# I.1 / Base de données

## a) Lexique

Une base de données informatique est un ensemble de données stockées sur un support, elle est organisées et structurées de manière à faciliter la manipulation des données.

## b) Exemple

Un site de vente possédant une liste de produit en évolution constante va utiliser une base de données pour faciliter la modification de son catalogue, de ses prix, de son stock et autres.

# I.1 / MySQL

## a) SGBD(R) ou (R)DBMS

Un Système de Gestion de Base de Données (Relationnel) est un logiciel permettant de manipuler les données d'une base de données.

Il existe différent SGBD, chacun ayant des avantages ou des inconvénients par rapport à leurs concurrents, ils disposent également de leur propre lexique.

Tous n'implémente pas le modèle relationnel et il n'est pas nécessaire de le maîtriser pour être capable d'utiliser convenablement un SGBDR.

MySQL est un logiciel Open Source, il est le deuxième SGBDR le plus utilisé.

## b) SQL

Le SQL (Structured Query Language) est un langage informatique utilisé par MySQL pour manipuler les données et interagir avec le SGBDR en se servant de requête.

# I.1 / Le modèle relationnel

## a) Description simplifié

Une relation est un tableau contenant des données.

Un tableau dans lequel une ligne représente un objet, et où chaque ligne représente des objets de même nature.

Chaque colonne est un attribut de l'objet, et contient une valeur qui le définit.

## b) Exemple

| ville | pays | superficie | population | densité |
|---|---|---|---|---|
| Paris | France | 105, 40 km² | 2 206 488 hab. | 20 934 hab./km² |
| New York | États-Unis | 1 214,4 km² | 8 537 673 hab. | 7 030 hab./km² |
| Tokyo | Japon | 2 190,93 km² | 13 622 267 hab. | 6 218 hab./km² |

# I.1 / Les outils & ressources

## a) Liste

- MySQL Community Server (Version 5.7.21)
- MySQL Workbench (Version 6.3.10)

## b) Où se procurer les outils ?

https://www.mysql.com

→ Onglet Téléchargements

→ MySQL Community Edition (GPL)

## c) Google Drive

https://drive.google.com/drive/folders/1DnJn6JNIhDd9EW-Bi-A9q64qcfWaTmEB?usp=sharing

→ studio@angamara.fr

# I.2 / Syntaxe

## a) Mots clés

Les mots clés associés à SQL s'écrivent en MAJUSCULES.

Exemple : CREATE | DROP | ALTER

## b) Tables & Attributs

Les noms des tables ou des attributs s'écrivent en minuscule et sont encadrés par des accents graves.

Si le nom comporte plusieurs mots, utilisez un ou plusieurs underscores.

Exemple : `table` | `attribut_trop_long`

## c) Arguments

Les argument s'écrivent de la manière désirés et sont encadrés par des apostrophes simples. Les apostrophes ne sont pas nécessaire dans le cas de valeur numérique.

Exemple : COMMENT 'Un commentaire !' | DEFAULT 'Sans valeur' | DEFAULT -10

# I.2 / Syntaxe

## d) [...]

Élément de syntaxe facultatif. Ne tapez pas les crochets.

Exemple : [IF NOT EXISTS]

## e) {...}

Élément de syntaxe obligatoire. Ne tapez pas les accolades.

Exemple : {DATABASE | SCHEMA}

## f) | (barre vertical)

Sépare les éléments de syntaxe placés entre crochets ou entre accolades. Vous ne pouvez utiliser qu'un seul de ces éléments.

Exemple : [NOT NULL | NULL]

## g) ;

Terminateur de requête.

Exemple : SHOW DATABASES;

# I.2 / Créer une base de données

## a) Syntaxe

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] `db_nom`;
```

SCHEMA est un alias de DATABASE ; l'un ou l'autre fourni le même résultat.

Si « IF NOT EXISTS » n'est pas présent dans la requête, cela peut déclencher une erreur si une base de données de ce nom existe déjà.

Avec MySQL il n'est pas possible de renommer une base de données après sa création.

## b) Exemple

```
CREATE DATABASE IF NOT EXISTS `ma_db`;
```

Cette requête créera une base de données nommée « ma_db » si elle n'existe pas.

# I.2 / Utiliser une base de données

## a) Syntaxe

```
USE `db_nom`;
```

## b) Exemple

```
USE `ma_db`;
```

Cette requête utilisera la base nommée « ma_db » pour exécuter les prochaines requêtes.

# I.2 / Supprimer une base de données

## a) Syntaxe

```
DROP {DATABASE | SCHEMA} [IF EXISTS] `db_nom`;
```

SCHEMA est un alias de DATABASE ; l'un ou l'autre fourni le même résultat.

Si « IF EXISTS » n'est pas présent dans la requête, cela peut déclencher une erreur si aucune base de données de ce nom existe.

## b) Exemple

```
DROP DATABASE IF EXISTS `ma_db`;
```

Cette requête supprimera la base de données nommée « ma_db » ainsi que tout son contenu si elle existe.

# I.2 / Créer une table

## a) Syntaxe

```
CREATE TABLE [IF NOT EXISTS] `tbl_nom` (
    `col_nom` data_type* [NOT NULL | NULL] [DEFAULT 'Valeur']
    [AUTO_INCREMENT] [COMMENT 'Valeur'],
    primary_key*,
    unique_key*,
    index_key*,
    constraint*
)
COMMENT = 'Valeur';
```

Si « IF NOT EXISTS » n'est pas présent dans la requête, cela peut déclencher une erreur si une table de ce nom existe déjà.

cas_specifique* : Voir plus loin dans le WorkShop

## b) NOT NULL | NULL

Définit si un attribut doit-être rempli ou non à l'insertion.

# I.2 / Créer une table

## c) DEFAULT [‘Valeur’ | 0]

Définit la valeur par défaut à l’insertion, dans le cas où l’attribut est laisser vide.

## d) AUTO_INCREMENT

Utiliser sur les types de données numériques, dans le cas où l’attribut est laisser vide, la valeur de 1 sera ajouté par rapport au précédant enregistrement.

## e) COMMENT [=] ‘Valeur’

Permet de spécifier un commentaire à l’attribut ou à la table.

Le signe égal est nécessaire quand il est question d’une table.

# I.2 / Les types de données

## a) Préambule

Il est important de comprendre chaque type de données afin de choisir le type le plus adéquat de votre attribut. Un mauvais type pourrait avoir plusieurs conséquences :

- Gaspillage mémoire (Ex : stocker de petites données dans un type prévu pour des données plus importantes.)

- Problème de performance (Ex : rechercher une donnée numérique avec un type alphanumérique.)

# I.2 / Les types de données (data_type*)

## b) Types numériques

Il y a deux sous-catégories : les entiers et les décimaux.

Les entiers sont désignés par le mot clé INT et ses déclinaisons TINYINT, SMALLINT, MEDIUMINT, BIGINT. La différence est le nombre d'octets réservés à la valeur.

| Type | Octets | Minimum | Maximum |
|---|---|---|---|
| TINYINT | 1 | -128 | 127 |
| SMALLINT | 2 | -32 768 | 32 767 |
| MEDIUMINT | 3 | -8 388 608 | 8 388 607 |
| INT | 4 | -2 147 483 648 | 2 147 483 647 |
| BIGINT | 8 | -9 223 372 036 854 775 808 | 9 223 372 036 854 775 807 |

Ex : Si vous essayer de stocker 1238 dans un type TINYINT, la valeur sauvegarder sera 127.

# I.2 / Les types de données (data_type*)

Le mot clé UNSIGNED peut être préciser pour définir une valeur positive ; l'intervalle entre la valeur minimum et maximum ne change pas, mais la valeur minimum est décalé à 0.

Ex : Pour TINYINT UNSIGNED, on peut stocker une valeur allant de 0 à 255.

Il y a 5 mots clés permettant de stocker des décimaux : DECIMAL, NUMERIC, FLOAT, REAL et DOUBLE.

NUMERIC et DECIMAL sont identiques, ils acceptent 2 paramètres facultatifs : la précision et l'échelle.

- La précision est le nombre de chiffres significatifs stockés. (Les 0 à gauche ne compte pas ; 0042 est équivalent à 42.)

- L'échelle définit le nombre de chiffre après la virgule.

Ex : DECIMAL(4, 2) permet de stocker 4 chiffres significatifs, 2 chiffre de chaque côté de la virgule.

-78.96, 12, 89.2, -45.

-112.89 deviendra -99.99 et 785.785 deviendra 99,99.

# I.2 / Les types de données (data_type*)

FLOAT, REAL et DOUBLE fonctionnent avec les mêmes paramètres que DECIMAL et NUMERIC.

FLOAT utilise 4 octets de stockage. REAL et DOUBLE sont équivalent et utilisent 8 octets. Pour DECIMAL et NUMERIC cela dépend du nombre de chiffres significatifs stockés.

# I.2 / Les types de données (data_type*)

## c) Les types alphanumériques

CHAR et VARCHAR sont utilisés pour stocker des valeur allant de 0 à 255 caractères. Ils s'utilisent avec un paramètre obligatoire : la longueur.

- La longueur est le nombre de caractères autorisées.

Ex : VARCHAR(6) permet de stocker un maximum de 6 caractères.

'abcdef', '123456'.

'CodingFactory' deviendra 'Coding'.

La différence entre CHAR et VARCHAR est la manière dont les données sont stockés en mémoire. CHAR(x) stockera toujours x caractères en complétant si nécessaire avec des espaces.

| Texte | CHAR(4) | Mémoire | VARCHAR(4) | Mémoire |
|---|---|---|---|---|
| '' | '  ' | 4 octets | '' | 1 octet |
| 'Co' | 'Co ' | 4 octets | 'Co' | 3 octets |
| 'Code' | 'Code' | 4 octets | 'Code' | 5 octets |
| 'Coding' | 'Codi' | 4 octets | 'Codi' | 5 octets |

# I.2 / Les types de données (data_type*)

Le type TEXT et ses déclinaisons TINYTEXT, MEDIUMTEXT, LONGTEXT permettent de stockées des textes beaucoup plus grand. La différence est le nombre d'octets réservés à la valeur.

| Type | Octets | Longueur |
|---|---|---|
| TINYTEXT | L+1 | 255 caractères |
| TEXT | L+2 | 65 535 caractères |
| MEDIUMTEXT | L+3 | 16 777 215 caractères |
| LONGTEXT | L+4 | 4 294 967 295 caractères |

Le type TEXT et ses déclinaisons sont insensible à la casse contrairement à CHAR et VARCHAR.

La casse est la différence entre 'E' et 'e'.

# I.2 / Les types de données (data_type*)

## d) Les données temporels

Il y a 5 mots clés permettant de stocker le temps : DATE, DATETIME, TIME, TIMESTAMP et YEAR.

| Type | Octets | Format | Interval |
|---|---|---|---|
| DATE | 3 | AAAA-MM-JJ | 1000-01-01 à 9999-12-31 |
| DATETIME | 8 | AAAA-MM-JJ HH:MM:SS | 1000-01-01 00:00:00 à 9999-12-31 23:59:59 |
| TIME | 3 | HH:MM:SS | -838:59:59 à 838:59:59 |
| TIMESTAMP | 4 | AAAAMMJJHHMMSS<br>AAMMJJHHMMSS<br>AAAAMMJJ<br>AAMMJJ | 1970-01-01 00:00:00 à 2038-01-19 03:14:07 |
| YEAR | 1 | YYYY<br>YY | 1901 à 2155<br>1970 à 2069 |

TIMESTAMP s'utilise avec le paramètre facultatif longueur, qui doit être égal à 14, 12, 8 ou 6. (Ex : TIMESTAMP(8))

Le mot clé CURRENT_TIMESTAMP peut-être utilisé comme valeur par défaut avec le type TIMESTAMP ou DATETIME pour définir la date actuelle.

Ex : DATETIME DEFAULT CURRENT_TIMESTAMP

# I.2 / La clé primaire (primary_key*)

## a) Qu'est ce que c'est ?

La clé primaire d'une table permet d'identifier de manière unique une ligne, cette clé peut-être composée d'un ou de plusieurs attributs.

Si aucuns attributs ne permet de différencier une ligne d'une autre, une clé artificielle sera utilisée.

| id | prenom | nom |
|---|---|---|
| 0 | George | Lucas |
| 1 | George | Clooney |
| 2 | Louis-Marie | Lucas |

Sur cet exemple, les attributs `prenom` et `nom` ne permettent pas d'identifier avec certitude une ligne d'une autre. `id` est donc une clé artificielle primaire.

## b) Syntaxe

```
PRIMARY KEY (`col_nom`)
```

Ex : `PRIMARY KEY (`id`)`

# I.2 / Une clé unique (unique_key*)

## a) Qu'est ce que c'est ?

Une clé unique d'une table permet d'assurer qu'il n'y aura jamais deux fois la même valeur, cette clé peut-être composée d'un ou de plusieurs attributs.

| id | compte | password |
|---|---|---|
| 0 | MrPanda | b6589fc6ab0dc82... |
| 1 | Lapindu93 | 356a192b7913b04... |
| 2 | Lapindu95 | d4ab9237bacccdf... |

Sur cet exemple, l'attribut `compte` est une clé unique.

## b) Syntaxe

UNIQUE KEY `nom_UNIQUE` (`col_nom`)

Ex : UNIQUE KEY `compte_UNIQUE` (`compte`)

# I.2 / Une clé indexée (index_key*)

## a) Qu'est ce que c'est ?

Une clé indexée est composée d'un ou de plusieurs attributs. Elle permet d'accélérer les requêtes et est nécessaire pour une clé étrangère.

| id | compte | animal_totem |
|---|---|---|
| 0 | MrPanda | Panda |
| 1 | Lapindu93 | Lapin |
| 2 | Lapindu95 | Lapin |

Sur cet exemple, l'attribut `animal_totem` est une clé indexée.

## b) Syntaxe

KEY `nom_idx` (`col_nom`)

Ex : KEY `animal_totem_idx` (`animal_totem`)

# I.2 / Une clé étrangère (constraint*)

## a) Qu'est ce que c'est ?

Une clé étrangère est composée d'un attribut. Elle permet de gérer des association entre plusieurs tables.

| id | compte | id_animal_totem |
|---|---|---|
| 0 | MrPanda | 0 |
| 1 | Lapindu93 | 1 |
| 2 | Lapindu95 | 1 |

| id | animal | Couleur |
|---|---|---|
| 0 | Panda | Orange |
| 1 | Lapin | Blanc |

Sur cet exemple, l'attribut `id_animal_totem` est une clé étrangère.

Les mots clés ON DELETE et ON UPDATE sont utilisés pour déterminer le comportement des clés étrangères en cas de suppression ou de modification.

# I.2 / Une clé étrangère (constraint*)

## b) Les options des clés

Il existe 4 comportements possibles : RESTRICT, NO ACTION, SET NULL et CASCADE.

Avec MySQL RESTRICT et NO ACTION ont le même comportement. Si l'on essaye de modifier ou de supprimer la valeur étrangère, la requête sera annulée.

Si on choisit SET NULL, alors la valeur modifiée ou supprimée sera passée à NULL.

Avec CASCADE, cela modifie ou supprime la ligne où la valeur est référencée.

## c) Syntaxe

```
CONSTRAINT `nom_FK` FOREIGN KEY (`col_nom`)
REFERENCES `tbl_nom` (`col_nom`)
ON DELETE {RESTRICT | CASCADE | SET NULL | NO ACTION}
ON UPDATE {RESTRICT | CASCADE | SET NULL | NO ACTION}
```

Ex : `CONSTRAINT `id_animal_totem_FK` FOREIGN KEY (`id_animal_totem`)`

`REFERENCES `tbl_animal` (`id`) ON DELETE SET NULL ON UPDATE CASCADE`

# I.2 / Créer une table

## f) Exemple

```
CREATE TABLE IF NOT EXISTS `ecoles` (
    `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
    `ecole` VARCHAR(45) NOT NULL DEFAULT 'Aucun Nom',
    `description` TEXT COMMENT 'Description de l''école',
    `horaire_ouverture` TIME,
    `horaire_fermeture` TIME,
    PRIMARY KEY (`id`)
)
COMMENT = 'Liste des écoles informatiques';
```

# I.2 / Supprimer une table

## a) Syntaxe

```
DROP TABLE [IF EXISTS] `tbl_nom`;
```

Si « IF EXISTS » n'est pas présent dans la requête, cela peut déclencher une erreur si aucune table de ce nom existe.

## b) Exemple

```
DROP TABLE IF EXISTS `ma_table`;
```

Cette requête supprimera la table « ma_table » ainsi que tout son contenu si elle existe.

# I.2 / Workbench

## a) Importer ou exporter une base de données

→ Server → Data [Export | Import]

Pour les Export, cocher « Export to Self-Contained File » et « Include Create Schema ».

Pour les Import, cocher « Import from Self-Contained File ».

Pour modifier le répertoire de destination par defaut :

→ Edit → Preferences → Administration → Export Directory Path

## b) Instance MySQL offline

→ Serveur → Startup / Shutdown

## c) Les requêtes

Nouvelle fenêtre de requête : Le premier icône à gauche en haut.

Exécuter les requêtes : L'icône du petit éclair dans la fenêtre de requête.

Enregistrer un fichiers SQL : L'icône de disquette dans la fenêtre de requête.

L'historique des requêtes : Le menu déroulant dans le panel Output en bas.

## d) Les tables

Dans le panel de gauche, après un clic droit sur une table

Informations : Table inspector

Modifier une table : Alter table