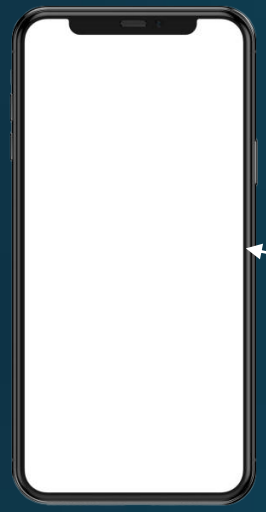
Le SQL


ADRAR
DIGIT@L ACADEMY
PÔLE NUMERIQUE DU CENTRE DE FORMATION ADRAR
> SUPPORT, ADMINISTRATION SYSTEMES & RESEAUX
> DEVELOPPEMENT D'APPLICATIONS WEB & MOBILES
> TRANSFORMATION NUMERIQUE DES ENTREPRISES
http://www.adrar-numerique.com

Le SQL



Le SQL

Pré-requis :
Pour la suite de votre formation et de ce cours, vous allez avoir besoin de certains outils installés sur votre machine.

Installation de Wamp:

- Version x64 pour windows 64 bits : https://sourceforge.net/projects/wampserver/files/latest/download
- Version x86 pour windows 32 bits : https://sourceforge.net/projects/wampserver/files/latest/download

Installation de Worbench:

Commençons par télécharger le logiciel MySQL Community :

- Version x64 pour windows 64 bits : https://dev.mysql.com/downloads/installer/

Définition d'une base données.

SQL veut dire langage de requête structurée. (en anglais : Structured Query Language).

Mais afin d'aller plus loin sur ce cours revenons sur les notions de bases:

- Les bases de données,
- Les SGBDR.

Définition d'une base de données :

Notions de bases:

- Champ
  Un champ est la zone qui permet le stockage des données, correspondant à une colonne dans une vue en liste.
- Enregistrement
  Un enregistrement est un ensemble de valeurs correspondant à une ligne toujours dans une vue en liste.
- Table
  Une table va contenir l'ensemble des enregistrements .
- Index
  Un index est le conteneur des clés.

Le SQL

id	nom	prénom	profession	code postal	ville
1	Durand	Michel	Directeur	75016	Paris
2	Dupond	Karine	Secrétaire	92000	Courbevoie
3	Mensoif	Gérard	Commercial	75001	Paris
4	Monauto	Alphonse	Commercial	75002	Paris
5	Emarre	Jean	Employé	75015	Paris
6	Abois	Nicole	Secrétaire	95000	St Denis
7	Dupond	Antoine	Assistant commercial	75014	Paris

intitulé du champ

enregistrement

Champ

Table

Le SQL

- Une base de données est un ensemble qui permet le stockage des données.
- Les données sont écrites de manière structurées ce qui signifie que chaque donnée est enregistrée dans un champ qui est inclus dans une table.
- Lorsque les tables ont des relations entre elles, on parle alors de bases de données relationnelles.
- Ces tables peuvent être indexées pour permettre des accès plus rapides à certaines données et permettre aussi des tris.

Le SGBR ou SGBDR

Le SGBDR va gérer l'accès à la base de données. Aucun logiciel n'accédera directement à la base de données, seul le SGBD le fera.

Cela implique que le SGBDR :

- disposera d'un langage pour pouvoir dialoguer avec les applications, (notamment le SQL).
- gérera l'écriture et la lecture des données,
- mettra à disposition les tables, et les droits associés,
- gérera le partage des données,
- s'assurera de l'intégrité des données,
- gérera la relation entre les tables (dans le cas de base de données relationnelles)

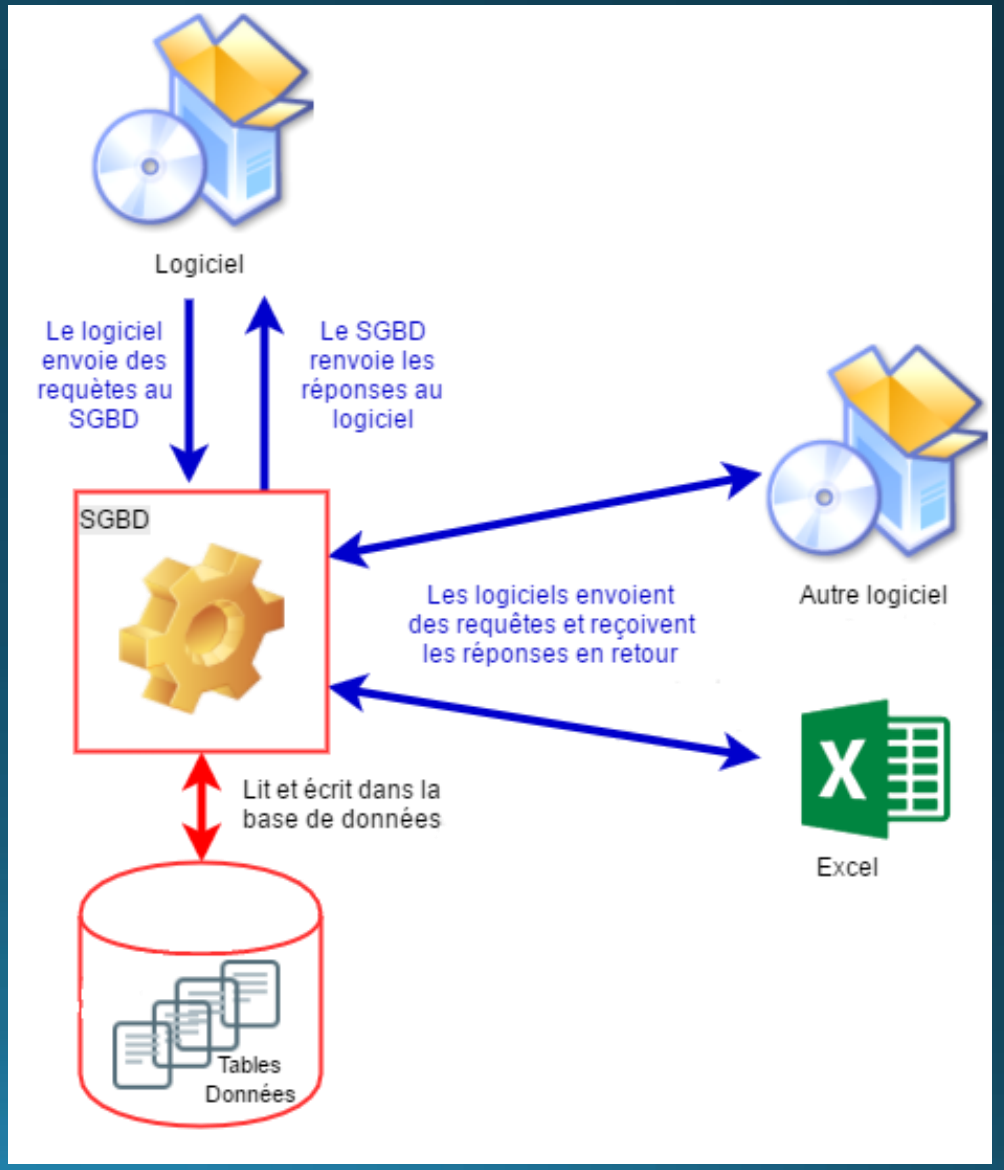
Le SQL

Les avantages à utiliser les bases de données sont les suivants :

Une standardisation des accès :
Chaque logiciel accède aux données via le SGBD en utilisant un langage standardisé. Il est possible de changer de base de données sans avoir un réécrire le logiciel.

Un partage des données ainsi que l'accès simultané :
Le SGBD gère les accès à la base de données, donc il gère aussi le partage au données. Il est donc possible d'avoir plusieurs logiciels qui lisent et écrivent en même temps sur la base de données.

Une grande fiabilité des données :
Chaque logiciel ne peut pas faire n'importe quoi. C'est le SGBD qui gère l'enregistrement des données.



Les principaux SGBDR utilisés en entreprise sont:

- Oracle : sans doute le plus connu,
- MySQL : Un système gratuit issu du monde libre,
- SQLServer : Le SGBD de Microsoft.

Bon à savoir : vous pourrez trouver les appellations de base de données sous les formes suivantes :

- BdD: Base de Données
- BD: Base Données
- DB: DataBase (nom en anglais)

Le SQL

Afin que les logiciels utilisés et le SGBD puissent se comprendre, ils utilisent un langage appelé SQL. Ce langage complet va être utilisé pour:

- Lire les données,
- Ecrire les données,
- Modifier les données,
- Supprimer les données

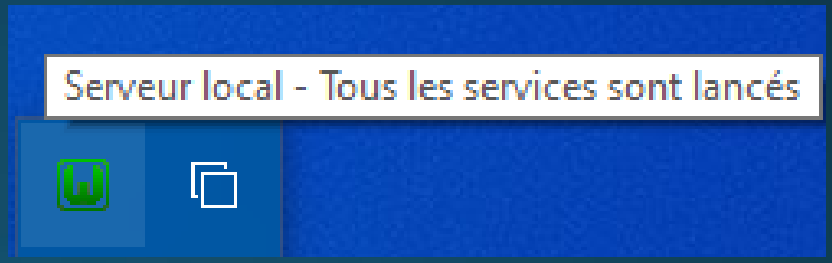
Il permettra aussi de modifier la structure de la base de données :

- Ajouter des tables,
- Modifier les tables,
- les supprimer
- Ajouter, ou supprimer des utilisateurs,
- Gérer les droits des utilisateurs,
- Gérer les bases de données : en créer de nouvelles, les modifier, etc …

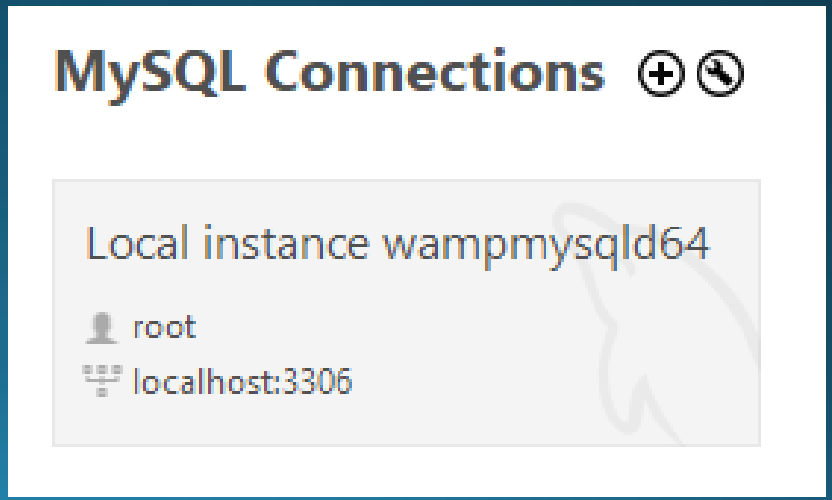
Démarrer son serveur local:

Lancez votre serveur wamp :

Vérifiez que votre serveur est bien lancé :



Lancez maintenant MySqlWorkbench puis cliquez sur votre instance locale :



Les requêtes sur la structure.

La requête **CREATE:**

Cette requête servira principalement à créer vos tables dans votre BDD, mais également à créer votre base de données. Mais étudions ensemble l'exemple ci-contre:

```
CREATE DATABASE test;
use test;
CREATE TABLE `users` (
  idUser bigint(20) NOT NULL,
  name varchar(50) NOT NULL,
  password varchar(255) NOT NULL,
  mail varchar(100) NOT NULL,
  age int(20) DEFAULT NULL,
  idTown bigint(20) DEFAULT NULL
) ;
```

Le SQL

En première ligne, nous avons la requête « create database test; » qui comme son nom l'indique, permet de créer une base de données nommée « test ».

En seconde ligne, nous avons la requete « use test ; » qui nous permet de spécifier quelle base de données utiliser lors des prochaines requêtes.

```
CREATE DATABASE test;
use test;
CREATE TABLE `users` (
  idUser bigint(20) NOT NULL,
  name varchar(50) NOT NULL,
  password varchar(255) NOT NULL,
  mail varchar(100) NOT NULL,
  age int(20) DEFAULT NULL,
  idTown bigint(20) DEFAULT NULL
) ;
```

Le SQL

La ligne « CREATE TABLE `users`» permet de créer une table « users » avec à la suite, la liste des colonnes ou des champs qui serons attendus.

ATTENTION: le mot « users » est entouré de « backquote », des apostrophe inversées.
Pour faire des backquote, il faut faire « alt gr + 7 ».

```
CREATE DATABASE test;
use test;
CREATE TABLE `users` (
  idUser bigint(20) NOT NULL,
  name varchar(50) NOT NULL,
  password varchar(255) NOT NULL,
  mail varchar(100) NOT NULL,
  age int(20) DEFAULT NULL,
  idTown bigint(20) DEFAULT NULL
) ;
```

Le SQL

'idUser' sera le nom du 1er champ de notre table, et à partir de là nous pouvons donc en conclure que 'name', 'password', 'mail', 'age', 'idtown' seront les noms de nos autres champs.

Cependant, le champ 'idtown' correspond également à la clé primaire d'une autre table. Nous l'aborderons à nouveau plus tard.

```
CREATE DATABASE test;
use test;
CREATE TABLE `users` (
  idUser bigint(20) NOT NULL,
  name varchar(50) NOT NULL,
  password varchar(255) NOT NULL,
  mail varchar(100) NOT NULL,
  age int(20) DEFAULT NULL,
  idTown bigint(20) DEFAULT NULL
) ;
```

‘bigint’, ‘varchar’, ‘int’ correspondent au type du champ il peut être numérique (int , double, real, float), une chaine de caractère (char, varchar, text, blob, enum, set), une date(date, datetime, timestamp), spatial (polygon, point, geometry) ou un JSON.

(20), (50), (100), etc … indiquent la taille ou la valeur maximale du champ (non obligatoire pour certains types, comme le ‘text’ par exemple).

```
CREATE DATABASE test;
use test;
CREATE TABLE `users` (
  idUser bigint(20) NOT NULL,
  name varchar(50) NOT NULL,
  password varchar(255) NOT NULL,
  mail varchar(100) NOT NULL,
  age int(20) DEFAULT NULL,
  idTown bigint(20) DEFAULT NULL
) ;
```

‘NOT NULL’, ‘DEFAULT NULL’, signalent à notre SGBD les valeurs par défaut de nos divers champs lors d’un enregistrement. Cela n’est pas obligatoire on aurait très bien pu écrire « password varchar(255), » sans conséquences pour notre requête. Cependant, si nous indiquons NOT NULL, le SGBD attendra de recevoir une valeur lors de la demande d’insertion d’un nouvel enregistrement, faute de quoi il nous retournera une erreur. Au contraire, Le DEFAULT NULL demande de remplir un champ avec une valeur null au cas ou il ne reçoit pas de valeur pour ce champ.

```sql
CREATE DATABASE test;
use test;
CREATE TABLE `users` (
  idUser bigint(20) NOT NULL,
  name varchar(50) NOT NULL,
  password varchar(255) NOT NULL,
  mail varchar(100) NOT NULL,
  age int(20) DEFAULT NULL,
  idTown bigint(20) DEFAULT NULL
) ;
```

La requête ALTER TABLE:

Cette requête vous permettra de modifier les tables dans votre BDD.
Mais afin de voir cela ensemble, vous aller reprendre la partie précédente en créant votre BDD 'test' sur votre propre serveur, ainsi que la table 'users' et la table 'towns' ayant les champs :

- idTown dont le type est un entier,
- townName dont le type est un varchar,
- townCp dont le type est un varchar.

```
CREATE TABLE towns (
   idTown bigint(20),
   townname varchar(100),
   towncp varchar(50)
);
```

Maintenant que nos deux tables sont créées, nous allons pouvoir les modifier.

```
ALTER TABLE users
   ADD PRIMARY KEY (idUser),
   ADD KEY idTown (idTown);
```

Cette requête demande à notre SGBD de modifier notre table « users ».
Lors de celle-ci on lui indique d'ajouter des clés (KEY) qui serviront d'index grâce au terme 'ADD' .
PRIMARY KEY (idUser) signifie clé primaire (PK). Cette dernière est unique et est attribuée au champ idUser de la table. Elle nous permettra d'utiliser l'enregistrement lui correspondant.
KEY idTown (idTown) est une clé index. Elle provient du champ idTown de la table (towns) et sera associée au champ idTown de la table users.

```
ALTER TABLE users
   ADD CONSTRAINT FOREIGN KEY (idtown) REFERENCES towns (idtown);
```

Ici, nous ajoutons une contrainte sur notre table ‘users ’. En effet, nous demandons à notre SGBD d’ajouter une contrainte avec le terme ADD CONSTRAINT .
Cette contrainte sera une FOREIGN KEY (FK) ‘ou clé étrangère en français’.
Cette FK correspondra au champ ‘idTown’ de la table ‘users’ et, comme nous l’indique REFERENCES, elle fera donc références à la table ‘towns’ et précisément à son champ ‘idTown’.

```
ALTER TABLE users
   MODIFY iduser bigint(20) NOT NULL AUTO_INCREMENT;
```

Sur cette requête nous voulons modifier ‘MODIFY’ notre champ idUser, qui est également une PK, en indiquant à notre SGBD que ce champ ne peut pas être null ‘NOT NULL’, mais également qu’il doit s’auto-incrémenter ‘AUTO_INCREMENT’ à chaque nouvel enregistrement.

Bon à savoir : pour renommer une table la requête est la suivante = « RENAME TABLE ‘nom_à_changer’ TO ‘nouveau_nom’ ».

Le SQL

```
ALTER TABLE users
    add column phone varchar(50);
```

Cette fois nous demandons d'ajouter une colonne (ou champ) 'add column' qui sera nommé 'phone' et de type varchar avec une taille de 50.

Je m'aperçois que le nom ou le type de la colonne ne vont pas convenir donc je vais les modifier :

```
ALTER TABLE users
    change phone telephone bigint(100);
```

Dans cet exemple j'ai changé le nom de la colonne et son type. Pour y parvenir vous constaterez que j'indique d'abord le nom à changé puis le nom souhaité et enfin le type du champ.

```
ALTER TABLE users
    drop phone ;
```

```
ALTER TABLE users
    drop column phone;
```

Ces 2 exemples font la même chose. Ils servent à supprimer une colonne d'une table.

```
ALTER TABLE users
    modify telephone varchar(100);
```

Ici, nous modifions notre colonne notamment son type. Nous pouvons garder le même type et changer sa taille également.

La requête DROP TABLE :

Cette requête est assez simple à écrire et à comprendre. En effet elle sert à supprimer une table de notre BDD.

```
drop TABLE roleBean;
```

Le SQL

Pour conclure sur cette partie, je vous invite d'abord de mettre à jour votre BDD en effectuant les requêtes nécessaires.
Et voici un exemple de création de table reprenant ce que nous venons de voir en condensant un peu l'écriture.

```
CREATE TABLE users (
  idUser bigint(20) NOT NULL primary key auto_increment,
  idsession varchar(255),
  name varchar(50),
  password varchar(255),
  mail varchar(100),
  phone varchar(50),
  idtown bigint(20),
  FOREIGN KEY (idtown) REFERENCES townBean (idtown)
);
```

Maintenant petit exercice

Dans une Database ‘facturation’, j’ai trois tables :

- Clients
- Factures
- Villes

La table clients aura 5 champs dont :

- idClient
- nom
- prenom
- telephone
- idVille

La table factures comportera 3 champs :

- idFacture
- numeroFacture
- idClient

La table villes intégrera 3 champs :

- idVille
- nom
- codePostal