







SQL

Les fondamentaux du langage

Le langage SQL : Définition






SQL (Structured Query Language ou langage de requêtes structurées) est un langage de manipulation de données :

-  **Normalisé** : Des règles universelles d'écriture ont été établies par un groupe de normalisation composés d'experts (A.N.S.I → American National Standard Institute).
-  **Non propriétaire** : Aucune société ne possède ni ne contrôle le langage
-  **Permet de gérer des bases de données relationnelles** :
 -  **Organiser les données**
Créer, modifier, supprimer des tables dans les bases de données.
 -  **Manipuler les données**
Ajouter, modifier, lire, supprimer les données dans les tables.
 -  **Contrôler les données**
Contrôle des accès aux données en fonction des utilisateurs.

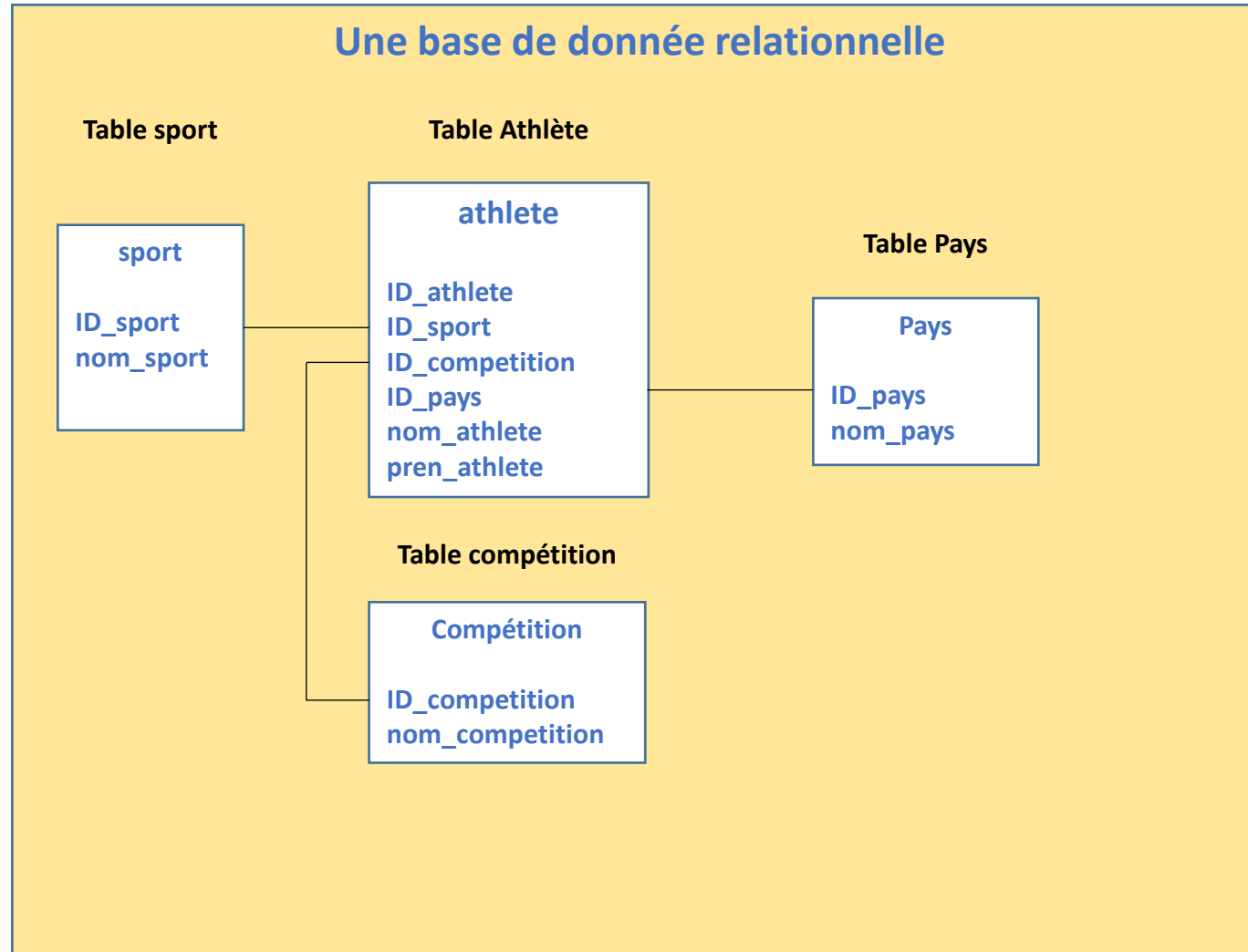
Base de donnée relationnelle : Définition

Une base de donnée relationnelle est une entité structurée, permettant le stockage, en grande quantité, d'informations décomposées et organisées sous forme de tables reliées entre elles.

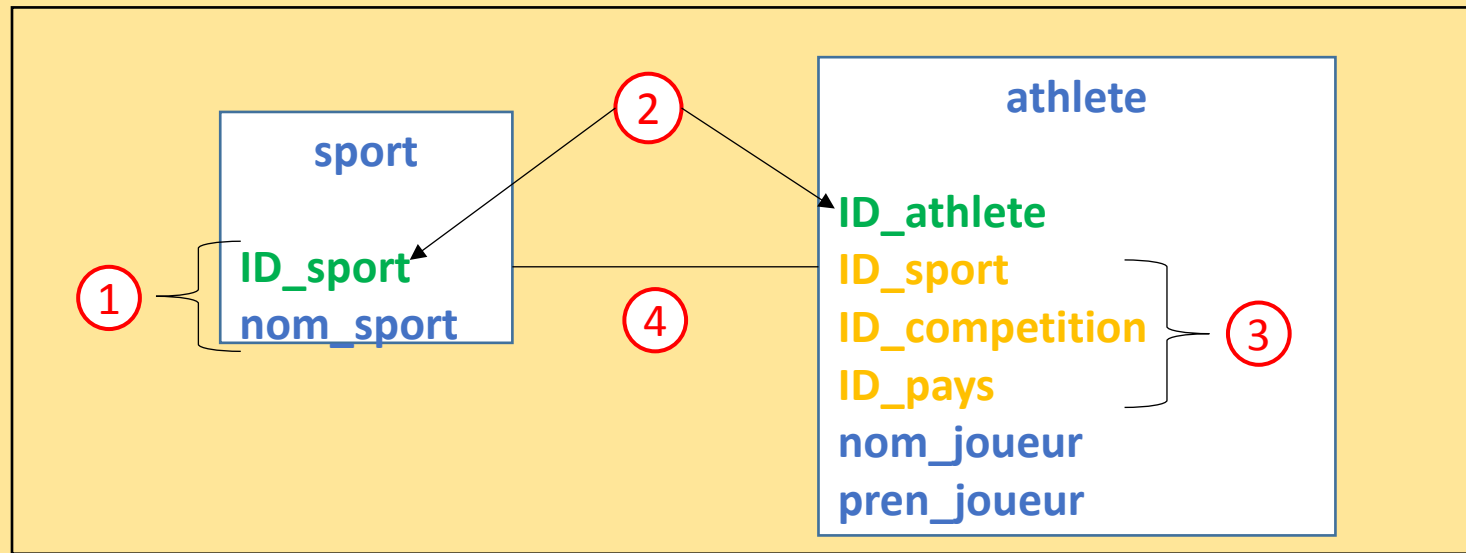
Par rapport à un tableur comme Excel, une base de donnée permet :

-  De gérer de gros volumes de données
-  De rendre plus fiable la cohérence des données
-  D'afficher les données comme nous le souhaitons (formulaires, listes, tableaux, etc....)
-  D'affiner les recherches d'informations grâce aux relations entre les tables
-  D'enregistrer des informations sous différents formats (texte, images, vidéos, sons)

Base de donnée relationnelle : Schéma



Description d'une relation entre deux tables



① Champs ou attributs de la table

④ Relation
Une relation est un lien logique entre deux tables souvent représentée par le couple clé primaire/clé étrangère ou par une table de jointure.

② Clés primaires

Une clé primaire est ce que l'on appelle une contrainte d'unicité et permet d'identifier de façon unique un enregistrement dans une table de base de donnée.

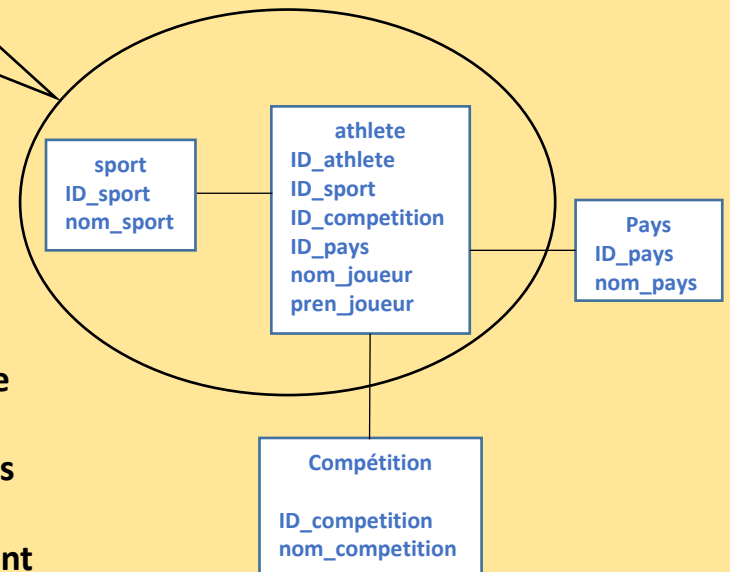
③ Clés étrangères

Une clé étrangère est un champ dans une table (table référençant) qui référence un champ ou un ensemble de champs dans une autre table (table référencée).

Cela signifie que pour un enregistrement de la table référençant, il existe un et un seul enregistrement dans la table référencée.

La clé étrangère dans la table référençant est la clé primaire de la table référencée et doit porter exactement le même nom.

Il peut y avoir plusieurs clés étrangères dans une table référençant.

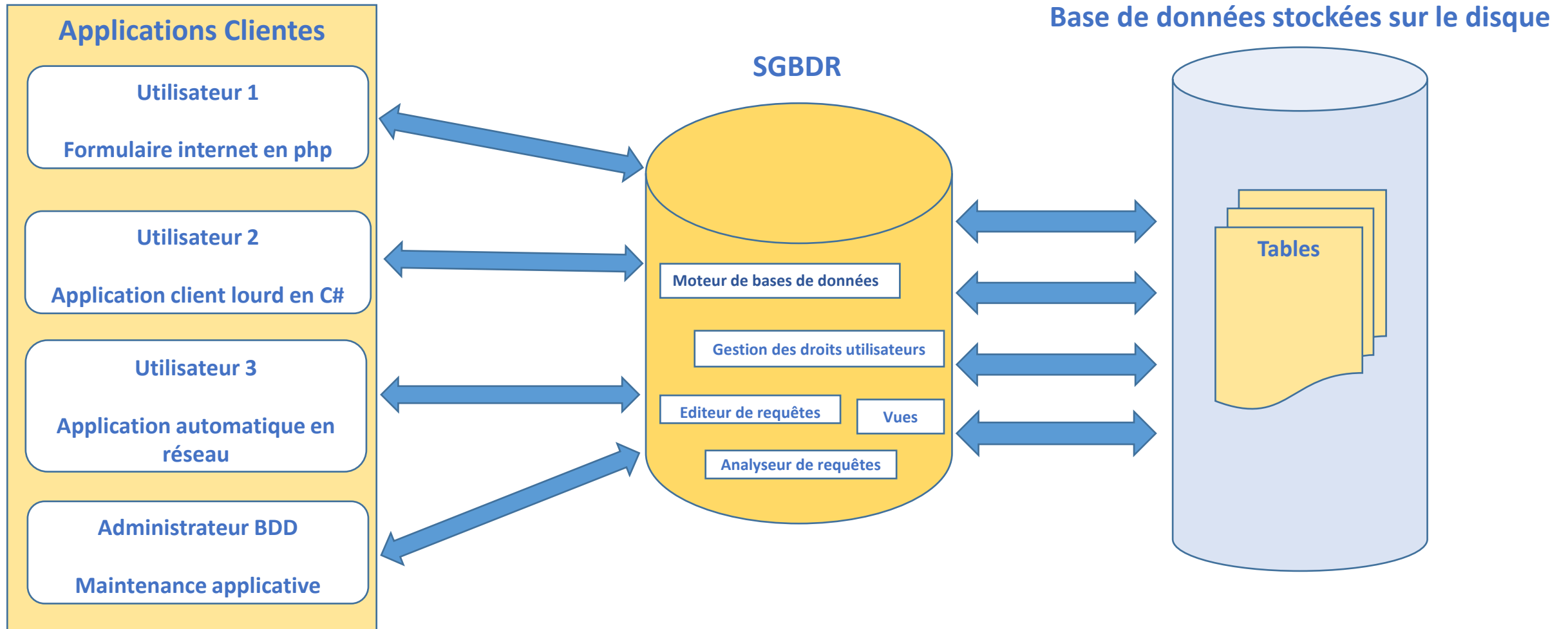


Système de Gestion de Bases de Données (SGBDR)

Un système de gestion de base de données relationnelles (SGBDR) est un ensemble de logiciels qui permet de gérer des bases de données ainsi que le stockage et l'accès aux informations contenues dans celles-ci.

- ➡ Contient plusieurs bases de données
- ➡ Permet le partage des données entre utilisateurs au même moment de façon transparente
- ➡ Assure la non redondance des informations
- ➡ Permet un accès rapide aux données
- ➡ Assure l'intégrité des données en cas de panne
- ➡ Gestion centralisée des données
- ➡ Les plus connus : SQL Server, Oracle, MySQL, Sybase, Access

Schéma d'un SGBDR



WAMP SERVER / phpMyAdmin

Wamp est l'acronyme de :

Windows	: Le système d'exploitation de Microsoft
Apache	: Serveur HTTP (ou serveur web) sur lequel sont hébergés des sites Internet
MySQL	: Système de gestion de bases de données open-source
PHP	: Langage de programmation utilisé pour la création de pages web dynamiques

PhpMyAdmin

phpMyAdmin est une interface accessible via un navigateur web pour gérer le SGBD MySQL.
Il a été écrit en langage PHP et utilise le serveur Apache.

WampServer

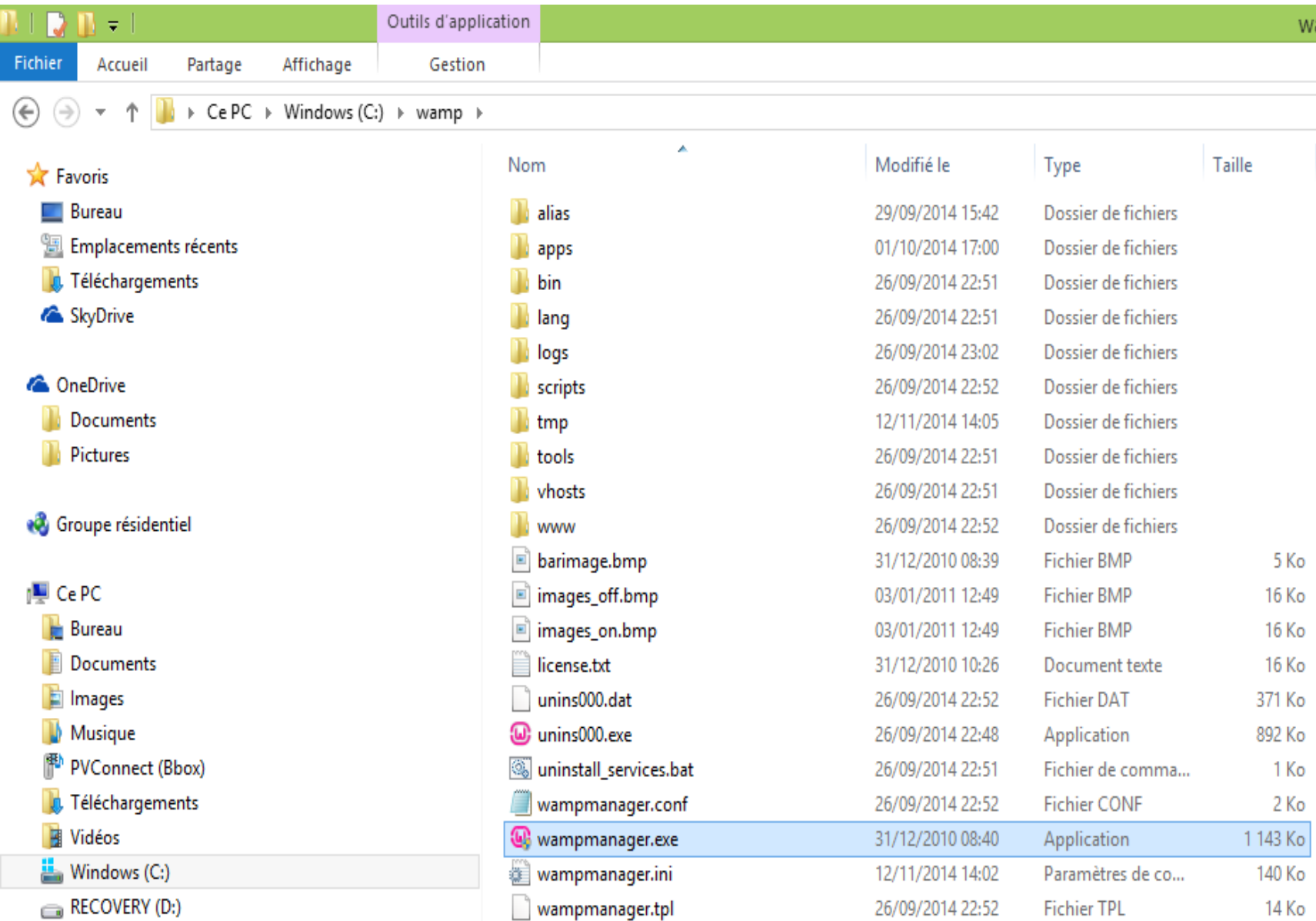
Environnement de développement qui permet de développer des sites Internet dynamiques grâce au serveur Apache, au langage PHP et avec la possibilité d'utiliser une base de donnée sous MYSQL gérée avec phpMyAdmin.

WampServer est téléchargeable à l'adresse : <http://www.wampserver.com>.

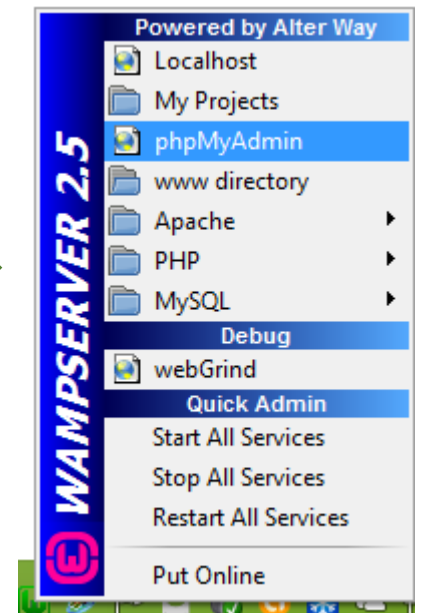
Il est conseillé d'installer WampServer à la racine du disque C:\ (ex: C:\wamp).


Une fois installé, cliquer sur le fichier exécutable C:\wamp\wampmanager.exe pour lancer WampServer.

phpMyAdmin : Ouvrir l'interface



(c) Philippe Maroudy - 2014



- 1) Ouvrez votre explorateur Windows
- 2) Rendez-vous dans le répertoire wamp
- 3) Double-cliquez sur le fichier wampmanager.exe
- 4) Cliquez sur l'icône  de la barre des tâches
- 5) Choisissez phpMyAdmin

phpMyAdmin : Création d'une base

phpMyAdmin

Recentes Préférences

Nouvelle base de données (1)

essai
information_schema (2)
mysql
performance_schema
test

Sever: mysql wampserver

Bases de données SQL État Utilisateurs Exporter Importer Paramètres Réplication Variables Jeux de caractères Moteurs

Bases de données

Créer une base de données (2)

Nom de base de données Interclassement (3) Créer

Note: L'activation des statistiques peut causer un trafic important entre le serveur web et le serveur MySQL.

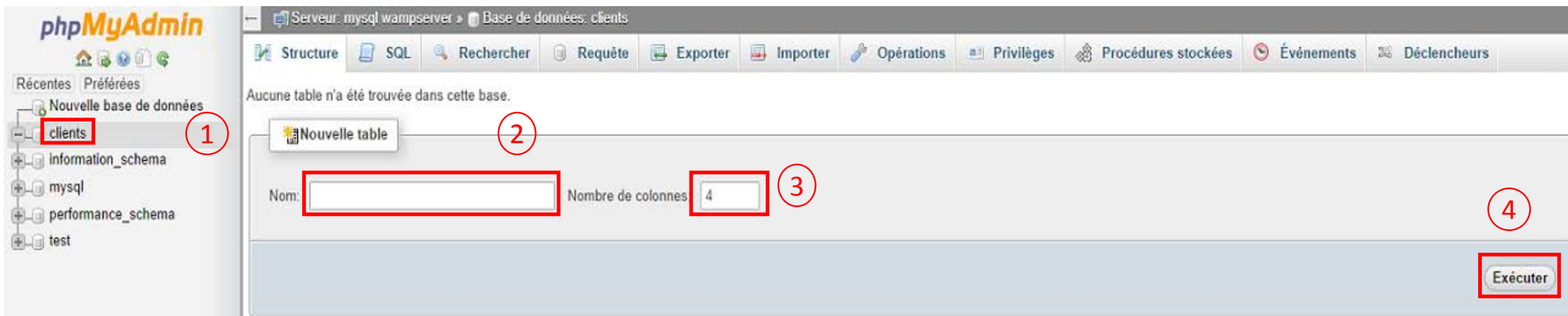
Base de données	Interclassement
<input type="checkbox"/> essai	latin1_swedish_ci Vérifier les privilèges
<input type="checkbox"/> information_schema	utf8_general_ci Vérifier les privilèges
<input type="checkbox"/> mysql	latin1_swedish_ci Vérifier les privilèges
<input type="checkbox"/> performance_schema	utf8_general_ci Vérifier les privilèges
<input type="checkbox"/> test	latin1_swedish_ci Vérifier les privilèges
Total: 5	latin1_swedish_ci

↑ ☐ Tout cocher Pour la sélection : Supprimer

- Activer les statistiques

- 1 Cliquez sur le bouton 'Nouvelle base de données'
- 2 Donnez un nom à votre base de données
- 3 Cliquez sur le bouton 'Créer'

phpMyAdmin : Création d'une table (1)



- ① Cliquez sur le nom de la base nouvellement créée
- ② Saisissez un nom à la table
- ③ Saisissez le nombre de colonne souhaitée
- ④ Cliquez sur le bouton 'Exécuter' pour créer la table

phpMyAdmin : Création d'une table (2)

← Serveur: mysql wampserver » Base de données: clients » Table: clients

Afficher Structure SQL Rechercher Insérer Exporter Importer Privileges Opérations Déclencheurs

Nom de la table: clients Ajouter 1 colonne(s) Exécuter

Structure ?

Nom 1	Type 2	Taille/Valeurs* 3	Défaut 4	Interclassement 5	Attributs 6	Null 7	Index 8	A_I 9	Commentaires 10
id_client	INT	5	Aucune		UNSIGNED	<input type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>	
nom_client	VARCHAR	50	Aucune			<input type="checkbox"/>	---	<input type="checkbox"/>	
prenom_client	VARCHAR	50	Aucune			<input type="checkbox"/>	---	<input type="checkbox"/>	
adresse_client	VARCHAR	70	Aucune			<input type="checkbox"/>	---	<input type="checkbox"/>	
code_postal_client	INT	5	Aucune			<input type="checkbox"/>	---	<input type="checkbox"/>	
ville_client	VARCHAR	50	Aucune			<input type="checkbox"/>	---	<input type="checkbox"/>	

Commentaires sur la table :

Moteur de stockage : InnoDB Interclassement :

Définition de PARTITION :

11

(c) Philippe Maroudy - 2014

12 Sauvegarder

phpMyAdmin : Création d'une table (3)

- ① **Nom** : Cette colonne permet de renseigner le nom des champs de la table.
- ② **Type** : Cette colonne permet de renseigner le type des champs de la table.
- ③ **Taille/Valeurs** : Cette colonne permet de renseigner la taille maximale que vous souhaitez pour chaque champ de la table. Utile pour les données de type varchar.
- ④ **Défaut** : Cette colonne permet de saisir des valeurs par défaut lorsque vous créer un enregistrement dans la table.
- ⑤ **Interclassement** : Cette colonne permet de renseigner le jeu de caractère lors du stockage de l'information et pour que l'ordinateur affiche correctement certains caractères propres à chaque langue.
- ⑥ **Attributs** : Cette colonne permet de spécifier des valeurs aux champs de la table. La valeur Unsigned est notamment utile pour la création de champs ID qui sont strictement positifs.
- ⑦ **Null** : Cette colonne permet de spécifier si le champ peut être enregistré sans valeur.
- ⑧ **Index** : Cette colonne est utile pour indexer un champ et pouvoir effectuer des recherches via ce champ sur la table. Utile pour les champs ID.
- ⑨ **A_I** : Cette colonne est très utile puisqu'elle permet d'auto-incrémenter un champ à chaque nouvelle entrée dans la table. Utile pour les champs ID.
- ⑩ **Commentaires** : Cette colonne permet de saisir des commentaires sur le champ de la table.
- ⑪ Le bouton '**Sauvegarder**' permet de sauvegarder les champs nouvellement créés dans la table.

Les commandes SQL : CREATE DATABASE

Objectif

La commande CREATE DATABASE permet de créer une base de donnée.

Syntaxe

CREATE DATABASE nom_de_la_base ;

OU

CREATE DATABASE IF NOT EXISTS nom_de_la_base; ①

Exemple

Créer une base clients

CREATE DATABASE IF NOT EXISTS DB_TECH;

①

Dans le langage SQL standard CREATE DATABASE n'existe pas. Il est recommandé de vérifier, pour chaque SGBD, la syntaxe exacte et les options de cette commande.

Avec MySQL, la première syntaxe retourne une erreur si la base existe. Il est recommandé d'utiliser la seconde syntaxe avec ce SGBD.

Les commandes SQL : DROP DATABASE

Objectif

La commande DROP DATABASE permet de supprimer une base de donnée.

Syntaxe

DROP DATABASE nom_de_la_base ;

OU

DROP DATABASE IF EXISTS nom_de_la_base;

Exemple

Supprimer la base clients

DROP DATABASE IF EXISTS clients;

Les commandes SQL : CREATE TABLE

Objectif

La commande CREATE TABLE permet de créer une table dans une base de donnée.

Syntaxe

```
CREATE TABLE nom_table  
(  
    nom_colonne1 type_données1,  
    nom_colonne2 type_données2,  
    nom_colonne3 type_données3,  
    ....  
    nom_colonneN type_donnéesN  
);
```

Exemple

Créer une table client

```
CREATE TABLE T_client  
(  
    id_client      INT PRIMARY KEY NOT NULL AUTO_INCREMENT, 1  
    nom_client     VARCHAR(50),  
    prenom_client  VARCHAR(50),  
    ville_client   VARCHAR(50)  
);
```

Champ id_client

- De type entier
- Qui est une clé primaire
- Non nul
- Qui s'incrmente de 1 à chaque nouvel enregistrement

Les commandes SQL : ALTER TABLE

Objectif

La commande ALTER TABLE permet de modifier une table existante soit en ajoutant, modifiant ou supprimant une colonne.

Syntaxe

```
ALTER TABLE nom_table  
Instructions;
```

Exemples

Ajouter une colonne pays à la table clients

```
ALTER TABLE T_client  
ADD pays_client VARCHAR(50);
```

Supprimer la colonne pays de la table clients

```
ALTER TABLE T_client  OU  ALTER TABLE T_client  
DROP pays_client;        DROP COLUMN pays_client;
```

Modifier la colonne pays de la table clients

```
ALTER TABLE T_client  
Modify pays_client VARCHAR(100);*
```

Renommer la colonne pays de la table clients

```
ALTER TABLE T_client  
CHANGE pays_client pays_du_client;*
```

* Syntaxe propre à MySQL et peut être différente selon les SGBD

Les commandes SQL : DROP TABLE

Objectif

La commande DROP TABLE permet de supprimer une table de la base de données.

Syntaxe

DROP TABLE **nom_table**;

OU

DROP TABLE IF EXISTS **nom_table**;

Exemple

Supprimer la table pays de la base clients

DROP TABLE T_pays;

Supprimer les tables techniciens et vehicules

DROP TABLE IF EXISTS T_tech, T_vehicule;

Les commandes SQL : INSERT INTO

Objectif

La commande INSERT INTO permet d'insérer des données dans une table.

Syntaxe

```
INSERT INTO nom_table ('nom_colonne1', 'nom_colonne2', 'nom_colonne3', 'nom_colonne_4', '....')  
VALUES ('valeur_colonne1', 'valeur_colonne2', 'valeur_colonne3', 'valeur_colonne4', '....');
```

Exemples

1) Insérer une seule ligne dans la table clients

```
INSERT INTO 'T_client' ('id_client', 'nom_client', 'prenom_client', 'adresse_client', 'code_postal_client', 'ville_client')  
VALUES (null, 'Dupont', 'Paul', '15 rue des peupliers', 75010, 'PARIS');
```

2) Insérer plusieurs lignes dans la table clients

```
INSERT INTO T_client ('id_client', 'nom_client', 'prenom_client', 'adresse_client', 'code_postal_client', 'ville_client')  
VALUES (null, 'Durant', 'Nicolas', '10 rue des amandiers', 75010, 'PARIS'),  
(null, 'Leblanc', 'Marc', '50 avenue des chênes', 34000 MONTPELLIER);
```

1 Guillemets simples facultatifs pour le noms de colonnes mais pas pour les valeurs alphanumériques

2 Ne pas oublier le point virgule à la fin de la commande

3 Indiquer une valeur null car le champ a été défini comme étant auto incrémenté

4 Les valeurs de type numérique n'ont pas besoin de guillemets

5 Ne pas oublier la virgule lors de l'insertion de plusieurs lignes

Les commandes SQL : UPDATE

Objectif

La commande UPDATE permet de mettre à jour des données dans les lignes existantes d'une table. Elle est le plus souvent utilisé avec la clause WHERE pour indiquer sur quelles lignes la mise à jour doit être faite

Syntaxe

```
UPDATE nom_table  
SET nom_colonne = valeur  
WHERE condition;
```

Exemples

1) Modifier une seule valeur d'un champ dans la table clients

```
UPDATE T_client  
SET adresse_client = "25 rue des Acacias"  
WHERE id_client=2; ①
```

2) Modifier plusieurs valeurs dans une ligne de la table

```
UPDATE T_client  
SET adresse_client="25 rue des Acacias", ②  
    code_postal_client=25000,  
    ville_client="BESANCON"  
WHERE id_client=2; ③
```

- ① Ne pas oublier le point virgule à la fin de l'instruction UPDATE
- ② Ne pas oublier la virgule lors de la mise à jour de plusieurs champs
- ③ Pas de guillemets pour les champs code_postal_client et id_client puisqu'ils sont de type numérique

Les commandes SQL : DELETE

Objectif

La commande DELETE permet de supprimer des lignes dans une table. Elle est le plus souvent utilisé avec la clause WHERE pour indiquer quelles lignes doivent être supprimées.

Syntaxe

```
DELETE FROM 'nom_table'  
WHERE condition;
```

Exemple

1) Supprimer une ligne de la table clients

```
DELETE FROM 'clients'  
WHERE id_clients=1;
```

2) Supprimer toutes les lignes de la table clients

```
DELETE FROM 'clients';
```

Les commandes SQL : TRUNCATE

Objectif

La commande TRUNCATE permet de supprimer toutes les lignes dans une table. Elle est similaire à la commande DELETE mais avec une fonctionnalité supplémentaire qui est la réinitialisation de l'auto incrémentation s'il y en a une.

Syntaxe

```
TRUNCATE TABLE 'nom_table' ;
```

Exemple

Supprimer les lignes de la table clients

```
TRUNCATE 'clients'
```

Les commandes SQL : SELECT

Objectif

La commande SELECT permet de lire des données provenant d'une ou plusieurs tables d'une base de données.

Syntaxe

```
SELECT nom_de_colonne [AS nom_alias]  
FROM nom_table ;
```

①

②

Exemples

Sélection de tous les noms des clients de la table clients

```
SELECT nom_client AS "nom du client"  
FROM T_client;
```

③

Sélection des noms, des prénoms et des villes de la table clients

```
SELECT nom_client, prenom_client, ville_client  
FROM T_client;
```

Sélection du nom des clients et de leur pays respectifs

```
SELECT nom_client AS Client, pays_client AS Pays  
FROM T_client, T_pays  
WHERE T_clients.id_client = T_pays.id_client;
```

Sélection de toutes les données sur les clients

```
SELECT *  
FROM clients;
```

①

En SQL standard, il n'est pas utile de mettre nom_table entre guillemets.
Sous MySQL, l'éditeur SQL le fait automatiquement (onglet SQL)

②

Il est possible de renommer le nom d'une colonne de table en lui attribuant un alias

③

Le nom de l'alias est à mettre entre guillemets lorsqu'il contient des espaces

Les commandes SQL : La clause WHERE

Objectif

La commande WHERE permet de sélectionner les lignes d'une table qui respectent une condition.

Syntaxe

```
SELECT nom_colonne  
FROM nom_table  
WHERE condition
```

Exemple

Sélectionner les noms et prénoms des clients qui habitent Paris

```
SELECT nom_client, prenom_client  
FROM T_client  
WHERE ville_client = "PARIS";
```

Supprimer le client N°2 de la table clients

```
DELETE  
FROM T_client  
WHERE id_client='2';
```


Les commandes SQL : La clause GROUP BY

Objectif

La commande GROUP BY renvoie un groupe de résultats qui est fonction d'une donnée choisie (une colonne de table). Il est ensuite possible d'effectuer des opérations statistiques sur ce groupe de résultats.

Syntaxe

```
SELECT nom_colonne1, fonction(colonne2)
FROM nom_table
GROUP BY nom_colonne 1;
```

Exemple

Connaitre le montant des ventes d'un magasin

```
SELECT nom_article, SUM(montant_article)
FROM T_article
GROUP BY (nom_article);
```

Connaitre le nombre d'habitants par ville

```
SELECT nom_ville, SUM(nombre_habitants)
FROM T_ville
GROUP BY (nom_ville);
```

Les commandes SQL : La clause ORDER BY

Objectif

La commande ORDER BY permet de trier les lignes d'une table en fonction d'un ou plusieurs champs. Par défaut, le tri se fait de façon ascendante mais il est possible d'inverser l'ordre avec le suffixe DESC. Pour effectuer un tri sur plusieurs champs, il est nécessaire de les séparer par une virgule.

Syntaxe

```
SELECT nom_colonne1, nom_colonne2  
FROM nom_table  
ORDER BY nom_colonne1 [DESC];
```

Si 2 clients portent le nom Dupont, l'affichage se fera ainsi :

-Dupont René
-Dupont Paul

1

Exemple

Sélectionner les clients de la table clients triés par noms.

```
SELECT nom_client, prenom_client  
FROM T_client  
ORDER BY nom_client;
```

Et non pas par ordre alphabétique:

-Dupont Paul
-Dupont René

Sélectionner les clients de la table clients triés par noms et par prénoms en ordre descendant

```
SELECT nom_client, prenom_client  
FROM T_client  
ORDER BY nom_client, prenom_client DESC;
```

1

Les commandes SQL : La clause LIMIT

Objectif

La clause LIMIT permet de spécifier le nombre maximum de résultats que l'on souhaite obtenir.

Syntaxe

1) Syntaxe générale

```
SELECT données  
FROM nom_table  
LIMIT nombre;
```

2) La clause LIMIT peut être utilisée avec la clause OFFSET qui permet d'effectuer un décalage sur le jeu de résultats. Il est à noter que l'OFFSET commence à 0.

```
SELECT données  
FROM nom_table  
LIMIT nombre, nombre;
```

1

OU

```
SELECT données  
FROM nom_table  
LIMIT nombre OFFSET nombre;
```

2

Avec cette 1^{ère} syntaxe :

1^{er} nombre représente l'OFFSET
2^e nombre représente la limite

Avec cette 2nd syntaxe :

1^{er} nombre représente la limite
2^e nombre représente l' OFFSET

Exemples

Sélection des 10 premières lignes de la table clients

```
SELECT *  
FROM T_client  
LIMIT 10;
```

Sélection de 5 lignes de la table clients à partir de la 5^e ligne

```
SELECT *  
FROM T_client  
LIMIT 5, OFFSET 4;
```

3

Cette requête affichera les données de la table clients de la ligne 5 à la ligne 9

FONCTIONS SQL UTILES POUR LES DATES ET LES HEURES		
FONCTIONS	DESCRIPTION	EXEMPLES
CURDATE()	Récupère la date courante	SELECT CURDATE();
DATE()	Stocke une date au format AAAA-MM-JJ	INSERT INTO empl(10, 'Dupont', 'Paul', DATE('2014-01-01'));
DATETIME()	Stocke une date et une heure au format AAAA-MM-JJ HH:MM:SS	INSERT INTO plan (5,'EPS',DATETIME('2014-05-05 09:00:00'));
DATEDIFF()	Spécifie le nombre de jours entre deux dates	SELECT DATEDIFF(NOW(), '2014-01-01');
DATE_FORMAT()	Spécifie l'affichage d'une date selon le format choisi	SELECT DATE_FORMAT('2014-02-25', '%W %M %v'); → Tuesday February 09 (9^e semaine de l'année)
DAYOFMONTH()	Retourne le jour dans le mois (de 1 à 31)	SELECT DAYOFMONTH('2014-02-25');
DAYOFWEEK()	Retourne le jour de la semaine (1= dimanche, 2=lundi,...)	SELECT DAYOFWEEK('2014-02-25'); (DAYNAME() retourne le nom du jour de la semaine)
DAYOFYEAR()	Retourne le jour de l'année (de 1 à 366)	SELECT DAYOFYEAR('2014-02-25');
HOUR()	Extrait le nombre d'heures pour une heure saisie au format HH:MM:SS	SELECT HOUR('15:15:25');
MINUTE()	Extrait le nombre de minutes pour une heure saisie au format HH:MM:SS	SELECT MINUTE('10:05:10');
SECOND()	Extrait le nombre de secondes pour une heure saisie au format HH:MM:SS	SELECT SECOND('09:10:01');
NOW()	Récupère la date et l'heure courante	SELECT NOW();
TIME()	Extrait l'heure, les minutes et les secondes d'une heure au format HH:MM:SS	SELECT TIME('2014-02-25 14:45:45');
TIMEDIFF()	Retourne la durée entre deux heures	SELECT TIMEDIFF(NOW(), '2014-10-05 14:00:00');
TIMESTAMP()	Permet de convertir une DATE au format DATETIME	SELECT TIMESTAMP('2014-10-10'); → '2014-10-10 00:00:00'
WEEK()	Spécifie le numéro de la semaine dans une année à partir d'une date	SELECT WEEK ('2014-02-15')
YEAR()	Extrait l'année d'une date	SELECT YEAR('2010-05-05');

FONCTIONS SQL UTILES POUR LES CHAINES DE CARACTERES

FONCTIONS	DESCRIPTION	EXEMPLES
CHAR_LENGTH(str)	Renvois le nombre de caractères de la chaine str	SELECT CHAR_LENGTH('Bonjour'); → 7
CONCAT(str1,str2,...)	Renvois une chaine représentant la concaténation des arguments	SELECT CONCAT('Bonjour', prenomClt); → 'Bonjour Paul' si prenomClt='Paul'
ELT(N,str1, str2,)	Renvois str1 si N=1, str2 si N=2, etc.... Renvoit NULL si N<1 ou > que le nombre d'arguments	SELECT ELT(4, 'Bonjour', 'tout', 'le', 'monde!'); → 'monde'
FIELD(str, str1, str2,...)	Retourne l'index de la chaine str dans la liste str1, str2, ...	SELECT ('tout', 'Bonjour', 'tout', 'le', 'monde');→ 4
INSTR(str, substr)	Renvois la position de la 1 ^{ère} occurrence de la chaine substr dans la chaine str	SELECT INSTR('Football', 'ball'); → 5
LEFT(str,len)	Renvois les len caractères les plus à gauche de la chaine str	SELECT LEFT('Kite-Surf',4); → Kite
LENGTH(str)	Renvois la taille de la chaine de caractères str	SELECT LENGTH(nomClt); → 7 si nomClt='Nicolas'
LOWER(str)	Renvois la chaine str avec tous les caractères en minuscules	SELECT LOWER('SALUT'); → 'salut'
LTRIM(str)	Renvois la chaine de caractères str sans les espaces initiaux à gauche	SELECT LTRIM(' Bonjour');→ 'Bonjour'
MID(str, pos, len)	Renvois une chaine de len caractères de long de la chaine str, à partir de pos	SELECT MID('Bonjour',4,4);→ 'jour'
REPLACE(str, from_str, to_str)	Renvois la chaine de caractère str dont toutes les occurrences de la chaine from_str sont remplacés par la chaine to_str	SELECT REPLACE('Bateau à voil et planche à voil','l','le');→ 'Bateau à voile et planche à voile'
RIGHT(str, len)	Renvois les len caractères les plus à droite de la chaine de caractères str	SELECT RIGHT ('Volley-ball',4);→ 'ball'
RTRIM(str)	Renvois la chaine de caractères str sans les espaces initiaux à droite	SELECT RTRIM('Bonjour ');→ 'Bonjour'
TRIM(str)	Renvois la chaine de caractères str sans les espaces initiaux à droite ou à gauche	SELECT TRIM(' bonjour ');→ 'bonjour';
UPPER(str)	Renvois la chaine str en majuscules	SELECT UPPER('bonjour');→ 'BONJOUR'

FONCTIONS SQL MATHÉMATIQUES

FONCTIONS	DESCRIPTION	EXEMPLES
ABS(X)	Renvois la valeur absolue de X	SELECT ABS(10); → 10 SELECT ABS(-15); → 15
CEILING(X)	Renvois la valeur entière supérieure de X	SELECT CEILING(1.55); → 2
EXP(X)	Renvois la valeur de e (base des logarithmes naturels) élevé à la puissance X	SELECT EXP(2); → 7.389056
FLOOR(X)	Renvois la valeur entière inférieure de X	SELECT FLOOR(1.55); → 1
MOD(N,M) N%M N MOD M	Modulo. Renvois le reste de la division de N par M	SELECT MOD(21,4); → 5
PI()	Renvois la valeur de PI avec 5 décimales par défaut. MySQL utilise la double précision	SELECT PI(); → 3.141593 SELECT PI() + 0.0000000000 → 3.1415926535 (10 chiffres après le séparateur)
POW(X,Y)	Renvois la valeur de X élevé à la puissance Y	SELECT POW(5,5); → 25
RAND() RAND(N))	Renvois un nombre aléatoire à virgule flottante compris entre 0 et -1.0 N est utilisé comme initialiseur du générateur.	SELECT RAND(); SELECT RAND(10)
ROUND() ROUND(X,D)	Renvois l'argument X arrondi à D décimales. Si D vaut 0, le résultat n'aura pas de partie décimale	SELECT (1.55); → 1 SELECT(1.499,1); → 1.5
SQRT()	Renvois la racine carrée de X	SELECT SQRT(25); → 5.000000
TRUNCATE(X,D)	Renvois l'argument X, tronqué à D décimales . Si D vaut 0, le résultat n'aura pas de partie décimal	SELECT TRUNCATE(5,55574); → 5,55

Les fonctions d'agrégation

Les fonctions d'agrégation permettent d'effectuer des opérations statistiques sur un ensemble d'enregistrements. Nous traiterons des fonctions suivantes :

- ➡ **AVG()** : Permet de déterminer la moyenne des valeurs d'un ensemble de données.
- ➡ **COUNT()** : Permet de compter le nombre d'enregistrement selon un certain nombre de critères choisis.
- ➡ **MAX()** : Permet de déterminer la plus grande valeur d'un ensemble de données.
- ➡ **MIN()** : Permet de déterminer la plus petite valeur d'un ensemble de données.
- ➡ **SUM()** : Permet de calculer la somme d'un ensemble de données.

Les fonctions d'agrégation : La fonction AVG()

Objectif

La fonction AVG() permet calculer une valeur moyenne sur un ensemble d'enregistrement de type numérique et non nul. Cette fonction s'utilise le plus souvent avec la clause GROUP BY.

Syntaxe

```
SELECT AVG (nom_colonne)  
FROM nom_table ;
```

Exemples

Connaitre la moyenne des notes de Paul qui a eut le notes suivantes : 15/20, 14/20, 13/20, 17/20 ,15/20

```
SELECT nom_eleve, AVG(note_eleve)  
FROM T_eleve  
GROUP BY nom_eleve;  
→ 14,8
```

Connaitre la moyenne des achats par clients

```
SELECT nom_client, AVG(montant_achat)  
FROM T_client  
GROUP BY nom_client;
```


Les fonctions d'agrégation : La fonction COUNT()

Objectif

La fonction COUNT() permet de connaître le nombre d'enregistrements dans une table.

Syntaxe

```
SELECT COUNT(*) FROM nom_table ;
```

```
SELECT COUNT(nom_colonne) FROM nom_table;
```

```
SELECT COUNT(DISTINCT nom_colonne) FROM nom_table (Valeurs Null non comptabilisées dans ce type de requête)
```

Exemples

Connaitre le nombre d'enregistrement de la table Vehicule

```
SELECT COUNT (*) FROM T_vehicule;
```

Connaitre les marques de véhicules vendus chez un concessionnaire automobile d'occasion, quelque soit le modèle
(Si un enregistrement contient la valeur 'Null' dans la colonne marque_vehicule, il n'est pas comptabilisé).

```
SELECT COUNT (marque_vehicule) FROM T_vehicule;
```

Connaitre, de façon distincte, les marques des véhicules vendus chez ce concessionnaire
(Les doublons ne sont pas comptabilisés).

```
SELECT COUNT(DISTINCT marque_vehicule) FROM T_vehicule;
```

Les fonctions d'agrégation : La fonction MIN()

Objectif

La fonction MIN() permet de retourner la plus petite valeur d'un groupe de données sélectionné dans une colonne. Cette fonction peut être utilisé avec la clause GROUP BY.

Syntaxe

```
SELECT MIN(nom_colonne) from nom_table;
```

Exemples

Connaitre l'article le moins cher d'un magasin

```
SELECT nom_article, MIN(prix_article)
FROM T_article;
```

Connaitre la note la plus basse de chaque élève

```
SELECT MIN(note_eleve)
FROM T_eleve
GROUP BY nom_eleve;
```

Les fonctions d'agrégation : La fonction MAX()

Objectif

La fonction MAX() permet de retourner la plus grande valeur d'un groupe de données sélectionné dans une colonne. Cette fonction peut être utilisé avec la clause GROUP BY.

Syntaxe

```
SELECT MAX(nom_colonne) from nom_table;
```

Exemples

Connaitre l'article le plus cher d'un magasin

```
SELECT nom_article, MAX(prix_article)  
FROM T_article;
```

Connaitre la note la plus élevée de chaque élève

```
SELECT MAX(note_eleve)  
FROM T_eleve  
GROUP BY nom_eleve;
```

Les fonctions d'agrégation : La fonction SUM()

Objectif

La fonction SUM() permet de calculer la somme totale d'une colonne contenant des valeurs numériques. Cette fonction peut être utilisé avec la clause WHERE.

Syntaxe

```
SELECT SUM (nom_colonne) from nom_table;
```

Exemples

Connaitre l'article le plus cher d'un magasin

```
SELECT nom_article, MAX(prix_article)  
FROM T_articles;
```

Connaitre le montant des achats du client DUPONT

```
SELECT nom_client, SUM(montant_article)  
FROM T_client  
WHERE nom_client='DUPONT';
```

La gestion des utilisateurs dans MySQL : Création des utilisateurs

Objectif

La création d'un utilisateur sous MySQL s'effectue en deux étapes :

- 1) La création de l'utilisateur avec les commandes SQL **CREATE USER** ou **INSERT INTO**
- 2) La création des privilèges de l'utilisateur sur les bases de données avec la commande **GRANT**

Syntaxe

Création de l'utilisateur

1

CREATE USER nom_utilisateur@typehôte [IDENTIFIED BY 'motdepasse'];

OU

2

INSERT INTO user(hôte, nom_utilisateur, motdepasse)
VALUES('valeur_hôte', 'valeur_nom_utilisateur', 'valeur_motdepasse');

Exemples

3

CREATE USER Nicolas@localhost IDENTIFIED BY 'Nico1';

INSERT INTO user('localhost', 'Nicolas', 'Nico1');

1

IDENTIFIED BY entre crochets signifie que le mot de passe est facultatif

2

La table user est une table système MySql

3

'localhost' représente l'ordinateur client sur lequel l'utilisateur va se connecter pour accéder à la base de données

La gestion des utilisateurs dans MySQL : Les privilèges des utilisateurs

Objectif

Lorsque l'utilisateur est créé, Il faut lui attribuer des privilèges pour pouvoir utiliser une base de donnée.

La commande GRANT donne les permissions et la commande REVOKE les supprime.

Voici les types de permissions :

- ALL PRIVILEGES : Accès complet à la base de donnée spécifiée ou au SGBD si aucune base n'est spécifiée.
- CREATE : Permission de créer de nouvelles tables ou de nouvelles bases de données.
- DROP : Permission de supprimer des tables ou des bases de données.
- DELETE : Permission de supprimer des enregistrements dans les tables d'une base de données.
- SELECT : Permission d'utiliser la commande SELECT pour lire des enregistrements dans une base de données.
- UPDATE : Permission de mettre à jour des enregistrements dans les tables d'une base de données.

Syntaxe

GRANT type_permissions ON liste_objets TO liste_utilisateurs [WITH GRANT OPTION];

REVOKE type_permissions ON liste_objets FROM liste_utilisateurs

1

1 L'option With Grant Option permet de définir si un utilisateur peut accorder les droits qu'on lui donne à un autre utilisateur.

Exemples

GRANT ALL PRIVILEGES ON Clients TO Nicolas WITH GRANT OPTION;	GRANT UPDATE ON Clients TO Paul, Fabrice;	REVOKE ALL PRIVILEGES, GRANT OPTION ON Clients FROM Nicolas	REVOKE UPDATE ON Clients TO Paul, Fabrice
--	---	---	---

Les jointures SQL

Les jointures SQL permettent d'associer plusieurs tables dans une requête grâce à un lien logique.

Les informations demandées dans la requête et réparties dans ces tables peuvent ainsi être combinées et affichées dans une seule et même table temporaire ou dans une vue.

Il existe différents types de jointure regroupées en deux catégories :

➡ Les jointures internes

Les résultats de ces jointures n'incluent que les enregistrements pour lesquels il existe une correspondance entre les occurrences des tables jointes.

Cette catégorie comprend les jointures de type

➡ INNER JOIN

➡ NATURAL JOIN

➡ Les jointures externes

Les résultats de ces jointures incluent chaque enregistrement d'une table même s'il n'y a pas de correspondance avec les occurrences de la table jointe.

Cette catégorie comprend les jointures

➡ LEFT JOIN,

➡ RIGHT JOIN,

➡ FULL JOIN

Les jointures internes : INNER JOIN

Objectif

La jointure interne INNER JOIN renvoie les enregistrements lorsqu'il y a au moins une correspondance entre les tables qui répond à la condition.

Syntaxe

```
SELECT nom_colonne_table FROM nom_table1  
INNER JOIN nom_table2 ON nom_table1.pk_id = nom_table2.fk_id;
```

① pk : primary key

② fk : foreign key

Exemple

Connaitre le véhicule de fonction de chaque employé

```
SELECT nom_empl, prenom_empl, marque_vehic, modele_vehic  
FROM employes  
INNER JOIN vehicules on employes.id_empl=vehicules.id_empl;
```

Table employes		
id_empl	nom_empl	prenom_empl
1	DUPONT	Paul
2	DURANT	Nicolas
3	LAGARDE	Sébastien
4	LEBLANC	Jean

Table vehicule			
id_vehic	Id_empl	Marque_vehic	Modele_vehic
1	4	MERCEDES	Classe A
2	1	MERCEDES	Classe C
3	3	BMW	525
4	2	BMW	X5

RESULTAT DE LA REQUETE			
nom_empl	prenom_empl	marque_vehic	modele_vehic
DUPONT	Paul	MERCEDES	Classe C
DURANT	Nicolas	BMW	X5
LAGARDE	Sébastien	BMW	525
LEBLANC	Jean	MERCEDES	Classe A

Les jointures internes : NATURAL JOIN

Objectif

La jointure interne NATURAL JOIN permet de faire une jointure sur deux tables à condition qu'il y ai des colonnes de même nom et de même type dans les deux tables.

Syntaxe

```
SELECT nom_colonne_table FROM nom_table1  
NATURAL JOIN nom_table2;
```

Exemple

Connaitre le véhicule de fonction de chaque employé

```
SELECT nom_empl, prenom_empl, marque_vehic, modele_vehic  
FROM employes  
NATURAL JOIN vehicules;
```

Table employes		
id_empl	nom_empl	prenom_empl
1	DUPONT	Paul
2	DURANT	Nicolas
3	LAGARDE	Sébastien
4	LEBLANC	Jean

Table vehicule			
id_vehic	Id_empl	Marque_vehic	Modele_vehi
1	4	MERCEDES	Classe A
2	1	MERCEDES	Classe C
3	3	BMW	525
4	2	BMW	X5

RESULTAT DE LA REQUETE			
nom_empl	prenom_empl	marque_vehic	modele_vehic
DUPONT	Paul	MERCEDES	Classe C
DURANT	Nicolas	BMW	X5
LAGARDE	Sébastien	BMW	525
LEBLANC	Jean	MERCEDES	Classe A

Les jointures externes : LEFT JOIN

Objectif

La jointure LEFT JOIN (aussi appelée LEFT OUTER JOIN) permet de lister tous les enregistrements de la table de gauche même s'il n'y a pas de correspondance dans la table de droite.

Syntaxe

```
SELECT nom_colonne_table FROM nom_table1  
LEFT JOIN nom_table2  
ON nom_table1.pk_id = nom_table2.fk_id;
```

OU

```
SELECT nom_colonne_table FROM nom_table1  
LEFT OUTER JOIN nom_table2  
ON nom_table1.pk_id = nom_table2.fk_id;
```

Exemple

```
SELECT nom_empl, prenom_empl, marque_vehic, modele_vehic FROM employes  
LEFT JOIN vehicules ON employes.id_empl=vehicules.id_empl;
```

Table employes		
id_empl	nom_empl	prenom_empl
1	DUPONT	Paul
2	DURANT	Nicolas
3	LAGARDE	Sébastien
4	LEBLANC	Jean
5	MARTY	Franck
6	MARECHAL	Thomas
7	MULTIER	Edouard
8	MULLIEZ	Henri

Table vehicule			
id_vehic	Id_empl	Marque_vehic	Modele_vehic
1	4	MERCEDES	Classe A
2	1	MERCEDES	Classe C
3	3	BMW	525
4	2	BMW	X5

Tous les employés de la table employés sont affichés même s'il n'ont pas de véhicule de fonction. Lorsque c'est le cas, la requête affiche NULL. La fonction **isnull(nom_colonne)** vérifie si un champ a une valeur a null.

RESULTAT DE LA REQUETE			
nom_empl	prenom_empl	marque_vehic	modele_vehic
DUPONT	Paul	MERCEDES	Classe C
DURANT	Nicolas	BMW	X5
LAGARDE	Sébastien	BMW	525
LEBLANC	Jean	MERCEDES	Classe A
MARTY	Franck	NULL	NULL
MARECHAL	Thomas	NULL	NULL
MULTIER	Edouard	NULL	NULL
MULLIEZ	Hendri	NULL	NULL

Les jointures externes : RIGHT JOIN

Objectif

La jointure RIGHT JOIN (aussi appelée RIGHT OUTER JOIN) permet de lister tous les enregistrements de la table de droite même s'il n'y a pas de correspondance dans la table de gauche.

Syntaxe

```
SELECT nom_colonne_table FROM nom_table1  
LEFT JOIN nom_table2  
ON nom_table1.pk_id = nom_table2.fk_id;
```

OU

```
SELECT nom_colonne_table FROM nom_table1  
LEFT OUTER JOIN nom_table2  
ON nom_table1.pk_id = nom_table2.fk_id;
```

Exemple

```
SELECT nom_empl, prenom_empl, marque_vehic_modele_vehic from employes  
RIGHT JOIN vehicules ON employes.id_empl=vehicules.id_empl;
```

La requête affiche l'ensemble des véhicules même ceux qui ne sont pas affiliés à un employé. Lorsque c'est le cas, la requête affiche NULL.

Table employes		
id_empl	nom_empl	prenom_empl
1	DUPONT	Paul
2	DURANT	Nicolas
3	LAGARDE	Sébastien
4	LEBLANC	Jean

Table vehicule			
id_vehic	Id_empl	Marque_vehic	Modele_vehic
1	4	MERCEDES	Classe A
2	1	MERCEDES	Classe C
3	3	BMW	525
4	2	BMW	X5
5	NULL	Bentley	Continental GT

RESULTAT DE LA REQUETE			
nom_empl	prenom_empl	marque_vehic	modele_vehic
DUPONT	Paul	MERCEDES	Classe C
DURANT	Nicolas	BMW	X5
LAGARDE	Sébastien	BMW	525
LEBLANC	Jean	MERCEDES	Classe A
NULL	NULL	BENTLEY	Continental GT

La jointure CROSS JOIN : Le produit cartésien

Objectif

Le produit cartésien de deux ensemble X et Y est l'ensemble de tous les couples (a, b) où a appartient à X et b appartient à Y. En SQL, il est possible d'effectuer un produit cartésien avec la jointure CROSS JOIN. Elle agit sur deux tables et permet de renvoyer chaque ligne d'une table avec chaque ligne de l'autre table.

Syntaxe

```
SELECT nom_colonne(s)_table
FROM nom_table1
CROSS JOIN nom_table2;

OU

SELECT nom_colonne(s)_table
FROM nom_table1, nom_table2;
```

Exemple

```
SELECT nom_ingrédient, nom_dessert FROM ingredients
CROSS JOIN desserts;
```

Table Ingrédient	
id_ingredient	nom_ingredient
1	Farine
2	Oeufs

Table Dessert	
id_dessert	nom_dessert
1	Tarte aux pommes
2	Mousse au chocolat
3	Crêpes
4	Choux à la crème

PRODUIT CARTESIEN	
nom_ingredient	nom_dessert
Farine	Tarte aux pommes
Farine	Mousse aux chocolat
Farine	Crêpes
Farine	Choux à la crème
Oeufs	Tarte aux pommes
Oeufs	Mousse aux chocolat
Oeufs	Crêpes
Oeufs	Choix à la crème

Les vues en SQL : Créer une vue

Objectif

Une vue peut être considérée comme une table virtuelle qui ne stocke pas les données et permet de réunir des informations provenant de plusieurs tables.

L'intérêt principal d'une vue est de sécuriser les tables des bases de données puisque l'on travaille sur des données non enregistrées.

Syntaxe

Création de la vue

```
CREATE VIEW nom_vue[(Alias_nom(s)_colonne(s)) AS  
SELECT nom(s)_colonne(s)  
FROM nom(s)_table(s);
```

1



Appel de la vue

```
SELECT nom(s)_colonne(s)_vue FROM nom_vue
```

2

Exemples

Créer une vue qui liste les employés avec leur véhicule de fonction attribué

```
CREATE VIEW V_emplcar (Nom, Prénom, Marque, Modèle) AS  
SELECT nom_empl, prenom_empl, marque_vehicule, modele_vehicule  
FROM T_employes, T_vehicules  
WHERE T_employes.id_empl = T_vehicule.id_empl;
```

Appel

```
SELECT * FROM V_emplcar;
```

1

Les alias sont facultatifs et représentent les noms affichés dans les colonnes de la vue

2

Les noms de colonnes sont soit les noms des colonnes des tables soit les alias que vous avez spécifiés lors de la création de la vue

Les vues SQL : La commande ALTER VIEW

Objectif

La commande ALTER VIEW permet de modifier une vue dans une base de données.

Syntaxe

```
ALTER VIEW nom_vue [(Alias_nom(s)_colonne(s))] AS  
SELECT nom(s)_colonne(s)  
FROM nom(s)_table(s);
```

Exemple

Ne plus afficher le prénom des employés dans la vue qui liste les employés et leur véhicule de fonction attribué.

```
ALTER VIEW v_emplcar (nom, marque, modele) AS  
SELECT nom_empl, marque_vehicule, modele_vehicule  
FROM T_employe, T_vehicule  
WHERE T_employe.id_empl = T_vehicule.id_empl;
```

Les vues en SQL : La commande DROP VIEW

Objectif

La commande **DROP VIEW** permet de supprimer une vue de la base de donnée.

Syntaxe

DROP VIEW nom_vue;

Exemple

Supprimer la vue employes

DROP VIEW V_employes;

Import/export de données avec MySQL

Objectif

Dans un système de gestion de bases de données, il peut être intéressant de pouvoir :

- ➡ Exporter l'ensemble des informations contenu dans une base de données pour
 - ➡ Effectuer une sauvegarde de sécurité (backup)
 - ➡ Stocker une base sur un support amovible pour travailler chez soi
 - ➡ Comparer les données de la base de données à un instant T
 - ➡ Tester la portabilité d'une base de données dans différents SGBD
 - ➡ Importer des informations dans une base de données pour
 - ➡ Un gain de temps lors de la création de la base
 - ➡ Fusionner deux bases de données
 - ➡ Tester la portabilité d'une base de données dans différents SGBD
 - ➡ Migrer des données de différents fichiers Excel vers une bases de données centralisée
- Pour effectuer ces opérations, nous emploierons deux méthodes :
- ➡ Avec l'interface graphique de phpMyAdmin
 - ➡ Avec MySQL en ligne de commande

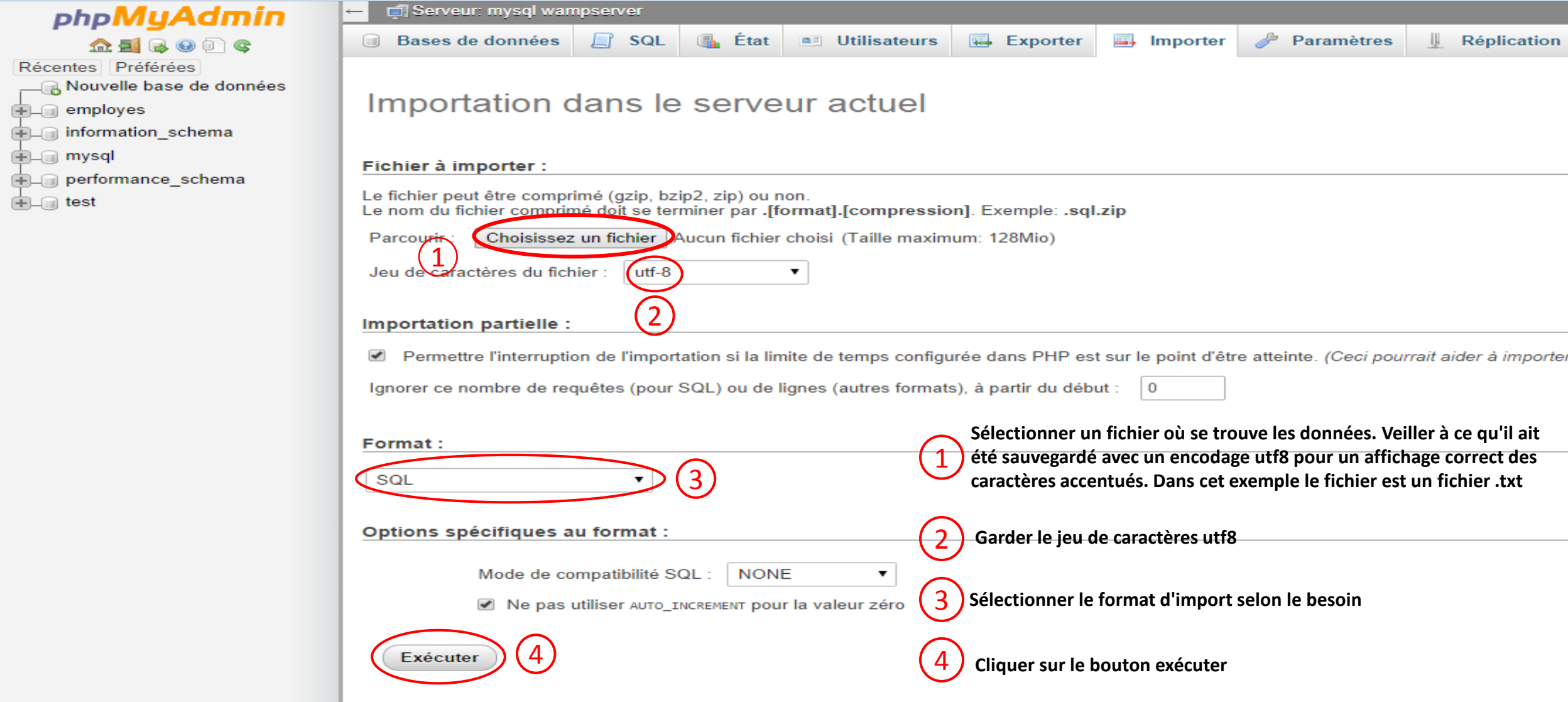
Import de données dans une base de données avec phpMyAdmin(1)

The screenshot displays the phpMyAdmin web interface. On the left sidebar, the 'employees' database is highlighted with a red box and a red circle with the number '1'. The main menu at the top includes 'Bases de données', 'SQL', 'État', 'Utilisateurs', 'Exporter', 'Importer' (circled in red with a red circle and the number '2'), 'Paramètres', and 'Répl'. The 'Paramètres généraux' section shows the 'Interclassement pour la connexion au serveur' set to 'utf8mb4_general_ci'. The 'Paramètres d'affichage' section shows the 'Langue - Language' set to 'Français - French', the 'Thème' set to 'pmahomme', and the 'Taille du texte' set to '82%'. A red circle with the number '1' is placed next to the 'Langue - Language' dropdown.

1 Se positionner sur la page d'accueil ou sur une base existante selon que l'on souhaite créer ou mettre à jour une base

2 Cliquer sur le l'onglet Importer

Import de données dans une base de données avec phpMyAdmin(2)



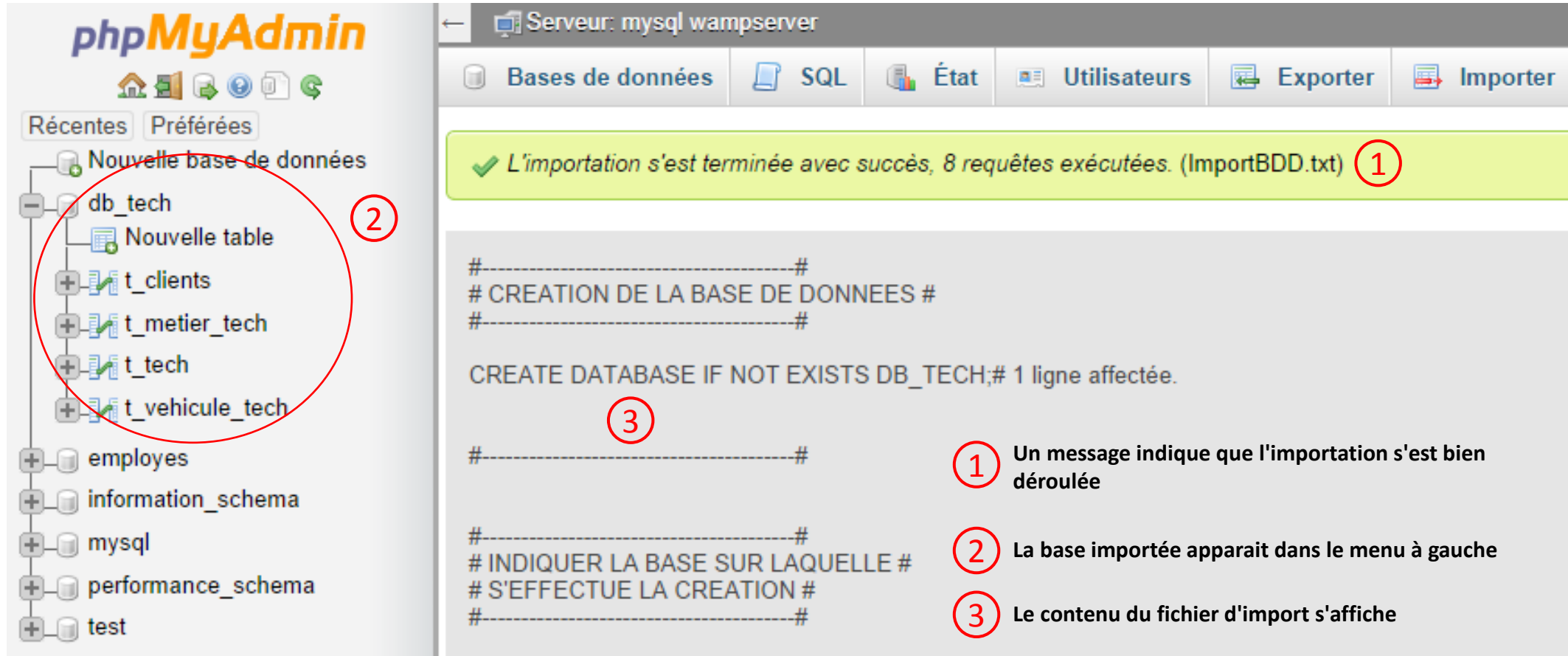
The screenshot shows the phpMyAdmin interface for importing data into the current server. The left sidebar lists recent databases: 'employes', 'information_schema', 'mysql', 'performance_schema', and 'test'. The main panel is titled 'Importation dans le serveur actuel' and contains the following sections:

- Fichier à importer :** Includes a text area for the file path, a 'Choisissez un fichier' button (circled with a red circle and labeled 1), and a 'Jeu de caractères du fichier' dropdown menu set to 'utf-8' (circled with a red circle and labeled 2).
- Importation partielle :** Includes a checkbox for 'Permettre l'interruption de l'importation si la limite de temps configurée dans PHP est sur le point d'être atteinte.' and a text input for 'Ignorer ce nombre de requêtes (pour SQL) ou de lignes (autres formats), à partir du début' set to '0'.
- Format :** A dropdown menu set to 'SQL' (circled with a red circle and labeled 3).
- Options spécifiques au format :** Includes a 'Mode de compatibilité SQL' dropdown set to 'NONE' and a checkbox for 'Ne pas utiliser AUTO_INCREMENT pour la valeur zéro'.
- Exécuter :** A button labeled 'Exécuter' (circled with a red circle and labeled 4).

Annotations on the right side of the image provide instructions for each numbered circle:

- 1 Sélectionner un fichier où se trouve les données. Veiller à ce qu'il ait été sauvegardé avec un encodage utf8 pour un affichage correct des caractères accentués. Dans cet exemple le fichier est un fichier .txt
- 2 Garder le jeu de caractères utf8
- 3 Sélectionner le format d'import selon le besoin
- 4 Cliquer sur le bouton exécuter

Import de données dans une base de données avec phpMyAdmin(3)



The screenshot displays the phpMyAdmin interface for a MySQL server. On the left, the database 'db_tech' is selected, and its tables are listed. A red circle highlights the tables 't_clients', 't_metier_tech', 't_tech', and 't_vehicule_tech', with a red circle and the number '2' next to it. The main panel shows a success message: 'L'importation s'est terminée avec succès, 8 requêtes exécutées. (ImportBDD.txt)' with a red circle and the number '1' next to it. Below the message, the SQL content of the import file is displayed, with a red circle and the number '3' next to the line 'CREATE DATABASE IF NOT EXISTS DB_TECH;# 1 ligne affectée.'

1 Un message indique que l'importation s'est bien déroulée

2 La base importée apparaît dans le menu à gauche

3 Le contenu du fichier d'import s'affiche

Exporter des données avec phpMyAdmin (1)

phpMyAdmin

Récentes Préférées

Nouvelle base de données

db_tech

Nouvelle table

t_clients

t_metier_tech

t_tech

t_vehicule_tech

employees

information_schema

mysql

performance_schema

test

Serveur: mysql wampserver » Base de données: db_tech

Structure SQL Rechercher Requête Exporter Importer Opérations

Exportation des tables depuis la base de données «db_tech»

Méthode d'exportation :

☒ Rapide - n'afficher qu'un minimum d'options

☐ Personnalisée - afficher toutes les options possibles

Format :

SQL

CodeGen

CSV

CSV for MS Excel

Microsoft Word 2000

JSON

LaTeX

MediaWiki Table

OpenDocument Spreadsheet

OpenDocument Text

PDF

PHP array

SQL

Texy! text

XML

YAML

Exécuter

- 1 Sélectionner la base à exporter
- 2 Cliquer sur l'onglet Exporter
- 3 Choisir la méthode d'exportation
- 4 Choisir le format du fichier dans lequel les données seront exportées
- 5 Cliquer sur le bouton Exécuter pour lancer l'exportation. Le fichier est sauvegardé dans le répertoire Téléchargements de l'utilisateur courant

MySQL en ligne de commande

Se connecter à MySQL en ligne de commande

Syntaxe

Shell> `mysql -h hostname -u username -p`

Enter password : password

OU

Shell> `mysql -h hostname -u username -ppassword`

hostname : représente le nom ou l'adresse IP de la machine sur laquelle se trouve la base de donnée. En général, il faut saisir localhost.

username : représente le nom de l'utilisateur avec lequel nous souhaitons nous connecter.

password : représente le mot de passe de l'utilisateur avec lequel nous souhaitons nous connecter.

Il est à noter que dans la seconde syntaxe, il n'y a pas d'espace entre l'option `-p` et password.

Exemples

Shell> `mysql -h localhost -u userclient -p`

Enter password : titi

Shell> `mysql -h localhost -u userclient -ptiti`

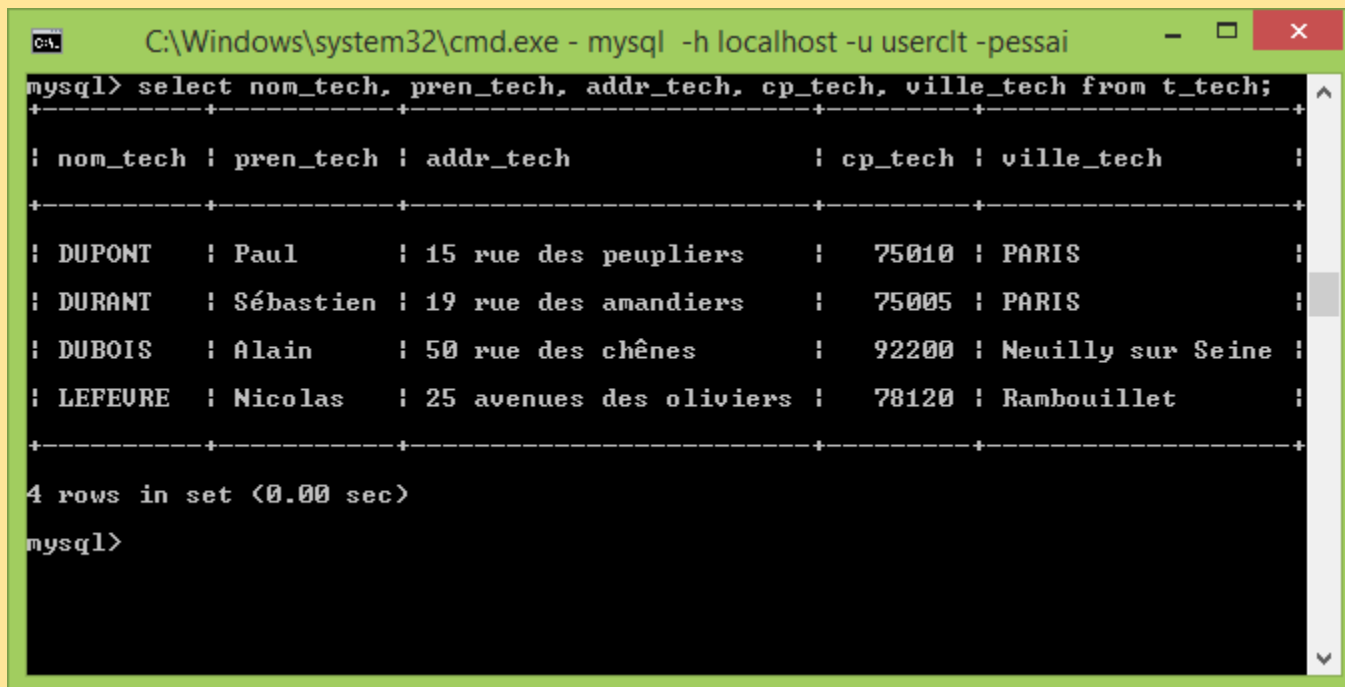
MySQL en ligne de commande

Sélectionner les identités et coordonnées des techniciens dans la base db_tech

1) Indiquer la base sur laquelle nous souhaitons travailler

```
mysql> use db_tech;  
Database changed  
mysql>
```

2) Saisir la requête souhaitée et afficher le résultat



The screenshot shows a Windows command prompt window with the title bar "C:\Windows\system32\cmd.exe - mysql -h localhost -u userclt -pessai". The command prompt displays the following text:

```
mysql> select nom_tech, pren_tech, addr_tech, cp_tech, ville_tech from t_tech;  
+-----+-----+-----+-----+-----+  
| nom_tech | pren_tech | addr_tech | cp_tech | ville_tech |  
+-----+-----+-----+-----+-----+  
| DUPONT   | Paul      | 15 rue des peupliers | 75010 | PARIS      |  
| DURANT   | Sébastien | 19 rue des amandiers | 75005 | PARIS      |  
| DUBOIS    | Alain     | 50 rue des chênes    | 92200 | Neuilly sur Seine |  
| LEFEURE   | Nicolas   | 25 avenues des oliviers | 78120 | Rambouillet |  
+-----+-----+-----+-----+-----+  
4 rows in set (0.00 sec)  
mysql>
```

Importer des donnée avec MySQL en ligne de commande(1)

Objectif

MySQL offre la possibilité d'importer des données dans une base de données à l'aide de l'utilitaire `mysqlimport`. Cet utilitaire agit sur les données d'une base mais ne permet pas de créer ou d'en modifier la structure. `mysqlimport` est similaire à `LOAD DATA INFILE`.

Syntaxe

Shell> `mysqlimport` [options] nom_de_la_base nom_du_fichier1.format

①

②

③

④

- ① Pour exécuter `mysqlimport.exe`, il est nécessaire de se placer dans son répertoire d'installation (en général le répertoire `mysqlXXXX\bin` où `XXXX` représente la version du serveur MySQL installé. (ex: `C:\wamp\bin\mysql\mysql5.6.17\bin\`)
- ② `mysqlimport` supporte de nombreuses options nécessaires lors du processus d'importation. Il faut, en général, indiqué l'hôte client (`-h localhost`) sur lequel se trouve la base de données ainsi que les paramètres de connexion de l'utilisateur qui va importer les données dans la base de données (cf exemple ci-après)
- ③ Il est nécessaire d'indiquer le nom de la base de données dans laquelle les données vont être importées.
- ④ Il est nécessaire d'indiquer le nom du fichier dans lequel se trouve les données à importer.
Le fichier peut être de différents formats (.csv, .txt, .sql,)
ATTENTION: 1) Le fichier doit être enregistré dans le répertoire où se trouve la base de données (`mysqlXXXX\data\nom_de_la_base`)
2) Le nom du fichier doit porter le nom de la table dans laquelle les données vont être importées.
Ex: Si les données doivent être importées dans une table `clients`, le fichier devra se nommer : `clients.txt`

Importer des données avec MySQL en ligne de commande(2)

Exemple

Ajouter un technicien dans la table technicien de la base DB_tech

```
Shell>mysqlimport - -fields-terminated-by=";" -h localhost -u userclient -pessai DB_tech T_tech.txt
```

--fields-terminated-by ";" : Avec cette option, j'indique que chaque valeur séparée par un point-virgule doit être enregistrée dans chaque champs de la table.
(Cf fichier T_tech.txt)

-h localhost : le client sur lequel se trouve la base de donnée

-u userclient : l'utilisateur qui va importer les données dans la base de données

-pessai : le mot de passe de l'utilisateur

DB_tech : le nom de la base de données qui va recevoir les données du fichier

T_tech : le nom du fichier texte qui porte le nom de la table dans laquelle les données vont être insérées

Exporter des données avec MySQL en ligne de commande

Objectif

Outre la possibilité d'importer des données en ligne de commande, MySQL permet d'exporter une ou des bases vers un fichier texte grâce à l'utilitaire mysqldump.

Syntaxe

Shell>mysqldump[options] nom_de_la_base>nom_du_fichier.txt

①

Exemple

Sauvegarde de la base DB_tech dans un fichier texte

Shell>mysqldump -h localhost -u userclt -pessai DB_tech>Backup.txt

②

① Ne pas oublier le signe > qui indique que la base va être exporter

② Le fichier Backup.txt sera sauvegardé dans le répertoire d'installation de l'utilitaire mysqldump
(C:\....\mysql\mysqlXXXX\bin\)

Les conventions de nommage en SQL

Quelques règles :

- ➡ Ne pas utiliser de mots réservés. (Ex: SELECT, ADD, ALTER, SQRT,...)
- ➡ Ne pas utiliser de caractères spéciaux. (Ex: @, \$\$)
- ➡ Nommer de façon pertinente les éléments d'une base de données (tables, noms de colonnes, relations,...).
- ➡ Ne pas utiliser d'espaces ou d'accents lors du nommage des éléments d'une base de données
- ➡ Préfixer les éléments d'une base de données. (Ex: T_client, pour une table client, DB_commande pour une base de données Commandes)
- ➡ Les majuscules sont déconseillées dans le nommage des éléments d'une base de donnée.
- ➡ Privilégier les majuscules pour l'écriture du code SQL (Ex: SELECT NOM_CLI FROM CLIENT WHERE ID_CLI = 2)
- ➡ Privilégier le singulier dans le nommage des éléments d'une base de données. (Ex: T_voiture)
- ➡ Utiliser l'underscore (_) pour les noms composés (Ex: T_vehicule_a_moteur)
- ➡ Les champs clé primaire doivent commencer par les lettres ID suivi du nom du champ (Ex: ID_commande)
- ➡ Les noms de champs doivent commencer par le trigramme de la table dans laquelle ils se trouvent (Ex: nom_cli pour le nom d'un client)
- ➡ Il est utile d'indiquer en commentaires le nom de l'auteur d'une requête ainsi que la date de sa création et une explication sur sa fonction.

Webographie

OUTILS DE DEVELOPPEMENTS SQL

WampServer : <http://www.wampserver.com>

MySQL : <http://www.mysql.fr>

DOCUMENTATION SQL

SQL

<http://sql.developpez.com>

<http://sql.sh>