10/07/2017

Une base de donnait contient des tables qui, elles-mêmes sont organisées en colonnes, dans lesquelles sont stockées des données.

My sql est un système de gestion de base de données relationnelles

Logiciel qui permet de gérer les bases de données, les grosses quantités d’informations

Il utilise le langage SQL

Base de données (CRUD) : Create, Read, Update, Delete

Installation du serveur (base de données) et du client (logiciel qui permet de manipuler les données). Le langage sql pemet de parler (donner des instructions) au client qui fait le lien avec le serveur.

SGBDR = Théorie relationnelle

Relations => Tables

En têtes et corps

SQL => Langage qui permet d’interagir avec les bases de données relationnelles

Organisation

* Présentation des données sous forme de tables
* Chaque table définit un certain nombre de colonnes qui représentent les attributs de l’en tête (ex : colonnes => numéro, nom, prénom, email…)
* La table sera aussi composée de lignes où on introduit des données qui représentent les valeurs attribuées aux colonnes

Pour se connecter

Cmd

Mysql –h localhost –u root –p[mdp] **ou** mysql –u root –p

C:\Users\m.kouta>mysql –u random -p

On peut ajouter --default-character-set=utf8 pour spécifier l’encodage de la base.

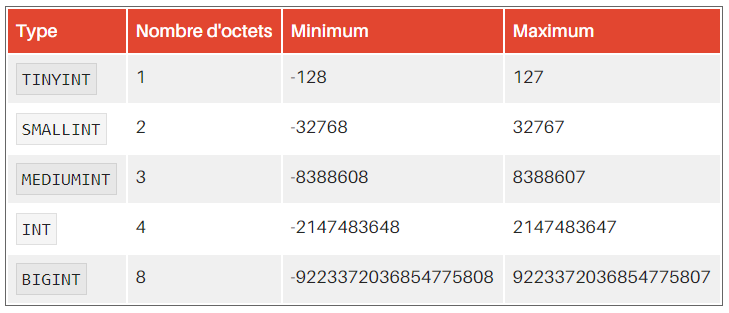
\n retour à la ligne

\t tabulation

Choisir la base de données sur laquelle on veut travailler : USE nom\_de\_la\_base ;

Avec le langage sql, il faut toujours terminer une instruction avec un ‘;’

Pour les commentaires, on peut utiliser ‘#’ et ‘—‘ (de préférence)



Quand on écrit des chaînes de caractères (une phrase, pas de chiffres), on utilise des guillements (‘ ‘).

SELECT

C’est la commande qui permet la sélection des données et leur affichage.

CREATE DATABASE exampleDB;

CREATE USER 'example\_user'@'localhost' IDENTIFIED BY 'password';

GRANT ALL PRIVILEGES ON exampleDB.\* TO 'example\_user'@'localhost';

Résumé premier chapitre :

* MySQL peut s'utiliser en ligne de commande ou avec une interface graphique.
* Pour se connecter à MySQL en ligne de commande, on utilise :mysql -u utilisateur [-h hôte] -p.
* Pour terminer une instruction SQL, on utilise le caractère ;.
* En SQL, les chaînes de caractères doivent être entourées de guillemets simples '.
* Lorsque l'on se connecte à MySQL, il faut définir l'encodage utilisé, soit directement dans la connexion avec l'option--default-character-set, soit avec la commande *SET NAMES ‘utf8’*.

Les types numériques

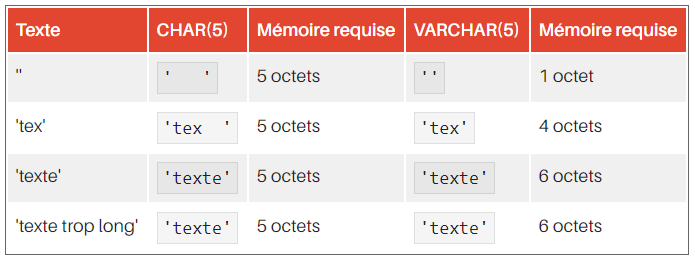
Cinq mots clés permettent de stocker des nombres décimaux dans une colonne :

* DECIMAL => Exemple DECIMAL (5,3) => On peut stocker des nombres de 5 chiffres, dont 3 chiffres après la virgule => ex : 12.345, -67.891
* NUMERIC => Pareil que DECIMAL
* FLOAT +> Ne supporte pas de paramètres => Stockage dans 4octets dans les autres SGBDR => Stckogae de 56,6789 en 56,6790000000000001.
* REAL => Pareil que FLOAT sauf stockage dans 8 octets => Il conseillé de l’utiliser à la place de FLOAT.
* DOUBLE

Les types alphanumériques

On peut utiliser les types CHAR et VARCHAR pour stocker des textes relativement courts (moins de 255 caractères).

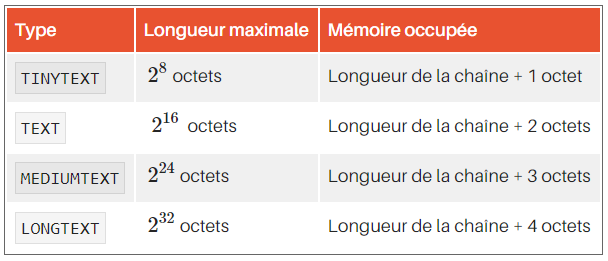
VAR(5) remplira 5 caractères. S’il y’en a moins, le reste sera complété par des espaces vides



11/07/2018

**Les textes**

On peut aussi stocker des textes de plus de 255 caractères.



On utilise les chaînes binaires pour stocker les données brutes (images, sons…). On parle des types BLOB avec des variantes :

* TINYBLOB
* BLOB
* MEDIUMBLOB
* LONGBLOB

Les types SET et NUM sont propres à MySQL. Ils permettent de stocker des chaînes de caractère dont les valeurs sont prédéfinies par l’utilisateur.

Règle de base

Lorsqu’on conçoit une base de données, sa structure (tables + colonnes) ne doit pas changer lorsqu’on ajoute des données.

Les types temporels

Il s’agit de :

* DATE => Date => ‘AAAA-MM-JJ’
* DATETIME => Date et heure => ‘AAAA-MM-JJ HH:MM:SS’
* TIME => Heure => ‘HH:MM:SS
* TIMESTAMP => Stockage d’une date sous un format numérique => AAAAMMJJHHMMSS

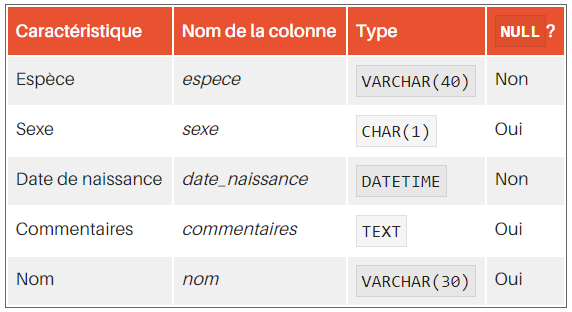
Résumé second chapitre

* MySQL définit plusieurs types de données : des numériques entiers, des numériques décimaux, des textes alphanumériques, des chaînes binaires alphanumériques et des données temporelles.
* Il est important de toujours utiliser le type de données adapté à la situation.
* SET et ENUM sont des types de données qui n'existent que chez MySQL. Il vaut donc mieux éviter de les utiliser.

Création d’une base de données

* Ne jamais utiliser d’accents ou d’espaces dans les noms de bases, de tables ou de colonnes
* Si un nom de base, table ou colonne est composé de plusieurs mots, on les sépare par un ‘\_’ ou une majuscule. Ex : premier\_test ou premierTest.
* Les commandes et mots-clés SQL doivent toujours être en majuscule
* [ Suppression => DROP DATABASE nom\_de\_la\_base ; ] ou [ DROP DATABASE IF EXISTS elevage ; ] =>>>>>> DROP TABLE Nom\_de\_la\_table

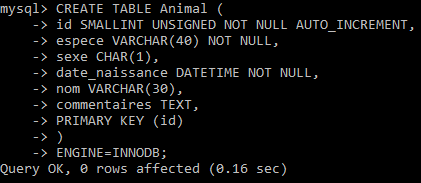
1. Création base => CREATE DATABASE nom\_de\_la\_base CHARACTER SET ‘utf8’ ;
2. Sélectionner la base créée => USE nom\_de\_la\_base
3. Créer les tables qui composent la base
   1. Avant de les créer, il faut bien choisir les colonnes qui la composeront. Cela signifie, par exemple, que lorsqu’on veut créer une table « Animal » pour une base devant contenir les données d’un élevage d’animaux, on essaye de réfléchir d’abord à toutes les caractéristiques des animaux qu’on veut dans notre table. Ex : Espèce, sexe, date de naissance, nom. On peut aussi ajouter un champs « commentaires » qui peut s’avérer utile pour des infos supplémentaires à ajouter.
      1. Espèce : VARCHAR(40)
      2. Sexe : CHAR(1) => M, F ou A
      3. Date de naissance : DATETIME
      4. Nom : VARCHAR(30)
      5. Commentaires : TEXT
   2. Il faudra aussi définir si on autorise les colonnes à ne pas stocker de valeurs avec le type NULL
   3. Enfin, il faudra ajouter la colonne id qui correspond à un numéro unique qui permet d’identifier la table. Il faut utiliser le mot clé PRIMARY KEY(id).



1. Il faut ensuite choisir le moteur de table qu’on utilisera. Ce sont des moteurs de stockage des données contenues dans les tables. Pour MySQL, il s’agit de MyISAM et InnoDB (conseillé).

Ajouter à la fin de la création de la table => ENGINE = INNODB ;

Pour définir une colonne, il faut donc donner son nom en premier, puis sa description. La description est constituée au minimum du type de la colonne.



SHOW TABLES ; => Liste les tables de la base de données

DESCRIBE Nom\_de\_la\_table => Liste les colonnes de la table avec leurs caractéristiques

Apporter des modifications à la table

ALTER TABLE Nom\_de\_la\_table [ ADD (pour ajouter) ; DROP (pour supprimer) ; CHANGE ou MODIFY (pour modifier) ]

* + ADD => ALTER TABLE Nom\_de\_la\_table ADD COLUMN nom\_de\_la\_colonne VARCHAR(x) NOT NULL (ou pas)

Pour plus d’informations sur la commande ALTER => <https://dev.mysql.com/doc/refman/5.5/en/alter-table.html>

Insérer des lignes dans une table

INSERT INTO Nom\_de\_la\_table

VALUES (données 1ère colonne, données 2nde colonne, NULL (si pas de données pour la 3e colonne) ………) ;

Ou alors

INSERT INTO Nom\_de\_la\_table (colonne3, colonne1, colonne2…)

VALUES (‘valeur colonne3’, ‘valeur colonne1’, ‘valeur colonne2’…)

Voir les lignes d’une table

SELECT \* FROM Nom\_de\_la\_table

Le (\*) peut être remplacé par le nom de la colonne que l’on souhaite voir. Sinon, ici, il permet de voir tout le contenu de la table.

Insérer des données dans une table, à partir d’un fichier externe

1. Ouvrir un éditeur de texte et y copier les données qu’on veut faire figurer dans notre table
2. Sauver le fichier, de préférence avec une extension .sql
3. Lancer la commande : SOURCE C:/chemin\_du\_fichier/fichier.sql [Seulement s’il s’agit d’un fichier contenant des requêtes SQL]
4. Lancer les commandes suivantes :

LOAD DATA [LOCAL] INFILE ‘fichier.sql’

INTO TABLE Nom\_de\_la\_table

FIELDS TERMINATED BY ‘ ; ‘ -- si les colonnes sont séparées par un ‘ ;’, comme pour les fichiers csv

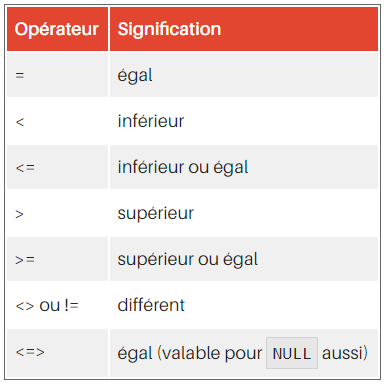
LINES TERMINATED BY ‘\n’ -- si les lignes sont séparées par un ‘saut à la ligne’

IGNORE 1 LINES -- Ignorer la dernière

Colonnes du fichier

12/07/2018

Sélection de données



Si on prend une table « Animal » avec différentes espèces d’animaux et qu’on ne voulait montrer que les chats, on ferait : SELECT \* FROM Animal WHERE espece=’chat’ ;

Après WHERE, on peut ajouter :

* AND (ou &&) ;
* OR (ou ||) ;
* XOR (qui est un OU exclusif => Ex soit A, soit B, mais pas les deux) ;
* AND NOT (AND ! => Tout sauf…)

16/07/2018

Pour vérifier qu’une valeur est nulle ou pas, on utilise IS NULL (ou < = > NULL) ou IS NOT NULL comme commandes.

Tri des données [ASC et DESC]

Ajouter après les commandes SELECT et WHERE (s’il existe), on ajoute ORDER BY nom\_de\_colonne (on peut y ajouter ASC [ascendant] et DESC [descendant]).

On peut aussi trier sur plusieurs colonnes en faisant : ORDER BY nom\_de\_colonne1, nom\_de\_colonne2.

Eviter les doublons [DISTINCT]

Commande : SELECT DISTINCT nom\_de\_colonne FROM nom\_de\_table ;

Si on avait, dans une base de données d’animaux, 40 chiens, 30 chats et 20 moutons, cette commande n’affichera que les espèces présentes, donc chien, chat et mouton.

Restreindre les résultats [LIMIT - OFFSET]

La commande LIMIT permet de limiter le nombre de lignes renvoyées par notre commande. On l’écrit après SELECT et les autres (WHERE, ORDER BY).

Par exemple, la commande LIMIT 10 permet de n’’afficher que les 10 premières lignes. Si on décide d’y ajouter OFFSET 0 par exemple, le résultat commencera par la 1ère ligne. Si c’est par exemple OFFSET 3, le résultat commencera par la 4e ligne.

Sélection avec précision [WHERE… IN]

Avec WHERE, on peut faire des sélections plus précises :

Si on voulait juste sortir les lignes concernant 5 noms d’animaux, on peut faire :

SELECT \* FROM Nom\_de\_la\_table WHERE nom=’nom1’ OR nom=’nom2’ OR nom=’nom3’ OR nom=’nom4’ OR nom=’nom5’ ;

Ou encore :

SELECT \* FROM Nom\_de\_la\_table WHERE nom IN (‘nom1’, ‘nom2’, ‘nom3’, ‘nom4’, ‘nom5’) ;

16/07/2018

Aller plus loin avec le WHERE

Choix de chaînes de caractère [LIKE – NOT LIKE]

Si on chaine des mots précis dans un champ (ou colonne) bien défini, ou utilise LIKE comme par exemple : SELECT \* FROM Nom\_de\_la\_table WHERE commentaires LIKE ‘attention%’ ;

Ici, on aura toutes les lignes où le champ « commentaires » contient le mot « bizarre ». L’identification est faite grâce au « % » qu’on appelle ici un « joker ».

Il y’a aussi le joker « \_ ». Si on mettait par ex « b\_ » après le LIKE, cela signifierait qu’on cherche toutes les chaînes de caractère contenant 2 lettres dont la 1ère est ‘b’ (ba, bf, b8, b&…).

'%ch%ne' cherchera toutes les chaînes de caractères contenant 'ch' et finissant par 'ne' ('chne', 'chine', 'échine', 'le pays le plus peuplé du monde est la Chine').

'p\_rl\_' cherchera toutes les chaînes de caractères commençant par un 'p' suivi d'un caractère, puis de 'rl' et enfin se terminant par un caractère ('parle', 'perla', 'perla').

Si la chaîne contient dèjà ‘%’, on utilisera \ pour éviter l’amalgame :

WHERE commentaires LIKE ‘%\%%’ ;

Exclusion [NOT LIKE]

Pour exclure les champs qui contiennent un caractère ou une chaîne spécifiques, on utilise NOT LIKE.

Par ex : SELECT \* FROM Nom\_de\_la\_table WHERE nom NOT LIKE ‘%a% ; => exclut tous les champs qui contiennent la lettre « a ».

Chercher dans les intervalles [BETWEEN…AND – NOT BETWEEN]

Par ex, pour chercher les animaux nés entre 2010 et 2018, on fait :

SELECT \* FROM Nom\_de\_la\_table WHERE date\_naissance BETWEEN ‘2010-01-01’ AND ‘2018-01-01’;

Le BETWEEN peut aussi s’utiliser avec des chiffres, des caractères, des chaînes de caractères…

On peut aussi utiliser NOT BETWEEN.

Sensibilité à la casse [BINARY]

Sans précision, les commandes sont insensibles à la casse. Sinon, pour prendre en compte les majuscules et minuscules, on utilise BINARY.

Ex : LIKE BINARY, BETWEEN BINARY…

Sauvegarde de la base de données [DUMP]

Il faut d’abord quitter le programme => mysql > exit

Ensuite : mysqldump -u nom\_utilisateur -p --opt Nom\_de\_la\_table > [nom\_de\_sauvegarde]\_sauvegarde.sql

30/07/2018

Supprimer des lignes d’une table

DELETE FROM Nom\_de\_la\_table WHERE nom = ‘Fatou’ ;

Pour supprimer toutes les lignes, on ne précise pas le WHERE => DELETE FROM Nom\_de\_la\_table

Pour mettre à jour les informations d’une ligne

UPDATE Nom\_de\_la\_table

SET nom=’Burkina’, sexe=’F’

WHERE id=’n° id’ ;

Pour mettre à jour toutes les lignes d’une table, on omet le WHERE.

Par ex, si on fait :

UPDATE Nom\_de\_la\_table

SET commentaires=’refaire un check up’ ;

Ce commentaire s’afficherait sur toutes les lignes de la table.

30/07/2018 2nde phase – Après le QUIZ

Les INDEX peuvent être créés de deux manières :

* Soit directement lors de la création de la table
* Soit en les ajoutant par la suite.

CREATE TABLE nom\_table (

colonne1 INT KEY, -- Crée un index simple sur colonne1

colonne2 VARCHAR(40) UNIQUE, -- Crée un index unique sur colonne2;