



## Cours 5

**Présentation des Bases de données Relationnelles et non relationnelles**

**Présentation de l'écosystème**

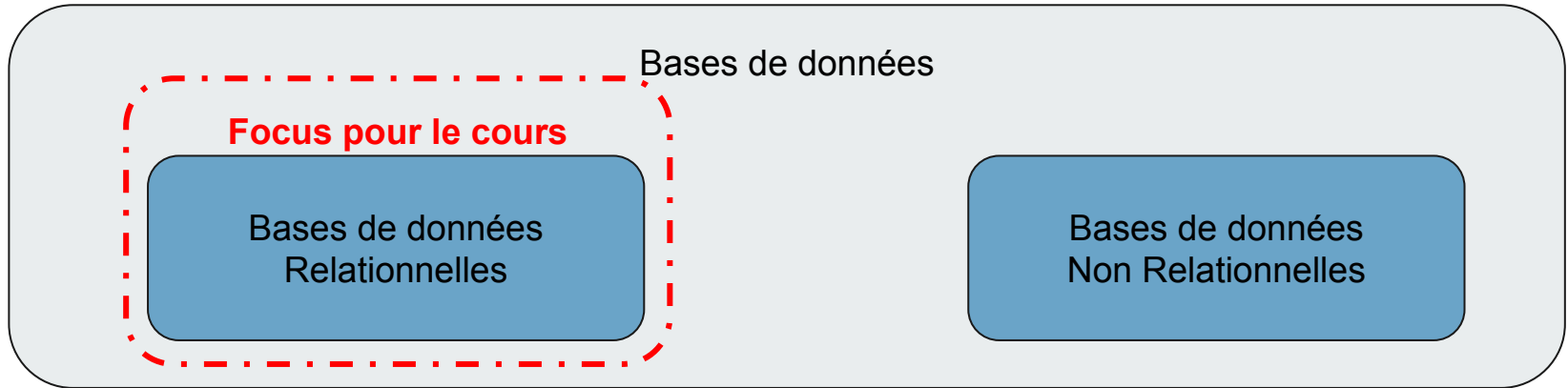
**Installation de WampServer**

**Présentation d'une requête SQL - décomposition**



# Présentation des bases de données

Il existe aujourd'hui 2 grands types de bases de données : **Relationnelles** et **Non Relationnelles**.



## Les technologies liées aux bases de données relationnelles



Bases de données  
Relationnelles





# MySQL

MySQL est un SGBDR (Système de Gestion de Bases de Données Relationnelles) open-source, développé par Oracle.

MySQL  SQL

Ne pas confondre MySQL qui est un SGBDR et SQL (Structured Query Language) qui est un langage qui permet d'effectuer des requêtes sur des bases de données relationnelles.



## MySQL et MariaDB

MySQL et MariaDB sont des logiciels qui nous permettent d'exploiter une BDD. Ils servent d'interface entre l'utilisateur et la BDD.

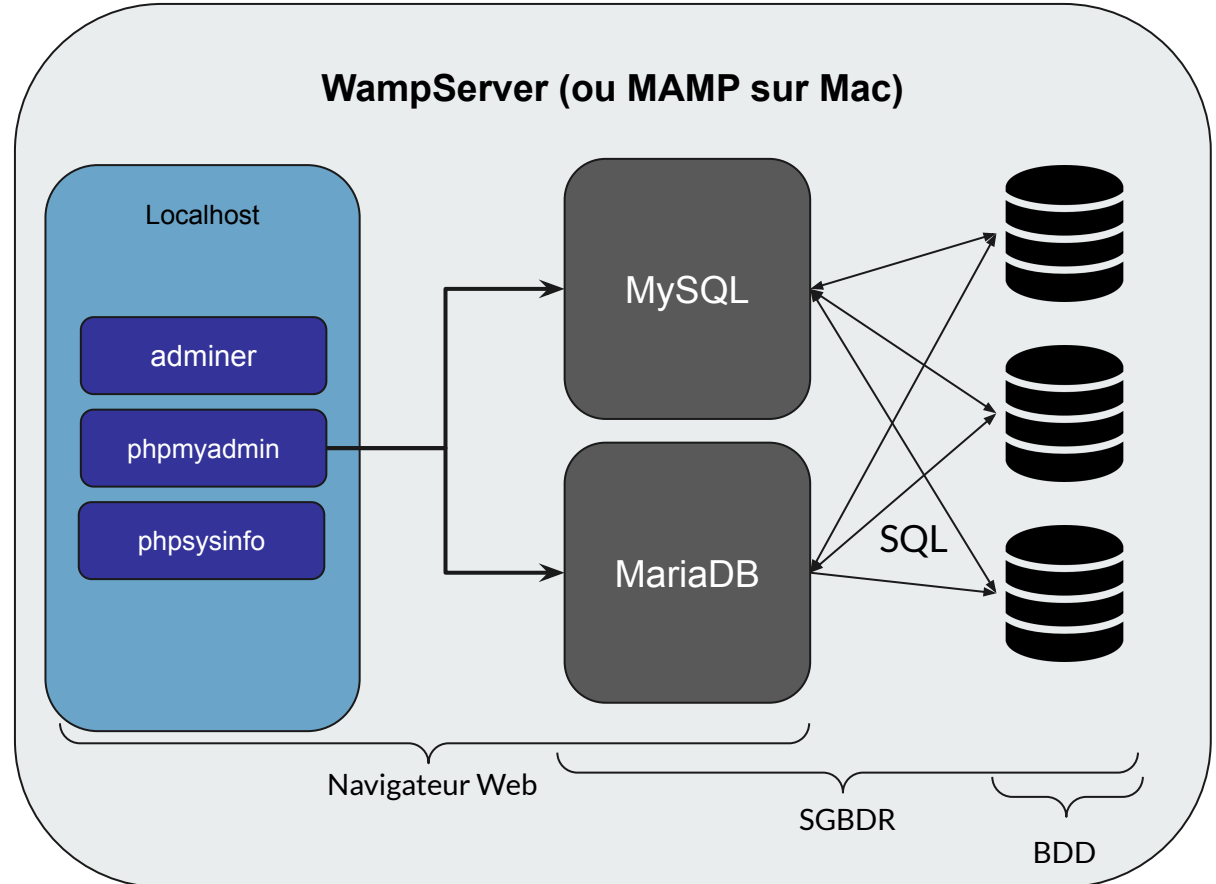




# Installation de WampServer



# Architecture

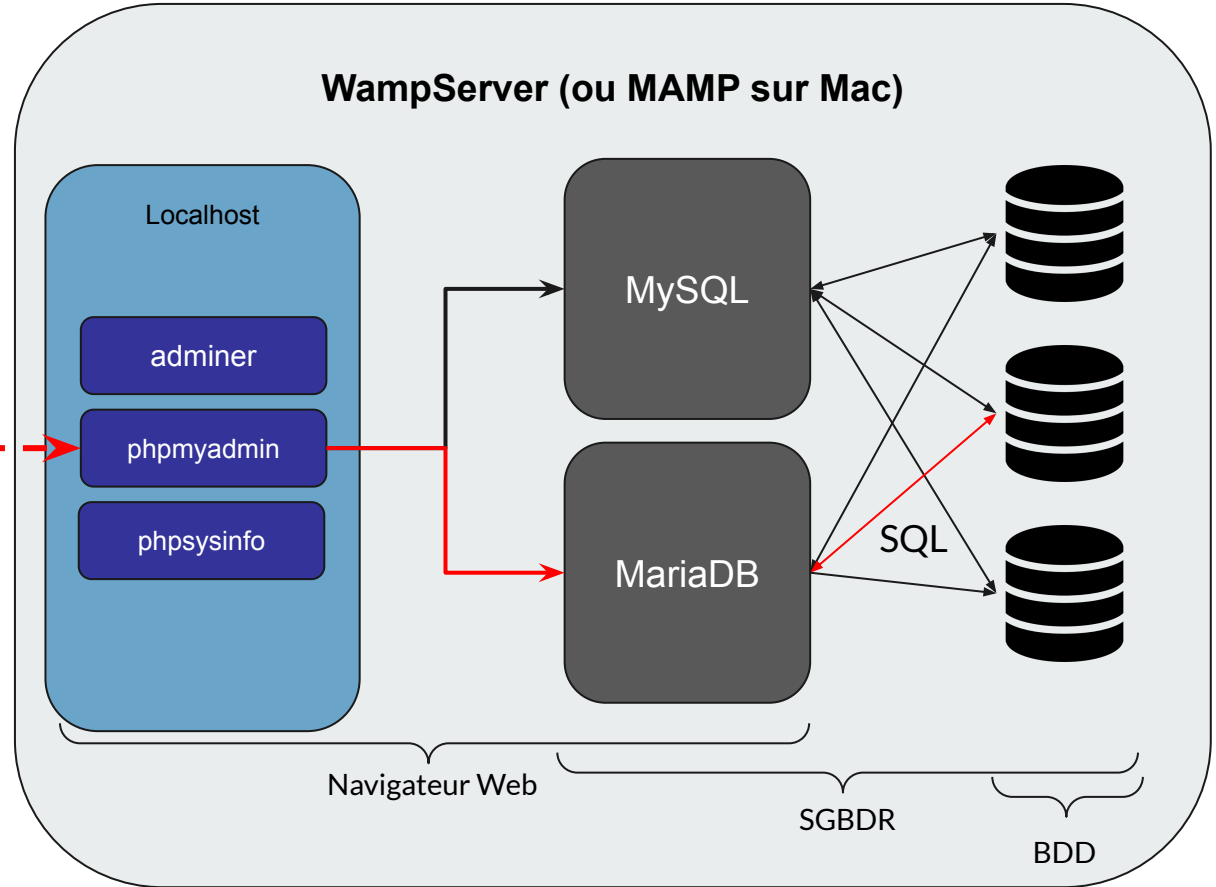


# Architecture



User

Se connecte





# Installation de WAMP SERVER

Rendez-vous sur la page : <https://www.wampserver.com/>, téléchargez le logiciel et installez-le.



## TÉLÉCHARGER WAMPSERVER 64 BITS (X64) 3.2.6

WampServer est disponible gratuitement (sous licence GPL). Vous pouvez remplir ce formulaire qui nous permettra de vous faire parvenir actualités formation d'Alter vway, société editrice, ainsi que toutes les informations liées aux évolutions de WampServer. Si vous ne le souhaitez pas, vous pourrez [passer au téléchargement direct](#).

Prénom :

Nom :

Société :

Email (\*) :

Téléphone :

Pays :

Fonction (\*) :

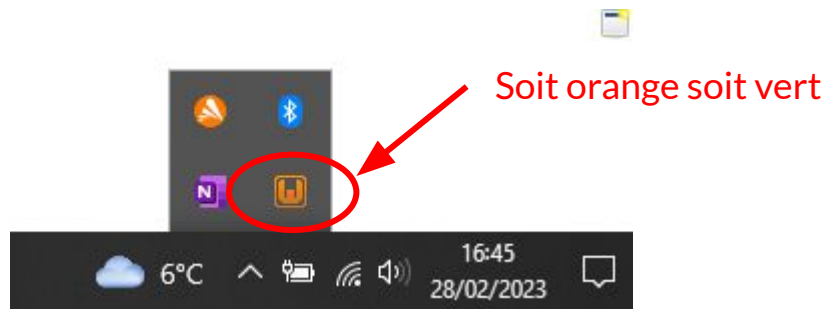
Vous avez des questions, des remarques, des commentaires ?

Votre utilisation de Wampserver :

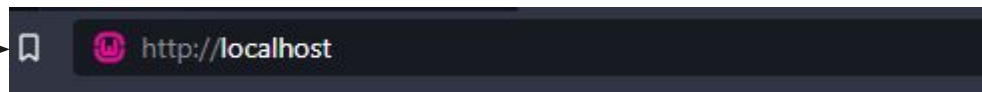
- ☐ Utilisation pour une application interne
- ☐ Utilisation pour développer en préproduction
- ☐ Ne prévoit pas d'utiliser WAMPSERVER

☐ Je souhaite recevoir des informations de WampServer

ENVOYER



Dans votre navigateur préféré





# Wampserver

Apache 2.4 - MySQL 5 & 8 - MariaDB 10 - PHP 5 & 7

Version 3.2.3 - 64bit

french

classic

## Configuration Serveur

Version Apache : 2.4.46 - [Documentation](#)

Server Software : Apache/2.4.46 (Win64) PHP/7.3.21 - Port défini pour Apache : 80

Version de PHP : 7.3.21 - [Documentation](#)

### Extensions Chargées :

- apache2handler
- Core
- fileinfo
- hash
- ldap
- openssl
- Phar
- soap
- tokenizer
- xmlrpc
- zlib
- bcmath
- ctype
- filter
- iconv
- libxml
- pcre
- readline
- sockets
- wddx
- xmlwriter
- bz2
- date
- gd
- imap
- mbstring
- PDO
- Reflection
- SPL
- xdebug
- xsl
- calendar
- dom
- gettext
- intl
- mysqli
- pdo\_mysql
- session
- sqlite3
- xml
- Zend OPcache
- com\_dotnet
- exif
- gmp
- json
- mysqlnd
- pdo\_sqlite
- SimpleXML
- standard
- xmlreader
- zip

Version de MySQL : 5.7.31 - Port défini pour MySQL : 3306 - SGBD par défaut - [Documentation MySQL](#)

Version de MariaDB : 10.4.13 - Port défini pour MariaDB : 3307 - [Documentation MariaDB](#) - [MySQL](#) - [MariaDB](#)

## Outils

- [phpinfo\(\)](#)
- [phpmyadmin](#)
- [Ajouter un Virtual Host](#)

## Vos Projets

Aucun projet.  
Pour en ajouter un nouveau, créez simplement un répertoire dans 'www'.

## Vos Alias

- [adminer](#)
- [phpmyadmin](#)
- [phpsysinfo](#)

## Vos VirtualHost

- [localhost](#)



Bienvenue dans phpMyAdmin

Lange - *Language*

Français - French

Connexion ⓘ

Utilisateur :

Mot de passe :

Choix du serveur :

MySQL

Exécuter



Bienvenue dans phpMyAdmin

Lange - *Language*

Français - French

Connexion ⓘ

Utilisateur :

root

Mot de passe :

Choix du serveur :

MariaDB

Exécuter



Serveur courant :

MariaDB

Récentes Préférées



- Nouvelle base de données
- base1
- base2
- information\_schema
- mysql
- performance\_schema
- test
- test2

## Paramètres généraux

Modifier le mot de passe

Interclassement pour la connexion au serveur : utf8mb4\_unicode\_ci

Plus de paramètres

## Paramètres d'affichage

Langue - Language Français - French

Thème : pmahomme



Serveur courant :

MariaDB

Récentes Préférées

Nouvelle base de données

base1

base2

information\_schema

mysql

performance\_schema

test

test2

Bases de données

SQL

État

Comptes utilisateurs

Exporter

Importer

Paramètres

## Paramètres généraux

Modifier le mot de passe

Interclassement pour la connexion au serveur : utf8mb4\_unicode\_ci

Plus de paramètres

## Paramètres d'affichage

Langue - Language Français - French

Thème : pmahomme







# Bases de données

Création d'une base de données ?

TP\_esgf

latin1\_swedish\_ci


Créer

	Base de données ▲	Interclassement	Action
<input type="checkbox"/>	base1	latin1_swedish_ci	 Vérifier les privilèges
<input type="checkbox"/>	base2	latin1_swedish_ci	 Vérifier les privilèges
<input type="checkbox"/>	information_schema	utf8_general_ci	 Vérifier les privilèges
<input type="checkbox"/>	mysql	latin1_swedish_ci	 Vérifier les privilèges
<input type="checkbox"/>	performance_schema	utf8_general_ci	 Vérifier les privilèges
<input type="checkbox"/>	test	latin1_swedish_ci	 Vérifier les privilèges
<input type="checkbox"/>	test2	latin1_swedish_ci	 Vérifier les privilèges
Total : 7			



☐ Tout cocher

Avec la sélection :

 Supprimer





Serveur courant :

MariaDB

Récentes

Préférées

- Nouvelle base de données
- + base1
- + base2
- + information\_schema
- + mysql
- + performance\_schema
- + test
- + test2
- tp\_esgf

Serveur: MariaDB-3307 » Base de données: tp\_esgf



Structure



SQL



Rechercher



Requête



Exporter



Aucune table n'a été trouvée dans cette base de données.



Nouvelle table

Nom:

Nombre de colonnes:

4

phpMyAdmin

Serveur courant : MariaDB

Récentes Préférées

- Nouvelle base de données
- bdcommandes
- information\_schema
- mysql
- performance\_schema
- tp\_esgf
  - Nouvelle table
  - commandes
  - fournisseurs
  - produits

Serveur: MariaDB:3307 » Base de données: tp\_esgf

Structure SQL Rechercher Requête Exporter Importer Opérations Privileges Procédures stockées Événements Déclencheurs

Exécuter une ou des requêtes SQL sur la base de données « tp\_esgf »:

```
1 DROP DATABASE IF EXISTS BDCOMMANDES; CREATE DATABASE BDCOMMANDES ;
2 USE BDCOMMANDES ;
3 DROP TABLE IF EXISTS Fournisseurs ; DROP TABLE IF EXISTS Produits ; DROP TABLE IF EXISTS Commandes ;
4 CREATE TABLE Fournisseurs (
5   fno Numeric(6) NOT NULL primary key ,
6   nom VARCHAR(25) NOT NULL ,
7   adresse VARCHAR(25) ,
8   ville VARCHAR(25) NOT NULL
9 ) ;
10
11 CREATE TABLE Produits (
12   pno Numeric(6) NOT NULL primary key ,
```

**Copier/coller le script SQL**

Effacer Format Récupérer la requête auto-sauvegardée

☐ Lier les paramètres

[ Délimiteur ; ] ☐ Afficher à nouveau la requête après exécution ☐ Conserver la boîte de requêtes ☐ ROLLBACK à la fin ☒ Activer la vérification des clés étrangères

Exécuter

<https://github.com/Le-Minh-Phuc/Relational-DB/blob/main/script1-db>



Serveur courant :

MariaDB

Récentes Préférées

990

- Nouvelle base de données
- bdcommandes
- information\_schema
- mysql
- performance\_schema
- tp\_esgf
  - Nouvelle table
  - commandes
  - fournisseurs
  - produits

Serveur: MariaDB:3307 » Base de données: tp\_esgf

Structure SQL Rechercher Requête Exporter Importer Opérations Privileges Procédures stockées

Afficher la zone SQL

✓ MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0011 seconde(s).)

DROP DATABASE IF EXISTS BDCOMMANDES

✓ MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0007 seconde(s).)

CREATE DATABASE BDCOMMANDES

✓ MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0003 seconde(s).)

USE BDCOMMANDES

✓ MySQL a retourné un résultat vide (c'est à dire aucune ligne). (traitement en 0,0009 seconde(s).)

DROP TABLE IF EXISTS Fournisseurs

✓ Affichage des lignes 0 - 9 (total de 10, traitement en 0,0003 seconde(s).)

```
SELECT * FROM `commandes`
```

☐ Tout afficher

Nombre de lignes : 25 ▼

Filtrer les lignes:

Trier pa

+ Options

						cno	fno	pno	qute	
<input type="checkbox"/>		Éditer		Copier		Supprimer	1001	17	103	10
<input type="checkbox"/>		Éditer		Copier		Supprimer	1003	15	103	2
<input type="checkbox"/>		Éditer		Copier		Supprimer	1005	17	102	1
<input type="checkbox"/>		Éditer		Copier		Supprimer	1007	15	108	1
<input type="checkbox"/>		Éditer		Copier		Supprimer	1011	19	107	12
<input type="checkbox"/>		Éditer		Copier		Supprimer	1013	13	107	5
<input type="checkbox"/>		Éditer		Copier		Supprimer	1017	19	105	3
<input type="checkbox"/>		Éditer		Copier		Supprimer	1019	14	103	10
<input type="checkbox"/>		Éditer		Copier		Supprimer	1023	10	102	8
<input type="checkbox"/>		Éditer		Copier		Supprimer	1029	17	108	15



☐ Tout cocher

Avec la sélection :

 Éditer

 Copier

 Supprimer

 Exporter

☐ Tout afficher

Nombre de lignes : 25 ▼

Filtrer les lignes:

Trier pa



# Présentation d'une requête SQL

Une requête SQL pour une manipulation de donnée s'écrit d'une certaine manière :

[Action] [Element] FROM [Table] WHERE [Condition]



## Exemples

“SELECT \* FROM commandes” - Sélectionne toutes les entrées dans la table commandes

“SELECT \* FROM commandes WHERE fno = 15” - Sélectionne toutes les entrées dont l’attribut “fno” est égal à 15.



## A utilisier

<https://sql.sh/>



## Cours 6

# Les Manipulations de données





# SELECT

L'utilisation la plus courante du SQL consiste à lire des données issues de la base de données. C'est ce qu'on appelle l'interrogation.

Son utilisation se fait de la manière suivante :

```
SELECT nom_du_champ FROM nom_du_tableau
```

***SELECT** \* FROM commandes*



# DISTINCT

DISTINCT se combine avec SELECT pour éviter des redondances dans les résultats il faut simplement ajouter DISTINCT après le mot SELECT.

“Sélectionner les différents fno dans la table commandes”

```
SELECT DISTINCT fno FROM commandes
```



# WHERE

La commande **WHERE** dans une requête SQL permet d'extraire les lignes d'une base de données qui respectent une condition. Cela permet d'obtenir uniquement les informations désirées.

“Sélectionner les commandes dont le pno est supérieur à 104”

```
SELECT * FROM commandes WHERE pno > 104
```



## AND & OR

Les opérateurs logiques AND et OR peuvent être utilisées dans la commande WHERE pour combiner des conditions.

Les opérateurs sont à ajoutés dans la condition WHERE. Ils peuvent être combinés à l'infini pour filtrer les données comme souhaités.

```
SELECT * FROM commandes WHERE fno>15 OR fno<12
```

```
SELECT * FROM commandes WHERE fno = 15 AND pno =12
```



# IN

L'opérateur logique IN s'utilise avec la commande WHERE pour vérifier si une colonne est égale à une des valeurs comprise dans set de valeurs déterminés. C'est une méthode simple pour vérifier si une colonne est égale à une valeur OU une autre valeur OU une autre valeur et ainsi de suite, sans avoir à utiliser de multiple fois l'opérateur OR.



# BETWEEN

L'opérateur BETWEEN est utilisé dans une requête SQL pour sélectionner un intervalle de données dans une requête utilisant WHERE. L'intervalle peut être constitué de chaînes de caractères, de nombres ou de dates. L'exemple le plus concret consiste par exemple à récupérer uniquement les enregistrements entre 2 dates définies.

```
SELECT * FROM commandes WHERE pno > 12 AND pno < 15
```

```
⇔ SELECT * FROM commandes WHERE pno BETWEEN 12 AND 15
```



# LIKE

L'opérateur LIKE est utilisé dans la clause WHERE des requêtes SQL. Ce mot-clé permet d'effectuer une recherche sur un modèle particulier. Il est par exemple possible de rechercher les enregistrements dont la valeur d'une colonne commence par telle ou telle lettre. Les modèles de recherches sont multiples.

- **LIKE '%a'** : Le caractère “%” est un caractère joker qui remplace tous les autres caractères. Ainsi, ce modèle **permet de rechercher toutes les chaînes de caractères qui se terminent par un “a”**.
- **LIKE 'a%'** : Ce modèle permet de rechercher toutes les chaînes de caractères qui **commencent par un “a”**.
- **LIKE '%a%'** : Ce modèle est utilisé pour rechercher **tous les enregistrements qui utilisent le caractère “a”**.
- **LIKE 'pa%on'** : Ce modèle permet de rechercher les chaînes qui **commencent par “pa” et qui se terminent par “on”** (“pantalon”, “pardon”...)
- **LIKE 'a\_c'** : peu utilisé, le caractère “\_” (underscore) **peut être remplacé par n'importe quel caractère, mais un seul caractère uniquement** (alors que le symbole pourcentage “%” peut être remplacé par un nombre incalculable de caractères). Ainsi, ce modèle permet de retourner les lignes “aac”, “abc” ou même “azc”.



## IS NULL / IS NOT NULL

L'opérateur IS permet de filtrer les résultats qui contiennent la valeur NULL. Cet opérateur est indispensable car la valeur NULL est une valeur inconnue et ne peut par conséquent pas être filtrée par les opérateurs de comparaison (cf. égal, inférieur, supérieur ou différent).





## Cours 7

# Les Créations et modifications de table



# CREATE DATABASE / CREATE TABLE

La création d'une base de données en SQL est possible en ligne de commande grâce à la commande CREATE DATABASE. Même si les systèmes de gestion de base de données (SGBD) sont souvent utilisés pour créer une base, il convient de connaître la commande à utiliser, qui est très simple. Pour créer une table dans une base de données, on utilise CREATE TABLE.

```
CREATE DATABASE IF NOT EXISTS esgf_db
```

```
CREATE TABLE IF NOT EXISTS commandes
```



# ALTER TABLE

La commande ALTER TABLE permet de modifier une table existante. Idéal pour ajouter une colonne, supprimer une colonne ou modifier une colonne existante, par exemple pour changer le type.

ALTER TABLE commandes

ADD nom\_commande CHAR

```
ALTER TABLE nom_table
```

```
ADD nom_colonne type_donnees
```



# UPDATE

La commande UPDATE permet d'effectuer des modifications sur des lignes existantes. Très souvent cette commande est utilisée avec WHERE pour spécifier sur quelles lignes doivent porter la ou les modifications.

```
UPDATE table
```

```
SET nom_colonne_1 = 'nouvelle valeur'
```

```
WHERE condition
```



# DELETE

La commande DELETE en SQL permet de supprimer des lignes dans une table. En utilisant cette commande associé à WHERE il est possible de sélectionner les lignes concernées qui seront supprimées.

```
DELETE FROM `table`
```

```
WHERE condition
```



# TRUNCATE TABLE

En SQL, la commande **TRUNCATE TABLE** permet de supprimer toutes les données d'une table sans supprimer la table en elle-même. En d'autres mots, cela permet de purger la table. Cette instruction diffère de la commande **DROP** qui à pour but de supprimer les données ainsi que la table qui les contient.



## GROUP BY

La commande GROUP BY est utilisée en SQL pour grouper plusieurs résultats et utiliser une fonction de totaux sur un groupe de résultat. Sur une table qui contient toutes les ventes d'un magasin, il est par exemple possible de liste regrouper les ventes par clients identiques et d'obtenir le coût total des achats pour chaque client.

```
SELECT colonne1, fonction(colonne2)
```

```
FROM table
```

```
GROUP BY colonne1
```



## ORDER BY

La commande ORDER BY permet de trier les lignes dans un résultat d'une requête SQL. Il est possible de trier les données sur une ou plusieurs colonnes, par ordre ascendant ou descendant.

```
SELECT colonne1, colonne2
```

```
FROM table
```

```
ORDER BY colonne1
```





# AS

Dans le langage SQL il est possible d'utiliser des **alias** pour renommer temporairement une colonne ou une table dans une requête. Cette astuce est particulièrement utile pour faciliter la lecture des requêtes.

```
SELECT * FROM commandes AS com WHERE com.pno = 12
```



## Cours 8

# Requêtes SQL



## TD/TP3

Rendez vous sur le fichier “TD TP BDDR.pdf” et commencez la section 3 - SQL1



## Cours 9

# Requêtes SQL



## TD/TP3

Rendez vous sur le fichier “TD TP BDDR.pdf” et continuez la section 3 - SQL1



## Cours 10

# Les fonctions d'agrégation



# HAVING

La condition HAVING en SQL est presque similaire à WHERE à la seule différence que HAVING permet de filtrer en utilisant des fonctions telles que SUM(), COUNT(), AVG(), MIN() ou MAX().

```
SELECT colonne1, SUM(colonne2)
```

```
FROM nom_table
```

```
GROUP BY colonne1
```

```
    HAVING fonction(colonne2) operateur valeur
```



# AVG

La fonction d'agrégation `AVG()` dans le langage SQL permet de calculer une valeur moyenne sur un ensemble d'enregistrement de type numérique et non nul.





# COUNT

En SQL, la fonction d'agrégation COUNT() permet de compter le nombre d'enregistrement dans une table. Connaître le nombre de lignes dans une table est très pratique dans de nombreux cas, par exemple pour savoir combien d'utilisateurs sont présents dans une table ou pour connaître le nombre de commentaires sur un article.



# MAX

Dans le langage SQL, la fonction d'agrégation MAX() permet de retourner la valeur maximale d'une colonne dans un set d'enregistrement. La fonction peut s'appliquée à des données numériques ou alphanumériques. Il est par exemple possible de rechercher le produit le plus cher dans une table d'une boutique en ligne.



# MIN

La fonction d'agrégation MIN() de SQL permet de retourner la plus petite valeur d'une colonne sélectionnée. Cette fonction s'applique aussi bien à des données numériques qu'à des données alphanumériques.



# SUM

Dans le langage SQL, la fonction d'agrégation SUM() permet de calculer la somme totale d'une colonne contenant des valeurs numériques. Cette fonction ne fonctionne que sur des colonnes de types numériques (INT, FLOAT ...) et n'additionne pas les valeurs NULL.



## Cours 11

# Requêtes SQL



## TD/TP3

Rendez vous sur le fichier “TD TP BDDR.pdf” et commencez la section 3 - SQL2



## Cours 12

### Évaluation sur table



## Cours 13

# Jointures





# INNER JOIN

Dans le langage SQL la commande INNER JOIN, aussi appelée EQUIJOIN, est un type de jointures très communes pour lier plusieurs tables entre-elles. Cette commande retourne les enregistrements lorsqu'il y a au moins une ligne dans chaque colonne qui correspond à la condition.

```
SELECT *
```

```
FROM table1
```

```
INNER JOIN table2 ON table1.id = table2.fk_id
```



# CROSS JOIN

Dans le langage SQL, la commande CROSS JOIN est un type de jointure sur 2 tables SQL qui permet de retourner le produit cartésien. Autrement dit, cela permet de retourner chaque ligne d'une table avec chaque ligne d'une autre table. Ainsi effectuer le produit cartésien d'une table A qui contient 30 résultats avec une table B de 40 résultats va produire 1200 résultats ( $30 \times 40 = 1200$ ). En général la commande CROSS JOIN est combinée avec la commande WHERE pour filtrer les résultats qui respectent certaines conditions.

```
SELECT *  
  
FROM table1  
  
    CROSS JOIN table2
```



# LEFT JOIN

Dans le langage SQL, la commande LEFT JOIN (aussi appelée LEFT OUTER JOIN) est un type de jointure entre 2 tables. Cela permet de lister tous les résultats de la table de gauche (left = gauche) même s'il n'y a pas de correspondance dans la deuxième tables.

```
SELECT *
```

```
FROM table1
```

```
LEFT JOIN table2 ON table1.id = table2.fk_id
```



# RIGHT JOIN

En SQL, la commande RIGHT JOIN (ou RIGHT OUTER JOIN) est un type de jointure entre 2 tables qui permet de retourner tous les enregistrements de la table de droite (right = droite) même s'il n'y a pas de correspondance avec la table de gauche. S'il y a un enregistrement de la table de droite qui ne trouve pas de correspondance dans la table de gauche, alors les colonnes de la table de gauche auront NULL pour valeur.

```
SELECT *
```

```
FROM table1
```

```
RIGHT JOIN table2 ON table1.id = table2.fk_id
```



# FULL JOIN

Dans le langage SQL, la commande FULL JOIN (ou FULL OUTER JOIN) permet de faire une jointure entre 2 tables. L'utilisation de cette commande permet de combiner les résultats des 2 tables, les associer entre eux grâce à une condition et remplir avec des valeurs NULL si la condition n'est pas respectée.

```
SELECT *  
  
FROM table1  
  
FULL JOIN table2 ON table1.id = table2.fk_id
```



# SELF JOIN

En SQL, un SELF JOIN correspond à une jointure d'une table avec elle-même. Ce type de requête n'est pas si commun mais très pratique dans le cas où une table lie des informations avec des enregistrements de la même table.

```
SELECT `t1`.`nom_colonne1`, `t1`.`nom_colonne2`, `t2`.`nom_colonne1`, `t2`.`nom_colonne2`  
  
FROM `table` as `t1`  
  
LEFT OUTER JOIN `table` as `t2` ON `t2`.`fk_id` = `t1`.`id`
```



# NATURAL JOIN

Dans le langage SQL, la commande NATURAL JOIN permet de faire une jointure naturelle entre 2 tables. Cette jointure s'effectue à la condition qu'il y ai des colonnes du même nom et de même type dans les 2 tables. Le résultat d'une jointure naturelle est la création d'un tableau avec autant de lignes qu'il y a de paires correspondant à l'association des colonnes de même nom.

```
SELECT *
```

```
FROM table1
```

```
NATURAL JOIN table2
```



## Cours 14

# Requêtes SQL





## TD/TP3

Rendez vous sur le fichier “TD TP BDDR.pdf” et continuez la section 3 - SQL2 - Jointures.