SQL – Structured Query Language

Pré-requis – Outils et logiciels

Pour la suite de cette formation, vous allez devoir installer quelques outils

1. Serveur de Base de Données

   - La solution « simple » sur **Windows** : Installer **WampServer** ou **XAMPP**
     Cela vous fournira un serveur LAMP fonctionnel (Apache, MySQL, PHP),

   - La solution d'avenir : utiliser **Docker**, un gestionnaire de conteneurs.
     C'est la solution que vous aurez le plus de chances de croiser en entreprise.

     La courbe d'apprentissage est un peu plus ardue que pour la solution précédente, mais lorsque vous avez une config fonctionnelle, c'est beaucoup plus efficace et souple.

     Docker est une solution **multi-plateformes**, fonctionnant aussi bien sous Windows, Linux ou Mac.

Installation de Docker sous Windows

1. Installer Docker pour Windows (https://www.docker.com/)
2. Installer WSL 2 (Windows Subsystem for Linux)

- Ouvrir un terminal powershell et entrer les commande suivante :

  wsl.exe --update

  Puis :

  wsl --set-default-version 2

- Ajouter votre utilisateur local au groupe docker-users
  - Ouvrez un terminal PowerShell en tant qu'administrateur et entre la commande suivante :
    - net localgroup docker-users "your-user-id" /ADD

- Fermez votre session et rouvrez là

Installation de Docker sous Windows

- Créez un répertoire qui sera votre répertoire de travail, puis clonez-y le dépôt git suivant :

  git clone https://github.com/guirod/docker-lamp.git

  Ce dépôt contiendra l'essentiel pour la suite de votre formation (serveur MySQL, serveur Apache, PHP 8.1, serveur SMTP + mailcatcher) et sera mis à jour au besoin
- Quelques commandes utiles :
  - **Lancer et couper vos containers (serveurs) :**

    docker-compose up –d

    docker-compose down
  - **Lister les containers**

    docker container ls
  - **Obtenir une connexion en root sur le container**

    docker exec -it nom_du_container executable
  - **Exemple pour se connecter au container apache et lancer l'exécutable "bash"**

    docker exec -it docker-lamp-docker-lamp-php-1 bash

Installation d'une GUI (Graphic User Inteface) pour ses bases de données

- 2 principaux outils, je vous conseille d'installer les 2.
  - MySQL Workbench
    - https://www.mysql.com/fr/products/workbench/
    - Graphiquement plus agréable
    - Parfois un peu lourd et buggué pour certaines tâches (import et export notamment)
  - HeidiSQL
    - https://www.heidisql.com/
    - Un peu moins convivial, mais très puissant.
    - Compatible avec de nombreux SGBDr (MySQL/MariaDB, PostgreSQL, SQL Server)
  - Pour chacun de ces outils, configurez et sauvegarder votre connexion :
    - Host : localhost / 127.0.0.1
    - Username : root
    - Password : p@ssw0rd pour les utilisateurs Docker (configuré dans le fichier compose.yaml)

Connexion en ligne de commande

- Se connecter au container MySQL

  docker exec -it docker-lamp-docker-lamp-mysql-1 bash

- Se connecter avec son utilisateur

  mysql -hlocalhost -uroot -pp@ssw0rd (problème, le pass sera dans votre bash history)
  mysql –h localhost –u root –p (un prompt vous demandera votre mdp)

- Enjoy (Non. Mais parfois vous n'aurez pas le choix)

SQL
Introduction

SQL – Structured Query Language

- Langage informatique normalisé
- N'est pas un langage procédural
- Créé par IBM en 1974
- Normalisé depuis 1986
  - Recommandation de l'ANSI
  - Norme ISO/CEI 9075 - Technologies de l'information - Langages de base de données SQL

SQL – Structured Query Language

Le langage SQL permet principalement de faire du CRUD :

- Create : Ecrire, insérer des données
- Read : Lire des données
- Update : Modifier des données
- Delete : Supprimer des données

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

- Créer, modifier, supprimer des bases de données
- Créer, modifier, supprimer des tables
- Créer, modifier, supprimer des utilisateurs et gérer leurs privilèges

Base De Données Relationnelle

- 3 objectifs :
  - **Garantir l'intégrité des données**
    - Eviter l'altération des données (usure, pannes, erreurs, malveillance)
    - Eviter l'incohérence des données
      - La duplication des données : exemple 2 adresses différentes pour un utilisateur
      - Les valeurs aberrantes : exemple Age < 0
  - **Garantir l'indépendance entre données et traitements**
    - L'information correspondant à une donnée doit être compréhensible indépendamment
    - Si une donnée est calculée, on évitera de la stocker en BDD
  - **Traitements rationalisés**
    - Une fois les données définies, les traitements consistent principalement à ajouter, modifier, supprimer et consulter des données

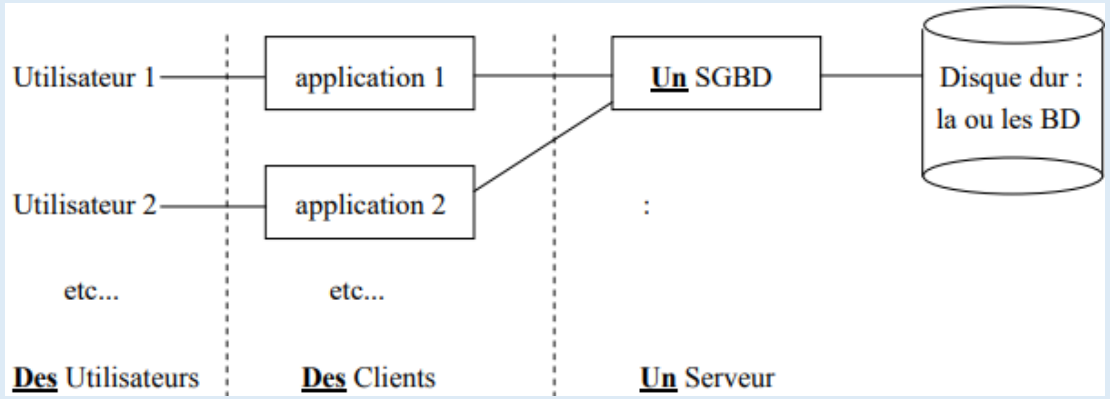
SGBDr – Système de Gestion de Base de Données Relationnelles

- Ensemble de logiciels faisant l'interface entre l'utilisateur et la BDD

- **Objectifs** :
  - Stockage et requêtage des données
  - Multi-utilisateurs => Gestion des droits d'accès, des collisions
  - Limiter la redondance (duplication de données) => Intégrité des données

SGBDr – Système de Gestion de Base de Données Relationnelles

- **Architecture**
  - Le plus souvent on utilise une architecture client/serveur



  - Une BDD est stockée sur 1 ou plusieurs disques durs
  - 1 seul SGBD par BDD
  - Plusieurs applications / API (clients) peuvent communiquer avec le SGBD

SGBDr – Système de Gestion de Base de Données Relationnelles

- L'objectif majeur du SGBD étant d'assurer l'intégrité, voici quelques moyens mis en œuvre
  - Conformité au modèle : Vérifier que les données saisies soient cohérentes
  - Accès concurrents : Eviter les incohérences dues à des modifications multiples au même moment (transactions)
  - Gestion sur panne : Garantir le maintien de la cohérence même quand une panne intervient
  - Autorisations d'accès

SGBDr – Système de Gestion de Base de Données Relationnelles

- Les SGBDr les plus connus :
  - MySQL (Oracle) / MariaDB (Fork open source de MySQL)
  - OracleDB
  - PostgreSQL : alternative open source à OracleDB
  - SQL Server (Microsoft)

SQL
Administration d'une BDD

Création de BDD

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name [create_option]...
create_option: [DEFAULT] {
    CHARACTER SET [=] charset_name |
    COLLATE [=] collation_name |
    ENCRYPTION [=] {'Y' | 'N'}
}
```

IF NOT EXISTS : Si cette option est présente, cette commande ne lèvera pas d'exception si la BDD existe déjà. Il n'y fera aucune modification.

create_option permet de définir des paramètres de la BDD.

Pour plus d'informations : https://dev.mysql.com/doc/refman/8.0/en/create-database.html

Suppression et modification de BDD

- SUPPRESSION

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

- MODIFICATION

```
ALTER {DATABASE | SCHEMA} [db_name] alter_option ...
alter_option: { [DEFAULT] CHARACTER SET [=] charset_name |
      [DEFAULT] COLLATE [=] collation_name |
      [DEFAULT] ENCRYPTION [=] {'Y' | 'N'} |
      READ ONLY [=] {DEFAULT | 0 | 1} }
```

- SELECTIONNER UNE BDD : Nécessaire pour pouvoir effectuer des opérations (création ou consultation)

```
USE db_name;
```

- DIVERS

```
SHOW DATABASES;
```

Gestion des droits et utilisateurs

- **Sécurité**
  - Limiter au strict nécessaire les droits des utilisateurs exposés (application web)
  - Utiliser des mots de passe forts et uniques
  - Maintenir ses logiciels à jour

Gestion des droits et utilisateurs

- Création d'un utilisateur
  CREATE USER 'alice'@'localhost' IDENTIFIED BY 'P@ssw0rd';

- Afficher les utilisateurs
  SELECT * FROM mysql.user;

- Assigner les privileges
  GRANT PRIVILEGE ON database.table TO 'username'@'host';

  Example :
  GRANT CREATE, ALTER, DROP, INSERT, UPDATE, DELETE, SELECT, REFERENCES, RELOAD
  ON *.* TO 'alice'@'localhost' WITH GRANT OPTION;

- Révoquer un privilege
  REVOKE type_of_permission ON database_name.table_name FROM 'username'@'host'

Gestion des droits et utilisateurs

- Assigner tous les privileges à un utilisateur
  GRANT ALL PRIVILEGES ON database.table TO 'username'@'host';

- Appliquer les modifications:
  FLUSH PRIVILEGES;

- Afficher les privileges d'un utilisateur
  SHOW GRANTS FOR 'username'@'host';

- Supprimer un utilisateur
  DROP USER 'username'@'localhost';

Exercice

- Coder un script pour :
  Créer un autre utilisateur ‘bob’
  Créer une BDD pour bob lui assignant tous les privileges

- Ressource :
  https://www.digitalocean.com/community/tutorials/how-to-create-a-new-user-and-grant-permissions-in-mysql

SQL
Structure des données

Les Tables

- Les tables sont les objets qui contiennent toutes les données d'une BDD.

- Données organisées dans un tableau à deux dimensions
  - Les lignes sont des tuples, enregistrements ou n-uplets
  - Les colonnes sont des attributs

Les Tables

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

Les types de données

- Numériques exacts :
  - Entiers signés :
    - INTEGER 4 octets => -2147483648 à +2147483647
    - SMALLINT (2 octets), BIGINT (8 octets)
  - Décimaux signés :
    - NUMERIC, DECIMAL
- Numériques approximatifs
  - REAL, DOUBLE PRECISION, FLOAT
- Chaînes de caractères
  - Longueur fixe : CHAR
  - Longueur variable : VARCHAR
  - Objets de type caractère : CLOB

Les types de données

- Chaînes binaires
  - Nombre d'octets fixe : BINARY
  - Longueur variable : VARBINARY
  - Binary Large Object : BLOB
- Booléens
  - 3 valeurs : true, false, unknown
- Dates et heures
  - DATE
  - TIME/TIMESTAMP WITH/WITHOUT TIMEZONE
- DIVERS
  - XML, ARRAY, INTERVAL

Opérations sur les tables

- Création

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    (create_definition,...)
    [table_options]
    [partition_options]
```

Exemple :

```
CREATE TABLE users (
    id INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
    name VARCHAR(100),
    email VARCHAR(255) NOT NULL,
    birthdate DATE
);
```

https://dev.mysql.com/doc/refman/8.0/en/create-table.html

Opérations sur les tables

- Modification :
  - ➢Ajouter/Modifier/Supprimer un attribut
  - ➢Ajouter/Modifier/Supprimer une contrainte
  - ➢Modifier des propriétés de la table

ALTER TABLE *tbl_name*
[*alter_option* [, *alter_option*] ...]
[*partition_options*]

Exemple :
ALTER TABLE *users*
ADD *firstname VARCHAR(100)* AFTER *name;*

https://dev.mysql.com/doc/refman/8.0/en/alter-table.html

Indexes

- Dans chaque table, on peut définir un ou plusieurs indexes.
- Un ROWID est créé pour chaque enregistrement
- Avantage : Requêtes plus rapides
- Inconvénient : Peuvent surcharger le système donc à utiliser avec précaution
- Exemple :

  Création d'un index : Notez que l'on peut le définir sur une ou plusieurs colonnes

  CREATE INDEX index_name ON table_name (column1, column2, ...);

  Suppression d'un index

  ALTER TABLE table_name DROP INDEX index_name;

Contraintes

Les contraintes sont des règles appliquées aux attributs d'une table.
Elles permettent d'améliorer l'intégrité des données de la base.

- Les contraintes les plus communes sont :

  - **NOT NULL** : Contraint à avoir une valeur différente de NULL.
  - **DEFAULT** : Fournit une valeur par défaut à la colonne.
  - **UNIQUE** : Ne permet pas que 2 enregistrements aient la même valeur pour une colonne (email)
  - **CHECK** : Active une condition pour qu'un enregistrement soit valide.
  - **INDEX**
  - **PRIMARY KEY** : La clé primaire permet d'identifier de manière unique chaque enregistrement. Une clé primaire peut être composée, elle fait dans ce cas référence à plusieurs attributs de la table.
  - **FOREIGN KEY** : Une clé étrangère permet de relier 2 tables.

Contraintes - Clés

- **Clé primaire :**

  Une clé primaire permet d'identifier chaque enregistrement d'une table.
  Par définition, elle doit être UNIQUE et NOT NULL.
  Une clé primaire peut être simple (définie par un seul attribut) ou composée de plusieurs attributs.

- **Clé étrangère :**

  Une clé étrangère est une contrainte qui permet d'associer plusieurs tables entre elles tout en assurant l'intégrité des données.
  Une clé étrangère fait référence à une clé primaire d'une autre table.

  A noter :

  - Il est impossible de faire référence à une clé primaire si celle-ci n'existe pas dans la BDD.
  - Il n'est par défaut pas possible de supprimer un enregistrement si sa clé primaire est référencée comme clé étrangère dans une autre table.

Contraintes – ON DELETE / ON UPDATE

- Clé étrangère :

  Si un enregistrement est référencé dans une autre table, le SGBD empêchera la modification ou suppression de la clé primaire.
  Si l'on souhaite forcer la modification / suppression, il est nécessaire de le renseigner lors de la création de la clé secondaire.

  Pour cela on doit utiliser les instructions ON DELETE et ON UPDATE

  - ON DELETE : A la suppression, effectue une action sur tous les enregistrements enfants.
  - ON UPDATE : A la modification, reporte le changement sur tous les enregistrements enfants.

  L'usage le plus courant est d'utiliser ON DELETE CASCADE ou ON UPDATE CASCADE.
  Toutefois, c'est un comportement assez destructeur ce qui n'est pas toujours souhaitable.
  Dans ce cas, on peut utiliser des alternatives, telles que :

  ON DELETE|UPDATE [SET NULL|SET DEFAULT|NO ACTION|CASCADE]

Contraintes - Exemple

```
CREATE TABLE employes(
        id      INT    NOT NULL,
        nom VARCHAR (100)    NOT NULL,
        email        VARCHAR (100)    NOT NULL UNIQUE,
        age   INT  NOT NULL CHECK (age >= 18),
        salaire  DECIMAL (18, 2) DEFAULT 3000.00,
        PRIMARY KEY (id)
);
CREATE TABLE conges (
        id      INT PRIMARY KEY NOT NULL AUTO_INCREMENT,
        date_debut       DATE    NOT NULL,
        date_fin    DATE    NOT NULL,
        id_employe INT,
        FOREIGN KEY (id_employe) REFERENCES employes(id) ON DELETE CASCADE
        -- Pour pouvoir nommer la contrainte, utiliser la syntaxe suivante
        CONSTRAINT fk_id_employe FOREIGN KEY (id_employe) REFERENCES employes(id)
);
```

Contraintes - Suppression

- En ciblant la contrainte :

```
ALTER TABLE table_name
    ALTER COLUMN column_name DROP CONSTRAINT;
```

Ex :

```
ALTER TABLE employe
    ALTER COLUMN salaire DROP DEFAULT;
```

- Si la contrainte a un alias

```
ALTER TABLE table_name DROP CONSTRAINT constraint_alias;
```

Associations

Lorsque l'on modélise notre BDD, on va identifier différents types d'associations qui vont lier les tables entre elles.

Bien concevoir ses associations est indispensable pour satisfaire les conditions des formes normales et assurer l'intégrité des données.

On dénombre **5 types de relations** :

- One-to-One
- One-to-Many / Many-to-One
- Many-to-Many
- Self-Reference (association récursive)

Associations – One-to-One

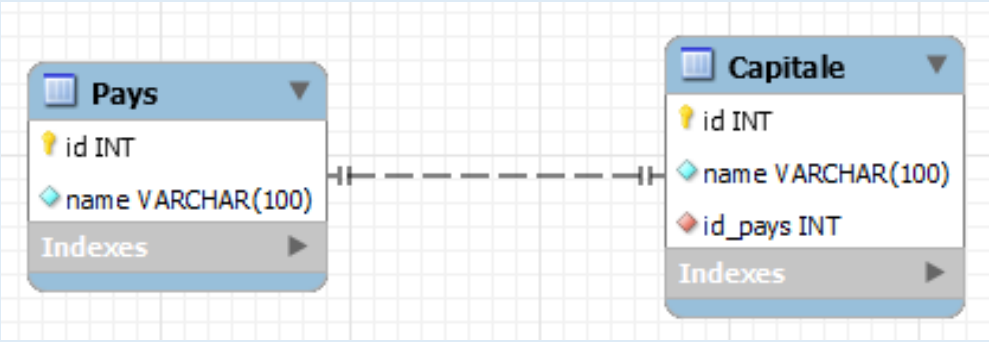
Permet d'associer un et un seul enregistrement d'une table T1 à un et un seul enregistrement d'une table T2.

Implémentée par l'utilisation d'une clé étrangère.

- Veiller à ajouter des contrôles (code et/ou BDD) pour assurer l'intégrité des données (ex : contrainte d'unicité sur la clé étrangère)
- Cette contrainte sera rarement utilisée, car généralement, la donnée peut directement être ajoutée dans la table « parent »

Exemples :

- Citoyen <-> Passeport
- Capitale <-> Pays



Associations – One-to-Many et Many-to-One

One-to-Many :

Associe un enregistrement d'une table T1 à de multiples enregistrements d'une table T2

Many-to-One :

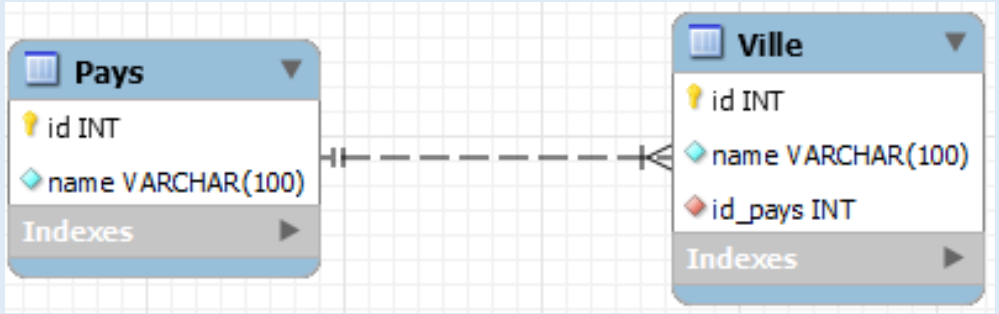
Associe de multiples enregistrements d'une table T1 à un unique enregistrement d'une table T2

Implémentée par l'utilisation d'une clé étrangère.

- Veiller à ajouter des contrôles (code et/ou BDD) pour assurer l'intégrité des données.

Exemple :

- Pays <-> Ville : Un pays possède plusieurs villes, une ville n'appartient qu'à un seul pays



Associations – Many-to-Many

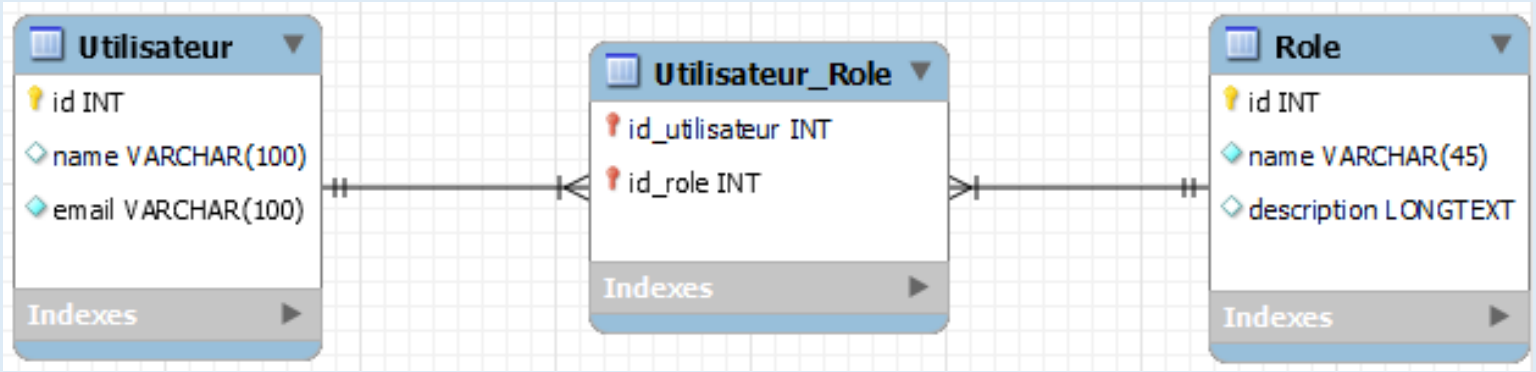
Associe un enregistrement d'une table T1 à de multiples enregistrements de la table T2 et vice versa.

Pour implémenter une association Many-to-Many, il est nécessaire de passer par une **table d'association**.

La table d'association peut aussi être **porteuse de donnée**, par exemple ci-dessous, la table Utilisateur_Role pourrait indiquer la date d'expiration du rôle.

Exemple :

- Utilisateur <-> Rôle : un utilisateur peut avoir plusieurs rôles et un rôle peut être possédé par plusieurs utilisateurs.



Associations – Self-Reference

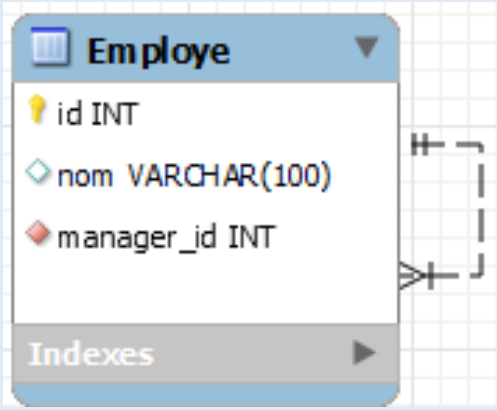
Une association Self-Reference permet d'associer un enregistrement d'une table T à un autre enregistrement d'une table T.

L'association sera souvent une association One-to-Many ou Many-to-One.

C'est notamment utilisé lorsque l'on souhaite gérer une arborescence.

Exemples :

- Catégories <-> Catégorie enfant
- Employé <-> Manager



SQL
Ressources

Documentation

- Site officiel :
  https://www.mysql.com/
- Documentation officielle :
  https://dev.mysql.com/doc/refman/8.0/en/
- Tuto W3Schools :
  https://www.w3schools.com/sql/default.asp
- MySQL Tutorial :
  https://www.mysqltutorial.org/
- Slide sur l'optimisation :
  https://www.slideshare.net/ZendCon/joinfu-the-art-of-sql-tuning-for-mysql-presentation

SQL
Travaux Pratiques

SQL
Insertion et modification de données

Insertion d'enregistrements

Maintenant que nous avons une structure de BDD cohérente, nous pouvons commencer à insérer des données dans notre base.

Pour insérer des données, nous allons utiliser la commande « INSERT INTO».

Nous allons détailler sa syntaxe.

INSERT INTO

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
    [INTO] tbl_name
    [PARTITION (partition_name [, partition_name] ...)]
    [(col_name [, col_name] ...)]
    { {VALUES | VALUE} (value_list) [, (value_list)] ... }
    [AS row_alias[(col_alias [, col_alias] ...)]]
    [ON DUPLICATE KEY UPDATE assignment_list]
```

Voici le lien vers la documentation officielle : https://dev.mysql.com/doc/refman/8.0/en/insert.html

Vous pourrez avoir des détails sur les différentes options.

INSERT INTO

Comme précédemment, la syntaxe semble barbare, mais la syntaxe que nous utiliserons le plus souvent sera beaucoup plus digeste :

```
INSERT INTO tbl_name(col_1, col_2, ...)
VALUES (value_col_1, value_col_2, …);
```

- **tbl_name** correspond au nom de notre table.
- **Col_1, col_2, …** : les noms de colonne pour lesquelles nous souhaitons préciser les valeurs de notre enregistrement
- ***value_col_1, value_col_2, … :*** *les valeurs que nous voulons insérer.*

Notez qu'il est possible de ne pas préciser la liste des colonnes, mais dans ce cas il faudra envoyer les valeurs pour toutes les colonnes de la table, dans l'ordre. Je ne suis personnellement pas fan de cette syntaxe.

INSERT INTO

Il est également possible de faire de l'insertion « de masse », en envoyant plusieurs listes de valeurs, séparées d'une virgule. On procéderait de la façon suivante :

```
INSERT INTO tbl_name(col_1, col_2)
VALUES (value_col_1_e1, value_col_2_e1), (value_col_1_e1, value_col_2_e1),
(value_col_1_e1, value_col_2_e1);
```

INSERT INTO : Exemple

- Insertion d'enregistrement unique :

```
INSERT INTO users(`name`,firstname,email)
VALUES('Cartman', 'Eric', 'eric@cartman.com');
```

- Insertion multiple d'enregistrements :

```
INSERT INTO users (`name`,firstname,email)
VALUES
    ('McCormick', 'Kenny', 'kenny@mccormick.com'),
    ('Broflovski', 'Kyle', 'kyle@broflovski.com'),
    ('Marsh', 'Stan', 'stan@marsh.com'),
    ('Garrison', 'Herbert', 'herbert@garrison.com');
```

UPDATE : Modification d'enregistrements

```
UPDATE [LOW_PRIORITY] [IGNORE] table_reference
    SET assignment_list
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

Documentation : https://dev.mysql.com/doc/refman/8.0/en/update.html

UPDATE

Ici encore, le plus souvent nous utiliserons une syntaxe allégée :

UPDATE [LOW_PRIORITY] [IGNORE] *table_reference*
    SET *col1 = val_col1, col2 = val_col2, …*
    [WHERE *where_condition*]

La clause “where” va nous permettre de cibler des enregistrements particuliers selon des critères.

Lorsque la clause “where” est absente d’une requête d’update, toutes les données de la table seront modifiées en consequence.

Il faut donc être très prudent lors de l’execution de ce genre de requête.

UPDATE : Exemple d'utilisation

Ajout d'une valeur à la colonne « birthdate » de tous les users.

```
UPDATE users
SET birthdate = '1997-08-13';
```

Modification de l'adresse mail d'un utilisateur en le recherchant par son nom et prénom.

```
UPDATE users
SET email = 'eric.cartman@south-park.com'
WHERE `name` = 'Cartman' AND firstname = 'Eric';
```

Pour visualiser les résultats :

```
SELECT * FROM users;
```

DELETE : Suppression d'enregistrements

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name [[AS] tbl_alias]
    [PARTITION (partition_name [, partition_name] ...)]
    [WHERE where_condition]
    [ORDER BY ...]
    [LIMIT row_count]
```

Documentation : https://dev.mysql.com/doc/refman/8.0/en/delete.html

DELETE : Suppression d'enregistrements

Ici encore, la commande allégée ressemblera à :

DELETE FROM *tbl_name* WHERE *where_condition*

Là encore, la clause WHERE est facultative. Si elle n'est pas renseignée (et que votre serveur MySQL le permet), toutes les données de la table seraient supprimées.

Il faut donc être extrêmement prudent lors de l'utilisation de cette commande.

DELETE : Exemples

- Suppression d'un enregistrement, ciblé par la primary key :

```
DELETE FROM users WHERE id = 1;
```

- *Suppression d'enregistrements dans une plage d'id :*

```
DELETE FROM users WHERE id BETWEEN 1 AND 3;
```

DELETE : Exemples

- Nous avons vu que la commande DELETE pouvait purger totalement une table.
- Cela peut parfois être utile. Toutefois, cela ne va pas reset le compteur “auto_increment” utilisé dans les identifiants.
- Si l’on souhaite reset le compteur d’auto_increment, il conviendra d’utiliser la commande “TRUNCATE” :

```
TRUNCATE TABLE users;
```

SQL
Opérateurs logiques

Opérateurs logiques

Nous avons commencé à manipuler quelques opérateurs logiques dans les clauses WHERE.

MySQL permet l'utilisation de nombreux opérateurs logiques. Nous allons en étudier quelques uns

Opérateurs

Opérateur	Description	Datatype concerné
=	Opérateur d'égalité	Tous
<>, !=	Différent	Tous
>	Supérieur	Numérique
<	Inférieur	Numérique
>=	Supérieur ou égal	Numérique
<=	Inférieur ou égal	Numérique
BETWEEN x and y	Compris entre x et y	Numérique, dates
IN(liste de valeurs)	Appartient à la liste de valeurs	Tous
IS NULL	Valeur est null	Tous

Opérateurs

Opérateur	Description	Datatype concerné
NOT	Opérateur logique de négation Renvoie true il évalue false, et renvoie false si il évalue true Exemple : IS NOT NULL	Opérateur logique
AND	Opérateur **ET logique** Ex : A AND B = true si A = true ET B = true	Opérateur logique
OR	Opérateur **OU logique** EX : A OR B = true si A = true ET/OU B = true	Opérateur logique
XOR	Opérateur **OU exclusif** EX : A XOR B = true si A = true OU B = true mais pas si les deux sont true	Opérateur logique

Opérateurs

Opérateur	Description	Datatype concerné
NOT	Opérateur logique de négation Renvoie true il évalue false, et renvoie false si il évalue true Exemple : IS NOT NULL	Opérateur logique
AND	Opérateur **ET logique** Ex : A AND B = true si A = true ET B = true	Opérateur logique
OR	Opérateur **OU logique** EX : A OR B = true si A = true ET/OU B = true	Opérateur logique
XOR	Opérateur **OU exclusif** EX : A XOR B = true si A = true OU B = true mais pas si les deux sont true	Opérateur logique

Opérateur « like »

- L'opérateur like est utilisé pour rechercher une chaine de caractères dans une autre.
- Cet opérateur est très utilisé pour filtrer un tableau de résultats.
- Il ne prend pas en compte la casse. Ainsi : « Cheval » LIKE « CHeVaL » vaudra true
- On l'utilise en combinaison avec le symbole '%', qui représente 0, 1 ou n caractères

Opérateur	Description	Datatype concerné
X LIKE Y	La chaine de caractères X devra être identique à Y, à l'exception de la casse.	Chaines de caractères
X LIKE %Y	La chaine Y devra se terminer par la chaine X	Chaines de caractères
X LIKE Y%	La chaine Y devra commencer par la chaine X	Chaines de caractères
X LIKE %Y%	La chaine X devra être comprise dans la chaine Y	Chaines de caractères

Opérateur « like » : exemples

Une démo vaut plus que de long discours.

Je vous invite donc à aller regarder et manipuler l'opérateur LIKE pour connaitre un peu mieux son fonctionnement, il vous sera très utile.

Documentation et exemples :

- https://www.w3schools.com/mysql/mysql_like.asp
- https://sql.sh/cours/where/like

Précédence des opérateurs

- Comme en mathématiques, les opérateurs n'ont pas tous la même priorité lors de l'évaluation d'une opération logique.
- Voici la liste des opérateurs, de la priorité la plus basse à la plus haute.

:=
||, OR, XOR
&&, AND
BETWEEN, CASE, WHEN, THEN, ELSE
=, <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN
|
&
<<, >>
-, +
*, /, DIV, %, MOD
^
- (unary minus), ~ (unary bit inversion)
- NOT, !
- BINARY, COLLATE