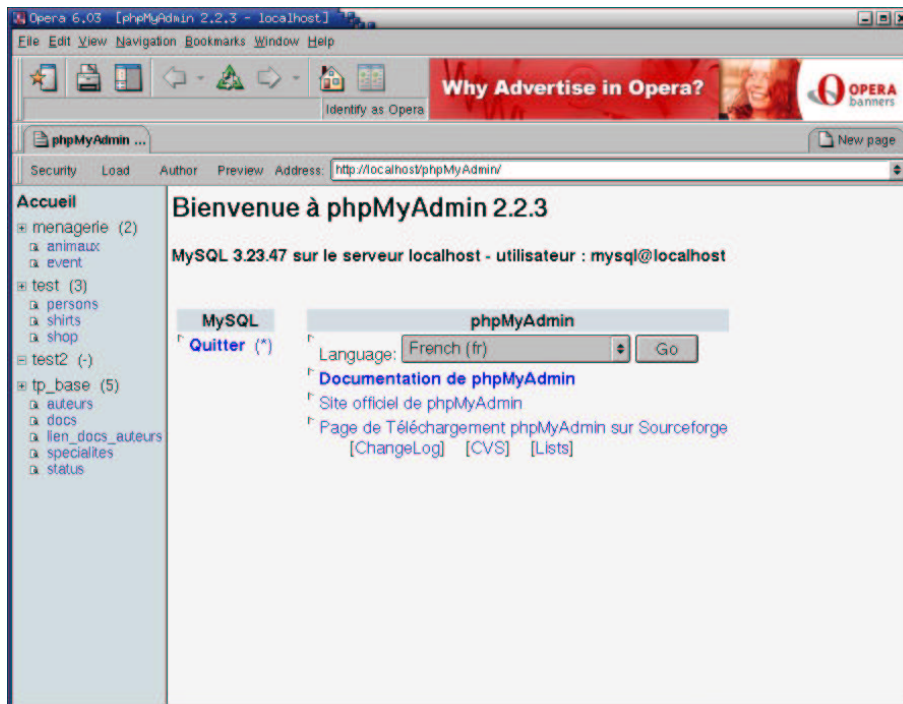


FIG. 6.2 – Interface d'accueil de phpMyAdmin

Vous retrouvez normalement dans la colonne de droite les bases de données que vous avez remplies et/ou vous avez la charge.

L'interface est assez intuitive. Son principale avantage est d'éviter la saisie de nombreuses lignes de commande SQL, pour l'affichage, la création de table, l'insertion de données.

Vous avez sûrement remarqué qu'une des bases est vide de tables. C'est tout à fait normal, elle va nous être utile pour ce troisième TP.

6.2 But du TP

Comme je l'ai écrit précédemment, l'interface est assez intuitive et facile d'utilisation, je ne ferai donc aucun rappel sur son utilisation.

Le but de votre travail est de réaliser la même base que test à l'aide de cette nouvelle interface en utilisant la base test2

Il est évident que ceci peut être fait en quelques secondes à l'aide de la commande appropriée, mais ce n'est pas une copie brute de la base que je vous demande.

Il s'agit de reprendre les étapes de création de la base (création des tables puis insertion des données), à l'aide des fonctionnalités de phpMyAdmin.

Normalement vous n'aurez pas besoin de saisir une seule ligne de SQL.

Ensuite pour la vérification, vous pourrez réaliser différentes requêtes décrites dans le chapitre 1.

La preuve écrite de votre travail sera une sortie papier de la structure et des données de la base, réalisée grâce à l'option **affichage de la structure**.

Cela devrait ressembler à une chose de genre de ce qui suit pour la base **menagerie** :

```
# phpMyAdmin MySQL-Dump
# version 2.2.3
# http://phpwizard.net/phpMyAdmin/
# http://phpmyadmin.sourceforge.net/ (download page)
#
# Serveur: localhost
# Généré le : Vendredi 25 Octobre 2002 à 20:07
# Version du serveur: 3.23.47
# Version de PHP: 4.1.2
# Base de données: 'menagerie'
# -----

#
# Structure de la table 'animaux'
#

DROP TABLE IF EXISTS animaux;
CREATE TABLE animaux (
  nom varchar(20) default NULL,
  propriétaire varchar(20) default NULL,
  espece varchar(20) default NULL,
  genre char(1) default NULL,
  naissance date default NULL,
  mort date default NULL
) TYPE=MyISAM;

#
# Contenu de la table 'animaux'
#

INSERT INTO animaux (nom, propriétaire, espece, genre, naissance, mort)
VALUES ('Fluffy', 'Harold', 'chat', 'f', '1993-02-04', NULL),
('Claws', 'Gwen', 'chat', 'm', '1994-03-17', NULL),
('Buffy', 'Harold', 'chien', 'f', '1989-05-13', NULL),
('Fang', 'Benny', 'chien', 'm', '1990-08-27', NULL),
('Bowser', 'Diane', 'chien', 'm', '1989-08-31', '1995-07-29'),
('Chirpy', 'Gwen', 'oiseau', 'f', '1998-09-11', NULL),
('Whistler', 'Gwen', 'oiseau', NULL, '1997-12-09', NULL),
('Slim', 'Benny', 'serpent', 'm', '1996-04-29', NULL),
('Puffball', 'Diane', 'hamster', 'f', '1999-03-30', NULL);
# -----

#
```

```
# Structure de la table 'event'
#

DROP TABLE IF EXISTS event;
CREATE TABLE event (
  nom varchar(20) default NULL,
  date date default NULL,
  type varchar(15) default NULL,
  remark varchar(255) default NULL
) TYPE=MyISAM;

#
# Contenu de la table 'event'
#

INSERT INTO event (nom, date, type, remark) VALUES ('Fluffy',
  '1995-02-04', 'portée', '4 chatons, 3 femelles, 1 male'),
('Buffy', '1991-05-17', 'portée', '5 chiots, 2 femelles, 3 male'),
('Buffy', '1992-07-13', 'portée', '3 chiots, 3 femelles');
```

Chapitre 7

TP4 : Interrogation d'une base de données via le Web

7.1 Introduction

Le but de ce TP est de réaliser l'interrogation de la base de données `tp_base` par l'intermédiaire d'Internet. Pour cela nous aurons besoin de formulaires et de fichiers `php` qui renverront les réponses aux requêtes passées par le formulaire.

Dans un premier temps une lecture attentive des deux chapitres précédents sera une bonne introduction à l'écriture de formulaire et la manière d'utiliser `php` pour permettre la connexion à la base de données.

Avant toutes choses, il faut créer dans votre répertoire `$HOME` un répertoire `public_html` dans lequel vous placerez les fichiers `html` et `php`.

Ensuite n'oubliez pas de donner les droits `drwxr-xr-x` à `public_html`.

Pour tester votre configuration, créer le fichier `index.html` suivant dans `public_html`.

```
<html>
<head>
  <title>test</title>
</head>
<body>
super
<?
echo "hello";
?>
</body>
</html>
```

Ensuite il est

7.1.1 Connection à la base

Les fonctions Mysql de `php` vont vous permettre de vous connecter à la base de votre choix grâce à un identifiant et un mot de passe. La fonction de connexion retourne un identifiant de connexion (identifiant `mysql` et sont mot de passe) qui est pour nous `mysql` :

```
$dbh=mysql_connect("localhost","mysql","mysql");
```

La fonction `mysql_connect` prend trois arguments en paramètre : l'hôte où réside le serveur (dans notre cas la base de données est située sur le même serveur qu'Apache), un identifiant d'utilisateur Mysql (ici il s'agit de `mysql` qui possède la base `tp_base`) et son mot de passe. La fonction renvoie un paramètre dans `$ dbh` qui est l'identifiant de connexion. La valeur contenue dans `$ dbh` importe peu, sauf si cette dernière est nulle. En effet dans ce cas une erreur s'est produite lors de la connexion à la base.

Cette technique est valable pour des tests, mais si vous envisagez une sérieuse exploitation de la base de données via `php`, il est nécessaire de programmer de telle manière que la mise à jour d'un mot de passe ne demande pas la modification de tous les fichiers faisant référence à la base de données.

Pour cela nous allons placer les paramètres de la base dans un fichier séparé, par exemple `base.php` :

```
<?
$DBdatabase='tp_base';
$DBuser='mysql';
$DBpass='mysql'
?>
```

Dans ce fichier, nous définissons trois variables qui contiendront les données relatives à la connexion à la base : le nom de la base, l'identifiant et le mot de passe.

Ensuite au lieu d'utiliser

```
$dbh=mysql_connect("localhost","mysql","mysql");
```

dans le fichier gérant la connexion, on insère

```
<?
require("base.php");
$dbh=mysql_connect("localhost",$DBuser,$DBpass);
?>
```

La directive `require` permet de charger le contenu d'un fichier `php` dans le fichier courant, à la manière d'un `#include` en C. Dans ce cas, après l'appel au fichier `base.php`, les variables `$DBdatabase`, `$DBuser` et `$DBpass` seront disponibles.

Cette directive peut aussi être utilisée dans le cas où du code `html` se retrouve dupliqué un grand nombre de fois.

7.1.2 Première requête

Ce premier fichier `php` aura pour but l'interrogation et l'affichage de la liste des auteurs contenus dans la base `tp_base`. Voici le code source du fichier final `liste_auteurs.php` :

```
<html>
<body>
<?
require("base.php");
$dbh=mysql_connect("localhost",$DBuser,$DBpass);
if (!$dbh)
{
echo "<font color=#FF0000>ERREUR! Impossible de se connecter à la base
$DBdatabase.</font><br>";
echo "</body></html>";
exit;
}
$res=mysql_db_query("$DBdatabase","select nom, prenom from auteurs order by nom;", $dbh);
$errorno=mysql_errno($dbh);
if ($errorno!=0)
{
$error=mysql_error($dbh);
echo "<font color=#FF0000>$error.</font><br>";
mysql_close($dbh);
echo "</body></html>";
exit;
}

$nbr_ligne=mysql_num_rows($res);
$nbr_champ=mysql_num_fields($res);
echo "<table border=2 cellpadding=2 width=50%>";
for($i=0;$i<$nbr_champ;$i++)
{
printf("<th>%s</th>",mysql_field_name($res,$i));
```

```

}
for($i=0;$i<$nbr_ligne;$i++)
{
$row=mysql_fetch_row($res);
echo "<tr>";
foreach($row as $col)
{
if(!$col) $col="&nbsp;";
echo "<td>$col</td>";
}
echo "</tr>";
}
echo "</table>";
?>
</body>
</html>

```

Quelques explications s'imposent peut être.
Le morceau de code

```

if (!$dbh)
{
echo "<font color=\"#FF0000\">ERREUR! Impossible de se connecter à la base
$DBdatabase.</font><br>";
echo "</body></html>";
exit;
}

```

teste une éventuelle erreur lors de la connexion à la base de données et termine le fichier `html` le cas échéants. Si tout se passe bien, nous passons à la requête proprement dite.

```

$res=mysql_db_query("$DBdatabase","select nom, prenom from auteurs order by nom;", $dbh);
$errorno=mysql_errno($dbh);
if ($errorno!=0)
{
$error=mysql_error($dbh);
echo "<font color=\"#FF0000\">$error.</font><br>";
mysql_close($dbh);
echo "</body></html>";
exit;
}

```

La fonction `mysql_db_query` prend trois paramètres :

- le nom de la base sur laquelle porte la requête,
- la requête elle même,
- et l'identifiant de la connexion.

Elle renvoie dans la variable `$ res` un identifiant ou un objet résultat. Attention cette variable n'est pas affichable directement. Ici encore on teste la présence d'une erreur. Comme il n'est pas possible de tester directement la variable `$ res`, on utilise la fonction `mysql_errno` en lui l'identifiant de la connexion comme paramètre. Cette fonction retourne un numéro d'erreur ou 0 si tout s'est bien passé. Il nous suffit alors de tester la valeur de `$ errorno` et d'afficher le message d'erreur renvoyer par la fonction `mysql_errno` sans oublier de terminer la page `html`.

Le résultat de la requête est une table qu'il convient de traiter. Il faut alors utiliser des fonctions spécifiques. On va tout d'abord se renseigner sur le nombre de lignes et de colonnes contenues dans cette table.

```

$nbr_ligne=mysql_num_rows($res);
$nbr_champ=mysql_num_fields($res);

```

La fonction `mysql_num_rows` renvoie le nombre de lignes de l'identifiant passé en paramètre. La fonction `mysql_num_fields` permet de connaître le nombre de colonnes contenues dans la table résultat.

En suite nous pouvons récupérer les noms des champs de la table grâce à la fonction `mysql_field_name` qui s'utilise de la manière suivante :

```
$nom_champ=mysql_field_name($res,numero_du_champ);
```

Il nous reste à récupérer les valeurs contenues dans les lignes de la tables. Pour cela nous allons utiliser la fonction `mysql_fetch_rows`. Cette dernière permet d'extraire une ligne de résultat via l'identifiant de connexion.

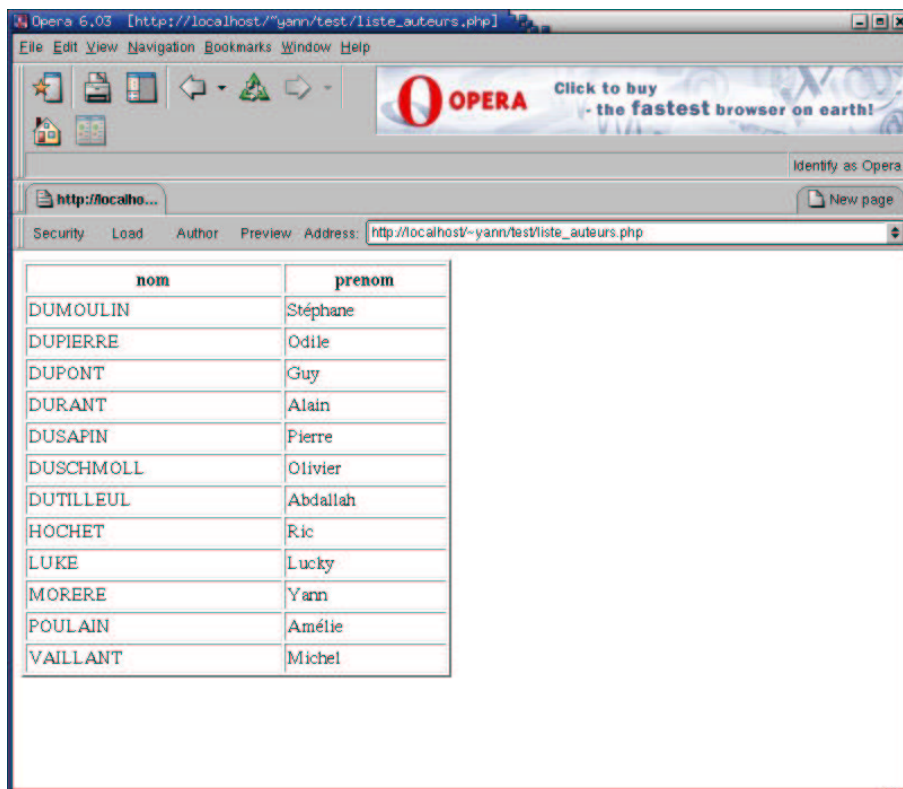
Le code suivant permet de placer le résultat dans des cases de tableau :

```
for($i=0;$i<$nbr_ligne;$i++)
{
$row=mysql_fetch_row($res);
echo "<tr><td>$row[0]</td><td>$row[1]</td></tr>";
}"
```

le code suivant permet de s'adapter à tout type de tableau :

```
for($i=0;$i<$nbr_ligne;$i++)
{
$row=mysql_fetch_row($res);
echo "<tr>";
foreach($row as $col)
{
if(!$col) $col="&nbsp;";
echo "<td>$col</td>";
}
echo "</tr>";
}
```

Tout ceci nous donne le résultat suivant :



The screenshot shows a web browser window with the URL `http://localhost/~yann/test/liste_auteurs.php`. The browser displays a table with two columns: 'nom' and 'prenom'. The table contains the following data:

nom	prenom
DUMOULIN	Stéphane
DUPIERRE	Odile
DUPONT	Guy
DURANT	Alain
DUSAPIN	Pierre
DUSCHMOLL	Olivier
DUTILLEUL	Abdallah
HOCHET	Ric
LUKE	Lucky
MORERE	Yann
POULAIN	Amélie
VAILLANT	Michel

FIG. 7.1 – Notre première requête

Maintenant c'est à vous de jouer.

7.2 Formulaire 1 : Lister les auteurs par nom et prénom

Dans cette première question, il s'agit de réaliser le formulaire ci-dessous qui nous reverra la liste des auteurs (nom et prénom) triés par un des champs de la table auteurs.

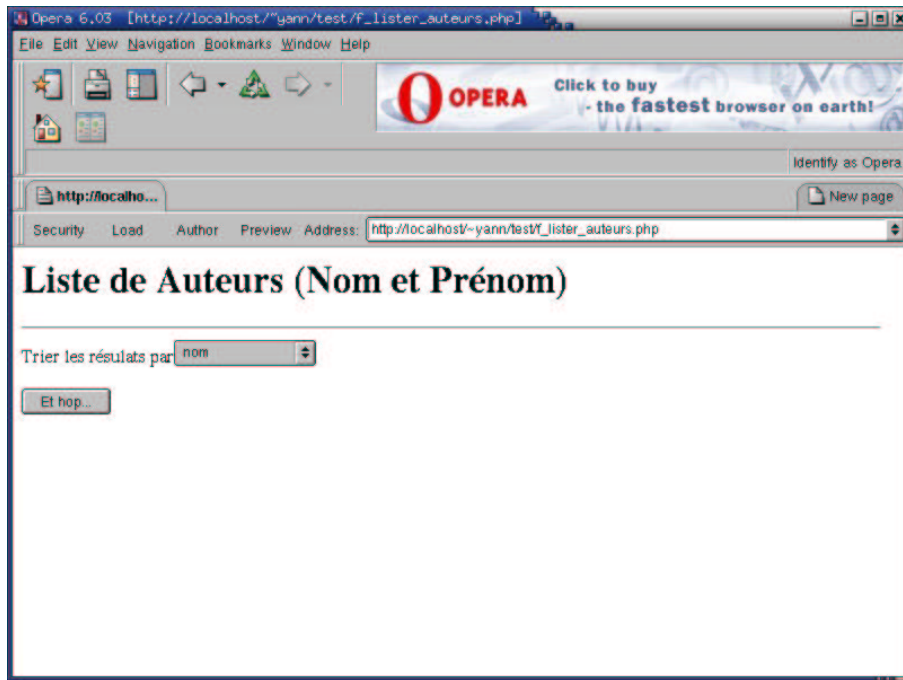


FIG. 7.2 – Notre premier formulaire

Ce qui nous donnera en réponse le résultat suivant :



FIG. 7.3 – La réponse à notre premier formulaire

Vous réaliserez donc deux fichiers : `f_lister_auteurs.php` qui contiendra le code du formulaire, et un fichier `lister_auteurs.php` qui contiendra le code source du traitement de la requête à la base de données, puis la mise en page web de cette réponse.

7.3 Formulaire 2 : Lister les auteurs en sélectionnant les champs à afficher

Dans cette question, il s'agit de réaliser le formulaire ci-dessous qui nous reverra une liste d'information sur les auteurs (que l'on aura au préalable sélectionnée) triée par un des champs de la table **auteurs**.

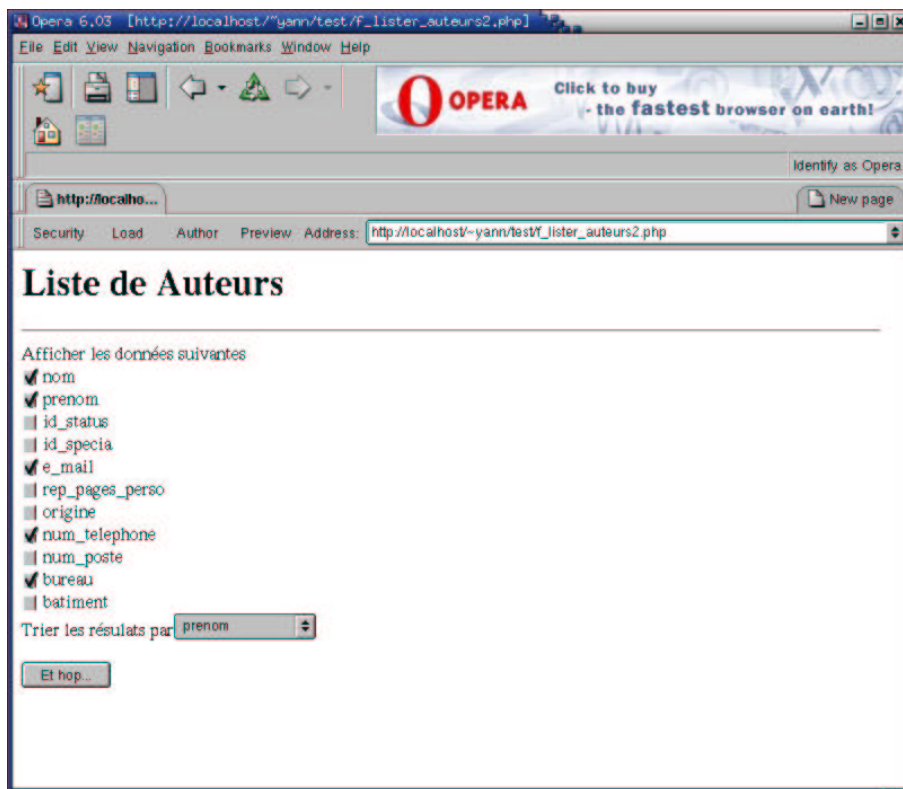


FIG. 7.4 – Notre second formulaire

Ce qui nous donnera en réponse le résultat suivant :



The screenshot shows a web browser window with the title "Opera 6.03 [http://localhost/~yann/test/lister_auteurs2.php]". The address bar shows "http://localho...". The page content is titled "Liste de Auteurs" and displays a table with the following data:

nom	prenom	e_mail	num_telephone	bureau
DUTILLEUL	Abdallah	dutilleul@imaginaire.org	555 555 559	12
DURANT	Alain	durant@imaginaire.org	555 555 556	1
POULAIN	Amélie	poulain@imaginaire.org	555 555 669	4
DUPONT	Guy	dupont@imaginaire.org	555 555 555	2
LUKE	Lucky	luke@imaginaire.org	555 555 668	4
VAILLANT	Michel	vaillant@imaginaire.org	555 555 669	4
DUPIERRE	Odile	dupierre@imaginaire.org	555 555 557	6
DUSCHMOLL	Olivier	duschmoll@imaginaire.org	555 555 553	9
DUSAPIN	Pierre	dusapin@imaginaire.org	555 555 560	11
HOCHET	Ric	hochet@imaginaire.org	555 555 669	4
DUMOULIN	Stéphane	dumoulin@imaginaire.org	555 555 552	8
MORERE	Yann	morere@imaginaire.org	555 555 554	4

FIG. 7.5 – La réponse à notre second formulaire

Vous réaliserez donc deux fichiers : `f_lister_auteurs2.php` qui contiendra le code du formulaire, et un fichier `lister_auteurs2.php` qui contiendra le code source du traitement de la requête à la base de données, puis la mise en page web de cette réponse.

7.4 Formulaire 3 : Ajouter un utilisateur dans la base de données

Dans cette première question, il s'agit de réaliser le formulaire ci-dessous qui nous reverra la liste des auteurs (nom et prénom) triée par un des champs de la table `auteurs`.

The screenshot shows a web browser window with the address bar displaying 'http://localhost/~yann/test/f_ajout_auteur.php'. The page title is 'Formulaire d'ajout d'auteurs'. The form contains the following fields:

Nom :	Tournesol
Prénom :	Tryphon
Statuts :	Professeur
Spécialité :	Mécanique
E-mail :	tournesol@tintin.be
Site Web :	http://www.professeur-tournesol.be
Origine :	Inconnue
Téléphone :	555 666 7775
Poste :	7775
Bureau :	420
Batiment :	Moulinart

Below the form is an 'Ajouter' button.

FIG. 7.6 – Formulaire d'ajout d'auteur

Ce qui nous donnera en réponse le résultat suivant :

The screenshot shows the same web browser window, but the address bar now displays 'http://localhost/~yann/test/ajouter_auteur.php'. The page content is a confirmation message: 'L'auteur Tournesol Tryphon a été ajouté convenablement'. Below the message is a blue link labeled 'retour au formulaire'.

FIG. 7.7 – La réponse à notre formulaire d'ajout

Vous réaliserez donc deux fichiers : `f_ajout_auteurs.php` qui contiendra le code du formulaire, et un fichier

`ajouter_auteurs.php` qui contiendra le code source du traitement de la requête à la base de données, puis la mise en page web de cette réponse.

Dans le cas où les champs `nom` et `prénom` seraient vides lors de l'envoi du formulaire, vous devrez afficher un message d'erreur comme dans la figure suivante :

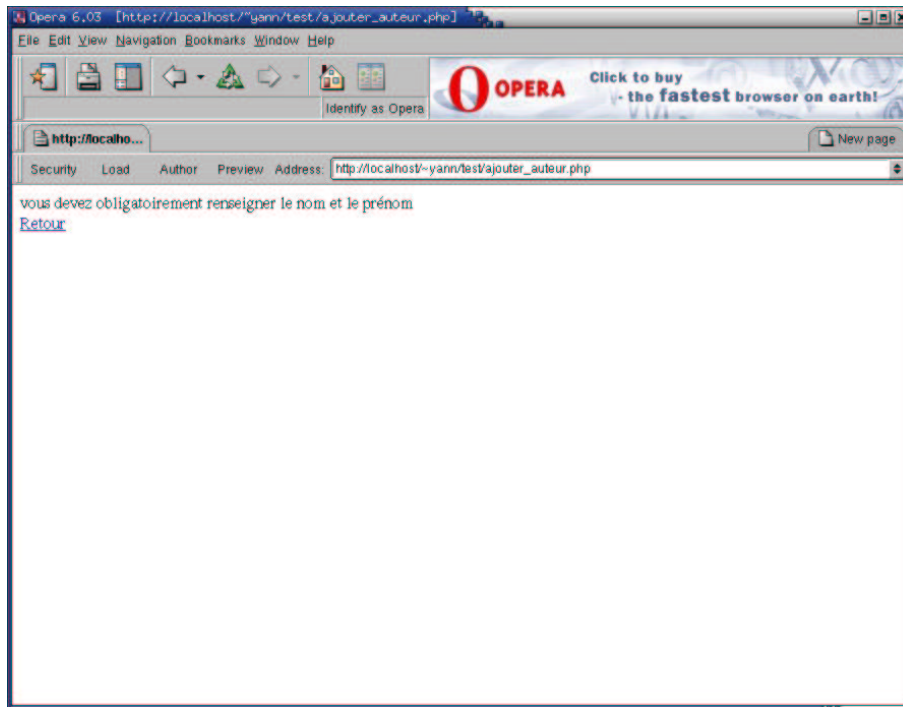


FIG. 7.8 – La réponse d'erreur de notre formulaire d'ajout

7.5 Formulaire 4 : Requête Bibliographique

Dans cette dernière question vous devrez réaliser un fichier `php` qui réalise une requête complexe et qui affiche le résultat de cette requête sous une forme bibliographique comme montré ci-dessous.

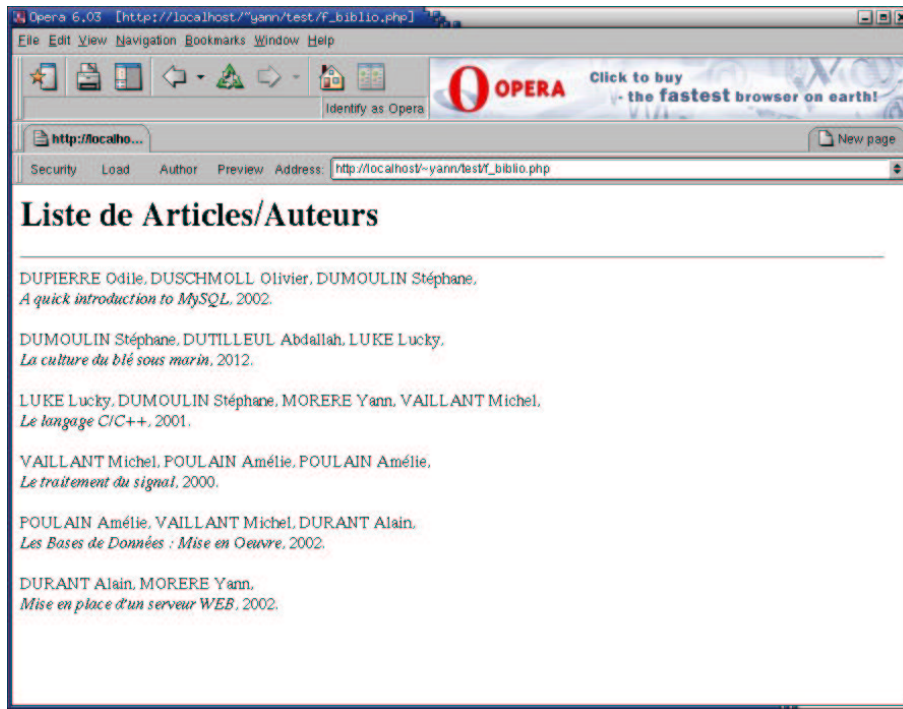


FIG. 7.9 – La réponse à notre requête de bibliographie

Il s'agit donc de lister tous les articles, avec les auteurs qui y ont participé en respectant l'ordre des auteurs (champ ordre de la table liens_doc_auteur).