

# SQL avec MySQL

Par Richard BONNAMY

## Sommaire

- Introduction
- Gestion des bases de données
- Gestion des tables - compléments
- Requêtes d'insertion
- Requêtes de lecture
- Requêtes de lecture avec jointures
- Requêtes de Mise à jour
- Requêtes de Suppression
- Gestion des clés étrangères
- Indexation
- Annexes

# Introduction

## □ SQL

- Structured Query Language
- Créé dans les années 70s par IBM



## □ Périmètre

- Gestion des bases de données relationnelles
  - Gestion d'utilisateurs
  - Gestion de tables
    - Structures
    - Contraintes
  - Gestion d'index
- Gestion des données
  - Insertion
  - Modification
  - Suppression
  - Lecture

## □ MySQL

- Le serveur de base de données MySQL permet de gérer plusieurs bases de données.

# Bases de données Gestion

## ❑ Création d'une base de données

- MySQL permet de gérer plusieurs bases de données.

```
CREATE DATABASE ma_base;
```

```
CREATE DATABASE ma_base COLLATE 'utf8_general_ci';
```

⚠ Erreur si la base de données existe

```
CREATE DATABASE IF NOT EXISTS ma_base;
```

## ❑ Suppression d'une base de données

```
DROP DATABASE ma_base;
```

⚠ Erreur si la base de données ma\_base n'existe pas

```
DROP DATABASE IF EXISTS ma_base;
```

## ❑ Utilisation d'une base de données

USE ma\_base;

- Commande inutile en Java sauf si vous travaillez avec plusieurs bases de données
- Commande utile dans PhpMyAdmin pour être sûr d'être positionné sur la bonne base de données

## ❑ Information sur la base de données courante

SELECT DATABASE();



# Gestion des tables

## ❑ Création d'une table:

```
CREATE TABLE table_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype,  
    ....  
);
```

## ❑ Exemple:

```
CREATE TABLE Persons (  
    PersonID int,  
    LastName varchar(255),  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255)  
);
```

## ❑ Création d'une table avec AUTO\_INCREMENT :

```
CREATE TABLE table_name (  
    column1 INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    column2 datatype,  
    column3 datatype, ....  
);
```

## ❑ Exemple:

```
CREATE TABLE Persons (  
    PersonID int NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    LastName varchar(255),  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255)  
);
```

- ❑ **Primary key:**

- ❑ Définit une contrainte d'unicité sur une colonne

- ❑ **Exemple:**

```
CREATE TABLE PERSONNES (  
    EMAIL VARCHAR(50) PRIMARY KEY,  
    NOM varchar(255),  
    PRENOM varchar(255),  
    ADRESSE varchar(255),  
    VILLE varchar(255)  
);
```

## ❑ Primary key:

- ❑ Définit une contrainte d'unicité sur une ou plusieurs colonnes d'une table
- ❑ Si colonne est en AUTO\_INCREMENT elle doit forcément être la clé primaire

## ❑ Exemple:

```
CREATE TABLE Persons (  
    PersonID int NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    LastName varchar(255),  
    FirstName varchar(255),  
    Address varchar(255),  
    City varchar(255)  
);
```

## ❑ Exemple :

- AUTO\_INCREMENT dans une primary key avec multiple colonnes

```
CREATE TABLE animals (  
  grp ENUM('fish','mammal','bird') NOT NULL,  
  id MEDIUMINT NOT NULL AUTO_INCREMENT,  
  name CHAR(30) NOT NULL,  
  PRIMARY KEY (grp,id)  
);
```

## ❑ Création d'une table avec la date/heure courante

- Pas besoin de renseigner la date dans la colonne curdate, c'est automatique.

```
CREATE TABLE ma_table (  
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    data VARCHAR(100),  
    curdate TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE  
    CURRENT_TIMESTAMP  
);
```

## ❑ Suppression d'une table

```
DROP TABLE ma_table;
```

## ☐ Afficher la description d'une table

```
DESC ma_table;
```

## ☐ Suppression des données dans une table

```
TRUNCATE TABLE ma_table;  
DELETE FROM ma_table;
```

## ☐ Différence entre TRUNCATE TABLE et DELETE FROM

- TRUNCATE Table
  - remet l'AUTO\_INCREMENT à 0
- DELETE FROM
  - ne remet pas l'AUTO\_INCREMENT à 0
  - on peut mettre un filtre avec la clause WHERE



- ❑ **Ajouter une FOREIGN KEY** sur la colonne **ID\_COL** de la table **ma\_table**, qui référence la colonne **ID** de la table **ma\_table2**
  - Prérequis: la colonne ID de la table ma\_table2 doit être une **PRIMARY KEY**

```
ALTER TABLE ma_table ADD CONSTRAINT FOREIGN KEY (ID_COL)  
REFERENCES ma_table2(ID);
```

- ❑ **Supprimer une FOREIGN KEY** de la table **ma\_table**
  - La suppression est réalisée à partir du nom de la FOREIGN KEY

```
ALTER TABLE ma_table DROP FOREIGN KEY fk_name
```

# Insertion

## ❑ **commande insert:**

- Permet d'insérer des données dans une table

## ❑ **Exemple de table:**

```
CREATE TABLE Persons (  
    ID int,  
    NOM varchar(255),  
    PRENOM varchar(255)  
);
```

## ❑ **Requêtes d'insertions:**

```
INSERT INTO Persons (ID, NOM, PRENOM) values (1, 'Martin', 'Jean');  
INSERT INTO Persons (ID, NOM, PRENOM) values (2, 'Dupont', 'Paul');
```

## ❑ commande insert avec colonne **AUTO\_INCREMENT**:

- MySQL gère la valorisation de la colonne

```
CREATE TABLE Persons (  
    ID int NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    NOM varchar(255),  
    PRENOM varchar(255)  
);
```

## ❑ Requêtes d'insertions:

- La colonne ID ne doit pas apparaitre dans la requête d'insertion

```
INSERT INTO Persons (NOM, PRENOM) values ('Martin', 'Jean');  
INSERT INTO Persons (NOM, PRENOM) values ('Dupont', 'Paul');
```

## ☐ commande insert avec une colonne de type DATETIME:

```
CREATE TABLE Persons (  
    ID int NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    NOM varchar(255),  
    PRENOM varchar(255),  
    DATE_MODIF datetime  
);
```

## ☐ Requêtes d'insertions:

```
INSERT INTO Persons (NOM, PRENOM, DATE_MODIF)  
values ('Dupont', 'Paul', '2017-10-10 18:26:01');
```

**Ou**

```
INSERT INTO Persons (NOM, PRENOM, DATE_MODIF)  
values ('Dupont', 'Paul', '2017/10/10 18:26:01');
```

## ❑ Création d'une table avec la date/heure courante

- Pas besoin de renseigner la date dans la colonne curdate, c'est automatique.

```
CREATE TABLE ma_table (  
    id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
    data VARCHAR(100),  
    curdate TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE  
    CURRENT_TIMESTAMP  
);
```

## ❑ Insertion dans la table

```
INSERT INTO ma_table (DATA) values ('Bonjour');
```

## TP n°3: Insertion de données



# Lecture



❑ **commande SELECT permet :**

- de lire des données issues de la base de données, de 1 ou plusieurs tables
- de filtrer les données
- de sélectionner tout ou partie des colonnes
- de réaliser des calculs (somme, moyenne, comptage)
- d'extraire les données sous forme de tableau.

```
SELECT col1, col2 FROM ma_table
```

```
SELECT * FROM ma_table
```

## ❑ Exemple :

- Imaginons une table nommée CLIENT

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24
6	Grosjean	Romain	Nantes	37

- Pour sélectionner uniquement la colonne VILLE:

`SELECT ville FROM CLIENT`

VILLE
Nantes
Lille
Angers
Toulouse
Angers
Nantes

- ❑ Pour sélectionner les colonnes ID et VILLE de la table CLIENT

SELECT id, ville FROM CLIENT

ID	VILLE
1	Nantes
2	Lille
3	Angers
4	Toulouse
5	Angers
6	Nantes

- ❑ Pour sélectionner toutes les colonnes

SELECT \* FROM CLIENT

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24
6	Grosjean	Romain	Nantes	37

## ❑ Les clauses des commandes SELECT :

- Faire une jointure avec une autre table
- Trier les résultats sur une colonne donnée, ou plusieurs colonnes
- Filtrer pour ne sélectionner que certains enregistrements
- Faire une moyenne, un comptage ou une somme

```
SELECT col1, col2  
FROM ma_table  
WHERE condition  
GROUP BY expression  
HAVING condition  
ORDER BY col1
```

## ❑ Clause WHERE:

- Elle se place après la clause **FROM**
- Elle permet de filtrer les données de la requête
  - Utilisation d'opérateurs de comparaison
  - Opérateurs logiques
  - Niveaux de parenthèses dans la condition

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24
6	Grosjean	Romain	Nantes	37

- Exemple:

```
SELECT * FROM CLIENT WHERE AGE >= 35 AND VILLE='Nantes';
```

## ❑ Les opérateurs dans les conditions:

Opérateur	Description
=	Comparaison d'un égalité
<>, !=	Comparaison d'une différence
>	Comparaison de plus grand que
>=	Comparaison de plus grand ou égal que
<	Comparaison de plus petit que
<=	Comparaison de plus petit ou égal que
+	Effectue une addition
-	Effectue une soustraction
*	Effectue une multiplication
/	Effectue une division
IN	Permet de déclarer une liste énumérée de valeurs possibles
BETWEEN	Permet de définir une plage de valeurs autorisées
LIKE	Opérateur qui permet de rechercher une valeur approchée d'une chaîne de caractères
Opérateur	Description
AND, &&	Et
OR,	Ou
NOT, !	Non

## ❑ Opérateur AND:

- Correspond au ET logique dans une condition
- Equivalent de l'opérateur &&
- Exemple: je veux sélectionner tous les clients ayant un age supérieur ou égal à 35 ans et habitant Nantes:

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24
6	Grosjean	Romain	Nantes	37

```
SELECT * FROM CLIENT WHERE AGE >=35 AND VILLE='Nantes';
```

## ❑ Opérateur OR:

- Correspond au OU logique dans une condition
- Equivalent de l'opérateur ||
- Exemple: je veux sélectionner tous les clients ayant un age supérieur ou égal à 35 ans ou qui habitent Nantes:

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24
6	Grosjean	Romain	Nantes	37

```
SELECT * FROM CLIENT WHERE AGE >=35 OR VILLE='Nantes';
```



## ❑ Opérateur NOT:

- Pour inverser une condition ou une partie de condition
- Équivalent de l'opérateur !
- Exemple: je veux sélectionner tous les clients sauf ceux qui ont un age supérieur ou égal à 35 ans et qui habitent Nantes:

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24
6	Grosjean	Romain	Nantes	37

```
SELECT * FROM CLIENT WHERE NOT (AGE >=35 AND VILLE='Nantes');
```

```
SELECT * FROM CLIENT WHERE ! (AGE >=35 AND VILLE='Nantes');
```

## ❑ Opérateur !=:

- Opposé de =
- Equivalent de <>
- Exemple: je veux sélectionner tous les clients qui n'habitent pas Nantes:

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24
6	Grosjean	Romain	Nantes	37

```
SELECT * FROM CLIENT WHERE VILLE!= 'Nantes';
```

```
SELECT * FROM CLIENT WHERE VILLE<> 'Nantes';
```

## ❑ Opérateur IN:

- Pour indiquer une liste de valeurs possibles pour une colonne dans une condition
- Exemple:
  - Je veux sélectionner tous les clients qui habitent Nantes ou Angers:

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24
6	Grosjean	Romain	Nantes	37

```
SELECT * FROM CLIENT WHERE VILLE IN ('Nantes', 'Angers');
```

## ❑ Opérateur IN combiné avec NOT:

- Pour indiquer une liste de valeurs exclues
- Exemple:
  - Je veux sélectionner tous les clients qui n'habitent pas Nantes et Angers:

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24
6	Grosjean	Romain	Nantes	37

```
SELECT * FROM CLIENT WHERE NOT VILLE IN ('Nantes', 'Angers');  
SELECT * FROM CLIENT WHERE VILLE NOT IN ('Nantes', 'Angers');
```

## ❑ Opérateur **BETWEEN**:

- Pour indiquer une plage de valeurs autorisées (bornes incluses)
- S'utilise avec l'opérateur **AND**
- Utilisable pour les temporels (DATE, DATETIME, etc) et les numériques.
- Exemple:
  - Je veux sélectionner tous les clients qui ont entre 25 et 35 ans.

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24
6	Grosjean	Romain	Nantes	37

```
SELECT * FROM CLIENT WHERE AGE BETWEEN 25 AND 35;
```

## ❑ Opérateur **BETWEEN** sur une plage de dates:

- Exemple:
  - Je veux sélectionner tous les clients dont la date d'ouverture de compte a été réalisée entre le 1<sup>er</sup> janvier et le 31 mars.

ID	NOM	PRENOM	VILLE	AGE	DATE_OPEN
1	Parker	Tony	Nantes	32	2017-01-05
2	Benzema	Karim	Lille	18	2017-02-02
3	Batum	Nicolas	Angers	27	2017-04-14
4	Cadamuro	Louisa	Toulouse	28	2017-05-08
5	Décosse	Lucie	Angers	24	2017-05-25
6	Grosjean	Romain	Nantes	37	2017-06-16

```
SELECT * FROM CLIENT
```

```
WHERE date_open BETWEEN '2017-01-01' AND '2017-03-31';
```

## ❑ Opérateur LIKE:

- Permet de faire une recherche sur le contenu d'une chaîne de caractères
- Sensible à la casse ou non (dépend du COLLATE choisi)
- Exemple:
  - Je veux sélectionner tous les clients dont le prénom commence par la lettre « L ».

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24
6	Grosjean	Romain	Nantes	37

```
SELECT * FROM CLIENT WHERE PRENOM LIKE 'L%';
```

## ❑ SELECT DISTINCT:

- Permet de supprimer les doublons
- S'utilise dans une clause SELECT
- Exemple:
  - Je veux sélectionner les divers ID\_CLI de la table COMPTE à droite.

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24
6	Grosjean	Romain	Nantes	37

TYPE_COMPTE	ID_CLI	SOLDE
Compte courant	1	8 524,50
Livret A	1	35 327,00
Compte courant	2	15 040,00
Compte courant	3	-2 535,10
Livret A	4	2 527,00
Compte courant	4	12 800,00

```
SELECT DISTINCT ID_CLI FROM COMPTE;
```



## ❑ Clause **ORDER BY**:

- Permet de trier le tableau de résultat suivant une colonne.
- La clause **ORDER BY** se place après la clause **WHERE**
- Peut s'utiliser avec **ASC** (par défaut) ou **DESC**
- Exemple:

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24
6	Grosjean	Romain	Nantes	37

**SELECT \* FROM CLIENT ORDER BY NOM;**

**SELECT \* FROM CLIENT WHERE AGE >= 25 ORDER BY NOM DESC;**

## ❑ Clause IS NULL:

- Permet de sélectionner des enregistrements qui ont une valeur de colonne NULL.
- Exemple:

ID	NOM	PRENOM	TEL	VILLE	AGE
1	Parker	Tony	06 07 08 09 10	Nantes	32
2	Benzema	Karim		Lille	18
3	Batum	Nicolas	06 09 07 08 05	Angers	27
4	Cadamuro	Louisa	06 01 03 04 05	Toulouse	28
5	Décosse	Lucie		Angers	24
6	Grosjean	Romain	06 04 05 03 02	Nantes	37

```
SELECT * FROM CLIENT WHERE TEL IS NULL;
```

## ❑ Clause IS NOT NULL:

- Permet de sélectionner des enregistrements qui ont une valeur de colonne non NULL.
- Exemple:

ID	NOM	PRENOM	TEL	VILLE	AGE
1	Parker	Tony	06 07 08 09 10	Nantes	32
2	Benzema	Karim		Lille	18
3	Batum	Nicolas	06 09 07 08 05	Angers	27
4	Cadamuro	Louisa	06 01 03 04 05	Toulouse	28
5	Décosse	Lucie		Angers	24
6	Grosjean	Romain	06 04 05 03 02	Nantes	37

**SELECT \* FROM CLIENT WHERE TEL IS NOT NULL;**

## TP n°4: requêtes de base



## ❑ Les JOINTURES internes:

- Elles permettent de réaliser des **liaisons** entre tables
- Exemple: je veux extraire la liste des comptes qui appartiennent aux clients de plus de 20 ans.

CLIENT

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28

COMPTE

TYPE_COMPTE	ID_CLI	SOLDE
Compte courant	1	8 524,50
Livret A	1	35 327,00
Compte courant	2	15 040,00
Compte courant	3	-2 535,10
Livret A	4	2 527,00
Compte courant	4	12 800,00

SELECT \*  
FROM CLIENT, COMPTE  
WHERE **CLIENT.ID=COMPTE.ID\_CLI**  
AND AGE>=20



## ❑ Les JOINTURES internes:

- Attention, si un client n'a pas de compte alors il n'apparaîtra pas dans le tableau de résultats issu de la jointure interne.

CLIENT

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24

COMPTE

TYPE_COMPTE	ID_CLI	SOLDE
Compte courant	1	8 524,50
Livret A	1	35 327,00
Compte courant	2	15 040,00
Compte courant	3	-2 535,10
Livret A	4	2 527,00
Compte courant	4	12 800,00

```
SELECT *  
FROM CLIENT, COMPTE  
WHERE CLIENT.ID=COMPTE.ID_CLI  
AND AGE>=20
```

## ❑ Les JOINTURES internes avec l'opérateur JOIN:

- Attention, si un client n'a pas de compte alors il n'apparaîtra pas dans le tableau de résultats issu de la jointure interne.

CLIENT

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24

COMPTE

TYPE_COMPTE	ID_CLI	SOLDE
Compte courant	1	8 524,50
Livret A	1	35 327,00
Compte courant	2	15 040,00
Compte courant	3	-2 535,10
Livret A	4	2 527,00
Compte courant	4	12 800,00

SELECT \*  
FROM CLIENT

INNER JOIN COMPTE ON CLIENT.ID=COMPTE.ID\_CLI  
WHERE AGE>=20

## ❑ Les JOINTURES externes – LEFT JOIN:

- Elles permettent de réaliser des **liaisons** entre tables même si une table n'a pas de correspondance dans l'autre.
- Exemple: je veux extraire la liste des comptes qui appartiennent aux clients de plus de 20 ans. Si un client n'a pas de compte, je veux quand même avoir une ligne dans le tableau de résultat.

CLIENT

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24

COMPTE

TYPE_COMPTE	ID_CLI	SOLDE
Compte courant	1	8 524,50
Livret A	1	35 327,00
Compte courant	2	15 040,00
Compte courant	3	-2 535,10
Livret A	4	2 527,00
Compte courant	4	12 800,00

SELECT \* FROM CLIENT

LEFT JOIN COMPTE ON CLIENT.ID=COMPTE.ID\_CLI

WHERE AGE>=20



## ❑ Les JOINTURES externes – RIGHT JOIN:

- Permet de réaliser une jointure externe en partant de la table de droite dans l'expression du SELECT.

## ❑ GROUP BY:

- Permet de réaliser une opération sur une ou plusieurs colonnes données:
  - Comptage: COUNT
  - Somme: SUM
  - Moyenne: AVG
- Exemple: je veux calculer la somme possédée par chaque client sur l'ensemble de ses comptes.

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24

TYPE_COMPTE	ID_CLI	SOLDE
Compte courant	1	8 524,50
Livret A	1	35 327,00
Compte courant	2	15 040,00
Compte courant	3	-2 535,10
Livret A	4	2 527,00
Compte courant	4	12 800,00

```
SELECT NOM, SUM(SOLDE) FROM COMPTE, CLIENT
WHERE CLIENT.ID=COMPTE.ID_CLI
GROUP BY NOM
```

## ❑ GROUP BY:

- Contraintes: la liste des colonnes entre la clause SELECT et l'opération d'agrégation (ex: SUM) doit être identique à celle du GROUP BY

```
SELECT NOM, SUM(SOLDE) FROM COMPTE, CLIENT  
WHERE CLIENT.ID=COMPTE.ID_CLI  
GROUP BY NOM
```



```
SELECT NOM, SUM(SOLDE) FROM COMPTE, CLIENT  
WHERE CLIENT.ID=COMPTE.ID_CLI  
GROUP BY NOM, PRENOM
```



```
SELECT NOM, PRENOM, SUM(SOLDE) FROM COMPTE, CLIENT  
WHERE CLIENT.ID=COMPTE.ID_CLI  
GROUP BY NOM
```

## □ HAVING:

- Permet de réaliser un filtre sur les colonnes utilisées dans le GROUP BY ou sur le résultat de l'opération.
- Exemple: je veux sélectionner uniquement les clients qui ont au moins 2 comptes.

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24

TYPE_COMPTE	ID_CLI	SOLDE
Compte courant	1	8 524,50
Livret A	1	35 327,00
Compte courant	2	15 040,00
Compte courant	3	-2 535,10
Livret A	4	2 527,00
Compte courant	4	12 800,00

```
SELECT NOM, COUNT(*) FROM COMPTE, CLIENT
WHERE CLIENT.ID=COMPTE.ID_CLI
GROUP BY NOM HAVING COUNT(*)>=2
```

## □ HAVING:

- Autre exemple: je veux sélectionner uniquement les clients qui ont un solde global de tous leurs comptes supérieur ou égal à 10 000 €.

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24

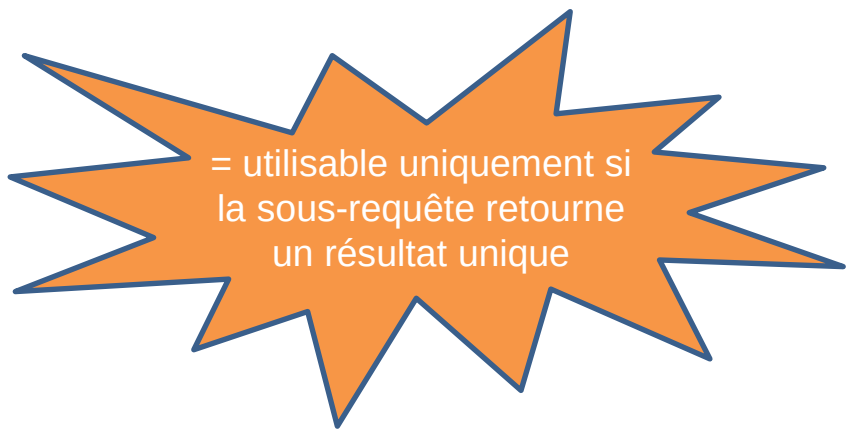
TYPE_COMPTE	ID_CLI	SOLDE
Compte courant	1	8 524,50
Livret A	1	35 327,00
Compte courant	2	15 040,00
Compte courant	3	-2 535,10
Livret A	4	2 527,00
Compte courant	4	12 800,00

```
SELECT NOM, SUM(SOLDE) FROM COMPTE, CLIENT  
WHERE CLIENT.ID=COMPTE.ID_CLI  
GROUP BY NOM HAVING SUM(SOLDE)>=10000
```

# Sous-Requêtes

## ❑ Il est possible de réaliser des sous-requêtes:

- Permettre de réaliser des requêtes du type: je veux des données de A mais avec une condition sur B.
- Ce type de requête peut la plupart du temps être réalisée avec une jointure mais ce n'est pas toujours le cas.
- Par exemple avec un opérateur **IN** ou **=**



= utilisable uniquement si  
la sous-requête retourne  
un résultat unique


```
SELECT *  
FROM bon  
WHERE ID_FOU IN (SELECT ID FROM fournisseur where nom like 'D%')
```

## ❑ Exemple d'une requête non réalisable avec une jointure classique:

- Je souhaite extraire la liste des bons de commande qui n'ont pas d'article.

```
SELECT *  
FROM bon  
WHERE ID NOT IN (SELECT distinct id_bon from compo)
```

⇒ Dans la sous-requête je recherche tous les identifiants de bons de commande qui sont dans la table compo.

-  S'ils sont dans la table compo c'est qu'ils ont forcément des articles.

⇒ Dans un second temps je recherche les bons de commandes de la table BON dont les ID n'existent pas dans le résultat de la sous-requête.



# Table virtuelle

- ❑ Il est possible de réaliser une jointure avec une table dite « virtuelle »
- ❑ Une table virtuelle est créée à partir du résultat d'une requête SQL.

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24

TYPE_COMPTE	ID_CLI	SOLDE
Compte courant	1	8 524,50
Livret A	1	35 327,00
Compte courant	2	15 040,00
Compte courant	3	-2 535,10
Livret A	4	2 527,00
Compte courant	4	12 800,00

```
SELECT *  
FROM  
    (SELECT * FROM CLIENT WHERE age >=25) VIRTUA,  
    COMPTE  
WHERE VIRTUA.ID=COMPTE.ID_CLI
```

## TP n°5: Sélection de données avancées



# Mise à jour

## ☐ **commande update:**

- Permet de modifier une ou plusieurs lignes d'une table
- S'utilise avec une clause de filtrage comme SELECT
- Cette clause de filtrage est facultative.

UPDATE nom\_table SET colonne1=valeur1, colonne2=valeur2  
WHERE condition;

## ☐ **Exemple sans filtre**

- **Attention sans clause de filtrage je mets à jour toutes les lignes de la table**

UPDATE Persons SET DATE\_MODIF='2017-11-11'

## ☐ **Exemple avec filtre**

UPDATE Persons SET DATE\_MODIF='2017-11-11' WHERE id=1

## ☐ Avec une sous-requête

- Il est possible de combiner un update avec une sous-requête.

```
UPDATE nom_table SET colonne1=valeur1, colonne2=valeur2  
WHERE id IN (select id from nom_table2 WHERE ...);
```

## ☐ **SAFE mode:**

- La commande UPDATE est considérée comme SAFE si la condition porte sur la clé primaire.
- Elle est bloquée dans le cas où la requête ne porte pas sur la clé primaire.

## ☐ **Désactivation du SAFE mode**

```
SET SQL_SAFE_UPDATES = 0;
```

## TP n°6: Mise à jour





# Suppression

## ❑ commande DELETE:

- Permet de supprimer une ou plusieurs lignes d'une table
- S'utilise avec une clause de filtrage comme SELECT
- Cette clause de filtrage est facultative.

**DELETE FROM** nom\_table  
**WHERE** condition;

## ❑ Exemple

**DELETE FROM** Persons **WHERE ID=9;**

## ❑ Importance du filtre

- Attention sans clause de filtrage je supprime toutes les lignes de la table

## ☐ **SAFE mode:**

- La commande DELETE est considérée comme SAFE si la condition porte sur la clé primaire.
- Elle est bloquée dans le cas où la requête ne porte pas sur la clé primaire.

## ☐ **Désactivation du SAFE mode**

```
SET SQL_SAFE_UPDATES = 0;
```

## ❑ DELETE et clé étrangère:

- Attention, vous n'avez pas le droit de supprimer des données parentes.

## ❑ Exemple

Supposons que j'ai une clé étrangère entre COMPTE et CLIENT

ID	NOM	PRENOM	VILLE	AGE
1	Parker	Tony	Nantes	32
2	Benzema	Karim	Lille	18
3	Batum	Nicolas	Angers	27
4	Cadamuro	Louisa	Toulouse	28
5	Décosse	Lucie	Angers	24

TYPE_COMPTE	ID_CLI	SOLDE
Compte courant	1	8 524,50
Livret A	1	35 327,00
Compte courant	2	15 040,00
Compte courant	3	-2 535,10
Livret A	4	2 527,00
Compte courant	4	12 800,00

**DELETE FROM CLIENT WHERE ID=1;**

## ❑ Importance du filtre

- Attention sans clause de filtrage je supprime toutes les lignes de la table

## TP n°7: Suppression



# Clé étrangère

## ❑ Définition:


- Une **clé étrangère** est une **contrainte entre 2 tables** qui permet de garantir l'intégrité des données.

## ❑ Exemple :

LIVRES			
ID	TITRE	AUTEUR	ID_EDI
1	Vingt mille lieues sous les mers	Jules Verne	1
2	Germinal	Emile Zola	1
3	Vipère au poing	Hervé Bazin	2

EDITEURS		
ID	NOM	ADRESSE
1	Le livre de poche	21 Rue du Montparnasse, 75006 Paris
2	Grasset	61 Rue des Saints-Pères, 75006 Paris



- La création d'une **clé étrangère** entre **LIVRES** (ID\_EDI) et **EDITEURS** (ID) permet d'empêcher les opérations suivantes:
- Supprimer un éditeur dans la table des éditeurs alors qu'il y a encore des livres associés dans la table des livres
  - Créer un livre associé à un identifiant éditeur qui n'existe pas.

## ❑ Ce que la clé étrangère n'est pas :


- La clé étrangère n'est pas une colonne. C'est une contrainte entre 2 colonnes appartenant à des tables différentes.

## ❑ Préalable :

LIVRES			
ID	TITRE	AUTEUR	ID_EDI
1	Vingt mille lieues sous les mers	Jules Verne	1
2	Germinal	Emile Zola	1
3	Vipère au poing	Hervé Bazin	2

EDITEURS		
ID	NOM	ADRESSE
1	Le livre de poche	21 Rue du Montparnasse, 75006 Paris
2	Grasset	61 Rue des Saints-Pères, 75006 Paris



- La colonne **ID** de la table **EDITEURS** doit être une clé primaire.
- Les colonnes **ID\_EDI** et **ID** doivent avoir les mêmes types (exemple INT(6) des 2 côtés)



## ❑ Exemple de création d'une clé étrangère:

```
ALTER TABLE LIVRES ADD CONSTRAINT FK_LIVRES_EDI FOREIGN KEY (ID_EDI)  
REFERENCES EDITEURS (ID);
```

```
CREATE TABLE LIVRES (  
    ID int NOT NULL,  
    TITRE varchar(100) NOT NULL,  
    AUTEUR varchar(100) NOT NULL,  
    ID_EDI int NOT NULL,  
    CONSTRAINT FK_LIVRES_EDI FOREIGN KEY (ID_EDI) REFERENCES EDITEURS (ID)  
);
```

## ❑ Exemple de suppression d'une clé étrangère:

```
ALTER TABLE LIVRES DROP FOREIGN KEY FK_LIVRES_EDI;
```

# Index

## ❑ Utilité d'un index:

- Structure de données qui permet d'accéder plus rapidement aux données d'une table.
- Un index est créé sur une colonne ou plusieurs colonnes sur lesquelles on effectue des recherches fréquentes

## ❑ Plusieurs types d'index

- BTREE: pour les opérations avec =, <, >, <=, >=, BETWEEN et LIKE
- HASH: pour les opérations = **uniquement**. Très rapide.
- BTREE est le type d'index par défaut

## ❑ Exemple:

- Si la table article est très volumineuse, la recherche des articles pour un fournisseur donné peut être assez longue.

```
SELECT * FROM article WHERE ID_FOU=28
```

- **Solution => création d'un index sur la colonne ID\_FOU**

## ❑ Création d'un index:

```
CREATE INDEX nom_index ON nom_table (col1, col2);
```

## ❑ Exemple:

```
CREATE INDEX iarticle ON article(id_fou);
```

## ❑ Suppression d'un index:

```
DROP INDEX nom_index ON nom_table;
```

## ❑ Exemple:

```
DROP INDEX iarticle ON article;
```

## ❑ Création d'un index de type HASH:

```
CREATE INDEX nom_index USING HASH ON nom_table (col1, ...);
```

## ❑ Exemple:

```
CREATE INDEX iarticle USING HASH ON article(id_fou);
```

## ❑ Création d'un index de type BTREE :

```
CREATE INDEX nom_index USING BTREE ON nom_table (col1, ...);
```

## ❑ Exemple:

```
DROP INDEX iarticle USING BTREE ON article;
```

# ANNEXES

## Gestion des utilisateurs

## ❑ Création d'un « utilisateur »:

- Par défaut: utilisateur root qui a tous les privilèges.
  - Conserver l'utilisateur root pour l'administrateur.
- Un utilisateur est un profil qui va être utilisé par une application pour accéder à la base de données.
- L'utilisateur reçoit des droits d'accès plus ou moins limité aux tables de la base de données

```
CREATE USER 'user1'@'localhost' IDENTIFIED BY 'myPwd1';
```

Paramètre	Description
user1	Nom de l'utilisateur
localhost	Machine depuis laquelle l'utilisateur est autorisé à se connecter
myPwd1	Mot de passe

## ❑ Création d'un « utilisateur »:

```
CREATE USER 'user1'@'%' IDENTIFIED BY 'myPwd1';
```

Paramètre	Description
user2	Nom de l'utilisateur
%	Connexion autorisée depuis n'importe quelle adresse IP
myPwd2	Mot de passe



## ❑ Définition des droits pour l'utilisateur:

- Exemple: j'autorise une application tierce à accéder à ma base mais uniquement en lecture.

```
CREATE USER 'lecture'@'localhost' IDENTIFIED BY 'lecturePwd';
```

```
GRANT SELECT ON ma_base.* TO 'lecture'@'localhost';
```

```
GRANT SELECT ON *.* TO 'lecture'@'localhost';
```

Type de permission	Description
ALL PRIVILEGES	Donne tous les droits à un utilisateur
CREATE	Donne les droits de création de bases et tables
DROP	Donne les droits de suppression de bases et tables
DELETE	Donne les droits de suppression de données dans les tables
INSERT	Donne les droits d'insertion de données dans les tables
SELECT	Donne les droits de lecture de données dans les tables
UPDATE	Donne les droits de modification de données dans les tables
GRANT OPTION	Donne les droits d'accorder des privilèges à d'autres utilisateurs

### ❑ **Suppression de droits utilisateur:**

- Exemple: je veux supprimer un droit particulier pour un utilisateur qui se connecte depuis localhost.

```
REVOKE [type of permission] ON ma_base.* FROM 'user1';
```

### ❑ **Suppression d'un utilisateur:**

```
DROP USER 'user1';
```

### ❑ **Recharge les privilèges :**

- A exécuter si une commande GRANT semble sans effet.

```
FLUSH PRIVILEGES;
```

❑ **Pour aller plus loin :**

- <https://dev.mysql.com/doc/refman/5.7/en/account-management-sql.html>