BAC Pro CIEL	Commandes SQL V1.1 20250527	Année 2025/2026

Table des matières

A. Connexion utilisateur

Pour se connecter à l'interface de base de données, on utilise la commande mysql telle que :

```
mysql –u <user> -p
```

avec :

- ⇨ mysql : la commande permettant d'exécuter le client
- ⇨ -u : l'option permettant de définir l'utilisateur avec lequel on va se connecter
- ⇨ <user> : le nom de l'utilisateur avec lequel on se connecte
- ⇨ -p : pour spécifier qu'un mot de passe sera à saisir pour se connecter avec cet utilisateur

Exemple :

```
mysql –u alainb –p
Enter password :
MariaDB[(none)]>
```

Dans cet exemple, je souhaite me connecter avec le nom d'utilisateur 'alainb' et ensuite je rentre le mot de passe. MariaDB m'a autorisé la connexion car j'ai le prompt 'MariaDB>' qui est apparu.

B. Déconnexion serveur mysql

La commande pour vous déconnectez de mysql est :

```
EXIT;
```

C. Généralités

SQL – Structured Query Language – Langage de requête structuré

Trois grands groupes de requêtes :

- ⇨ **Le Langage de Contrôle des Données (LCD – DCL Data Control Language)**

Cet ensemble de commande permet d'effectuer la gestion de la sécurité et des droits d'accès à la base de données grâce à GRANT, REVOKE, CREATE USER, ...

- ⇨ **Le Langage de Definition de Données (LDD – DDL Data Definition Language)**

Cet ensemble de commande permet de définir et modifier la structure de la base de données grâce à CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE DATABASE, ...

- ⇨ **Le Langage de Manipulation de Données (LMD – DML Data Manipulation Language)**

Cet ensemble de commande permet des travailler directement sur les données grâce à SELECT, INSERT INTO, UPDATE, DELETE, ...

Chaque bloc de commande doit finir par un point-virgule (;) pour que l'interpréteur SQL le comprenne.

D. Création d'utilisateur

```
CREATE USER 'nom_utilisateur'@'hôte' IDENTIFIED BY 'mot_de_passe';
```

Avec :

- ⇨ **nom_utilisateur** : le nom que l'utilisateur utilisera pour se connecter
- ⇨ **hôte** : spécifie d'où l'utilisateur peut se connecter. Cette valeur peut prendre en général 3 types d'arguments :
  - 'localhost' : Indique que l'utilisateur ne peut se connecter que depuis la machine où mariaDB s'exécute
  - '@IP' : Indique que l'utilisateur ne peut se connecter que depuis une adresse IP unique
  - '%' : Indique que l'utilisateur peut se connecter depuis partout (à utiliser avec prudence)
- ⇨ 'mot_de_passe' : Spécifie un mot de passe pour cet utilisateur

Exemple :

```
CREATE USER 'JeanMi'@'192.168.1.25' IDENTIFIED BY 'L0$cHe101225';
```

L'utilisateur JeanMi pourra se connecter sur MariaDB seulement depuis l'adresse IP 192.168.1.25 et en utilisant le mot de passe 'L0$cHe101225'.

```
CREATE USER 'abe@'%' IDENTIFIED BY 'potus';
```

L'utilisateur abe pourra se connecter sur MariaDB depuis n'importe où en utilisant le mot de passe 'potus'.

E. Suppression d'un utilisateur

```
DROP USER 'nom_utilisateur'@'hôte';
```

Avec :

- ⇨ **nom_utilisateur** : le nom que l'utilisateur à supprimer
- ⇨ **hôte** : L'hôte à partir duquel il se connecte

Exemple :

```
DROP USER 'alainb'@'%';
```

L'utilisateur alainb qui pouvait se connecter de n'importe où est supprimé.

F. Opération sur le compte utilisateur

```
ALTER USER 'nom_utilisateur'@'hôte' IDENTIFIED BY 'mot_de_passe';
```

Avec :

- ⇨ **nom_utilisateur** : Le nom d'utilisateur dont je souhaite apporter des modifications
- ⇨ **hôte** : Permet de modifier le lieu d'où peut se connecter l'utilisateur. Cette valeur pouvant prendre :
  - **'localhost'** : Indique que l'utilisateur ne peut se connecter que depuis la machine où mariaDB s'exécute
  - '**@IP**' : Indique que l'utilisateur ne peut se connecter que depuis une adresse IP unique
  - '**%**' : Indique que l'utilisateur peut se connecter depuis partout (à utiliser avec prudence)
- ⇨ 'mot_de_passe' : Modifier le mot de passe pour cet utilisateur où remettre celui qui était déjà en place

On peut renommer le nom d'utilisateur et son lieu de connexion :

```
ALTER USER 'nom_utilisateur'@'hôte' RENAME TO 'new_name'@'new_host';
```

Avec :

- ⇨ **nom_utilisateur** : Le nom d'utilisateur dont je souhaite apporter des modifications
- ⇨ **hôte** : Permet de modifier le lieu d'où peut se connecter l'utilisateur.
- ⇨ **new_name** : Le nouveau nom d'utilisateur
- ⇨ **new_host** : Le nouvel emplacement de connexion de l'utilisateur

On peut limiter le nombre de requête par heure par exemple :

```
ALTER USER 'alainb'@'%' WITH MAX_QUERIES_PER_HOUR 25;
```

Dans cet exemple, l'utilisateur alainb peut se connecter depuis n'importe où et ne peut effectuer que 25 requêtes par heure.

On peut limiter le nombre de connexion de l'utilisateur par heure :

```
ALTER USER 'alainb'@'%' WITH MAX_CONNECTIONS_PER_HOUR 2;
```

Dans cet exemple, l'utilisateur alainb peut se connecter depuis n'importe où et ne peut effectuer que 2 connexions maximum par heure.

On peut bloquer un compte utilisateur :

```
ALTER USER 'alainb'@'%' ACCOUNT LOCK;
```

Dans cet exemple, l'utilisateur alainb a son compte bloqué.

On peut débloquer un compte utilisateur :

```
ALTER USER 'alainb'@'%' ACCOUNT UNLOCK;
```

Dans cet exemple, l'utilisateur alainb a son compte débloqué

On peut spécifier une durée de validité du mot de passe du compte utilisateur :

```
ALTER USER 'alainb'@'%' PASSWORD EXPIRE INTERVAL 30 DAY;
```

Dans cet exemple, l'utilisateur alainb devra renouveler son mot de passe tous les 30 jours.

On peut désactiver le renouvellement de mot de passe du compte utilisateur :

```
ALTER USER 'alainb'@'%' PASSWORD EXPIRE NEVER;
```

Dans cet exemple, l'utilisateur alainb n'aura aucune expiration automatique du mot de passe.

On peut ajouter des commentaires à l'utilisateur :

```
ALTER USER 'alainb'@'%' COMMENT 'Cet utilisateur est le boss';
```

Dans cet exemple, je précise en commentaire que cet utilisateur est le boss.

ATTENTION : Lors de l'utilisation de la commande ALTER USER, il est important qu'à la fin de toutes modifications d'utiliser la commande :

```
FLUSH PRIVILEGES ;
```

Elle permet de mettre à jour les modifications effectuées.

G. Voir la liste des utilisateurs

Voir la liste de tous les utilisateurs du serveur de bases de données

```
SELECT User, Host FROM mysql.user;
```

H. Gestion des droits d'accès

On peut accorder tous les privilèges sur toutes les bases de données du serveur :

```
GRANT ALL PRIVILEGES ON *.* TO 'alainb'@'%';
```

Dans cet exemple, l'utilisateur alainb a accès à toutes les bases de données du serveur et peut y effectuer toutes les opérations sauf les commandes GRANT.

On peut accorder tous les privilèges et autoriser que l'utilisateur utilise les commandes GRANT :

```
GRANT ALL PRIVILEGES ON *.* TO 'alainb'@'%' WITH GRANT OPTION;
```

Dans cet exemple, l'utilisateur alainb à accès à toutes les bases de données du serveur et peut y effectuer toutes les opérations, y compris les commandes GRANT.

On peut accorder tous les privilèges sur une base de données du serveur :

```
GRANT ALL PRIVILEGES ON LYCEE.* TO 'alainb'@'%';
```

Dans cet exemple, l'utilisateur alainb a accès à la base de données LYCEE du serveur et peut y effectuer toutes les opérations

On peut accorder tous les privilèges sur une table d'une base de données du serveur :

```
GRANT ALL PRIVILEGES ON LYCEE.eleve TO 'alainb'@'%';
```

Dans cet exemple, l'utilisateur alainb a accès à la table eleve de la base de données LYCEE du serveur et peut y effectuer toutes les opérations.

On peut accorder des droits de lecture sur une base de données du serveur :

```
GRANT SELECT ON LYCEE.* TO 'alainb'@'%';
```

Dans cet exemple l'utilisateur alainb a l'autorisation de lecture sur toutes les tables de la base de données LYCEE.

On peut accorder des droits d'ajout de ligne de données sur une base de données du serveur :

```
GRANT INSERT ON LYCEE.* TO 'alainb'@'%';
```

Dans cet exemple, l'utilisateur alainb a l'autorisation d'insérer de nouvelles lignes de données sur toutes les tables de la base de données LYCEE.

On peut accorder des droits de modification de données existantes sur une base de données du serveur :

```
GRANT UPDATE ON LYCEE.* TO 'alainb'@'%';
```

Dans cet exemple, l'utilisateur alainb a l'autorisation de modifier des données sur toutes les tables de la base de données LYCEE.

On peut accorder des droits suppression de données sur une base de données du serveur :

```
GRANT DELETE ON LYCEE.* TO 'alainb'@'%';
```

Dans cet exemple, l'utilisateur alainb a l'autorisation de supprimer des donnée sur toutes les tables de la base de données LYCEE.

On peut accorder des droits de création de nouvelles tables dans une base de données du serveur :

```
GRANT CREATE TABLE ON LYCEE.* TO 'alainb'@'%';
```

Dans cet exemple, l'utilisateur alainb a l'autorisation de créer une nouvelle table dans la base de données LYCEE.

On peut accorder à utilisateur la création d'une nouvelle base de données sur le serveur :

```
GRANT CREATE DATABASE ON *.* TO 'alainb'@'%';
```

Dans cet exemple, l'utilisateur alainb a l'autorisation de créer une nouvelle base de données sur le serveur.

On peut accorder à utilisateur de créer de nouveaux utilisateurs sur le serveur :

```
GRANT CREATE USER ON *.* TO 'alainb'@'%';
```

Dans cet exemple, l'utilisateur alainb a l'autorisation de créer de nouveaux utilisateurs sur le serveur.

On peut accorder à l'utilisateur plusieurs privilèges :

```
GRANT SELECT,INSERT ON LYCEE.* TO 'alainb'@'%';
```

Dans cet exemple, l'utilisateur alainb a l'autorisation de lire et ajouter de nouvelles données dans la base de données LYCEE.

ATTENTION : Lors de l'utilisation de la commande GRANT, il est important qu'à la fin de toutes modifications d'utiliser la commande :
`FLUSH PRIVILEGES ;`
Elle permet de mettre à jour les modifications effectuées.

I. Voir les privilèges

La commande « SHOW GRANTS » permet de voir les privilèges de l'utilisateur en cours.

```
SHOW GRANTS;
```

```
+-----------------------------------------------------------------------------------------------------------------------+
| Grants for admin@%                                                                                                    |
+-----------------------------------------------------------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO `admin`@`%` IDENTIFIED BY PASSWORD '*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19' WITH GRANT OPTION |
+-----------------------------------------------------------------------------------------------------------------------+
```

Pour voir les privilèges d'un autre utilisateur :

```
SHOW GRANTS FOR 'alainb'@'%';
```

Dans cet exemple, je peux voir les privilèges de l'utilisateur alainb qui peut se connecter partout.

J. Voir la liste des bases de données

La commande « SHOW DATABASES » permet de voir la liste

```
SHOW DATABASES;
```

Cette commande permet de lister les noms des différentes bases de données présentes sur le serveur.

Exemple de résultat de cette commande :

```
+--------------------+
| Database           |
+--------------------+
| GestionIncident    |
| Mathematique       |
| information_schema |
| mysql              |
| performance_schema |
| phpmyadmin         |
| sys                |
+--------------------+
```

Dans cet exemple, on voit que le serveur possède 7 bases de données qui sont :

- GestionIncident
- Mathematique
- information_schema
- mysql
- performance_schema
- phpmyadmin
- sys

K. Créer une base de données

Pour créer une base de données simplement on utilise la commande suivante :

```
CREATE DATABASE LYCEE;
```

Dans cet exemple, on vient de créer une base de données nommée « LYCEE ».

Pour créer une base de données en toute sécurité sans erreur, il est possible d'indiquer que l'on crée une base de données seulement si elle n'existe pas déjà.

```
CREATE DATABASE LYCEE IF NOT EXIST;
```

Dans cet exemple, on crée la base de données « LYCEE » seulement si elle existe. Il n'y aura donc aucun message d'erreur si la base existe déjà.

L. Sélectionner une base de donnée

Si on possède plusieurs bases de données sur le serveur, à la connexion par défaut, l'utilisateur ne se trouve dans aucune base (NONE).

La commande « USE » permet de sélectionner la base de données sur laquelle on souhaite interagir.

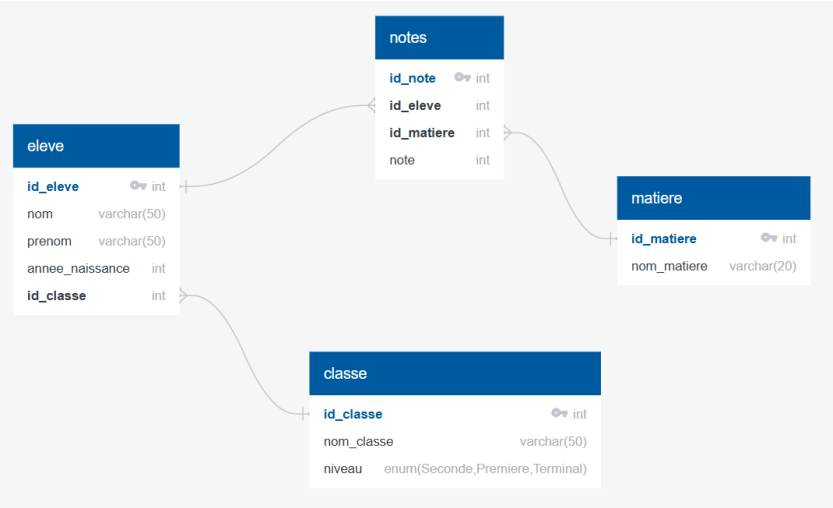
```
USE LYCEE;
```

Dans cet exemple, on sélectionne la base de données « LYCEE ». Le résultat est observable grâce au nouveau prompt :

```
MariaDB [LYCEE]>
```

M.Une base de donnée

Soit la base de données LYCEE suivante :



« eleve », « notes », « classe » et « matiere » sont des tables.

La table « eleve » par exemple possède 5 colonnes qui sont :

- ⇨ id_eleve
- ⇨ nom
- ⇨ prenom
- ⇨ annee_naissance
- ⇨ id_classe

La colonne id_eleve possède les paramètres suivants :

- ⇨ int : signifie que la colonne id_eleve ne pourra avoir comme valeur que des nombres entiers
- ⇨ [AI] : signifie qu'elle sera auto incrémentale. C'est-à-dire qu'à chaque fois qu'un nouvel élève sera ajouté à la table, la valeur id_eleve s'incrémentera automatiquement (1, 2, 3, ..., n)
- ⇨ PK : pour Primary Key signifie que id_eleve est la clé primaire de la table eleve. Elle pourra être utilisé dans d'autres tables.

Les colonnes nom et prenom possèdent le paramètre suivant :

- ⇨ varchar(50) : signifie que les colonnes nom et prenom ne pourront avoir comme valeur qu'une succession de caractère dans la limite de 50 caractères

La colonne annee_naissance possède le paramètre int signifiant que seul un nombre entier sera possible.

La colonne id_classe possède les paramètres suivants :

- ⇨ int : signifie que la colonne id_classe ne possèdera que des nombres entiers
- ⇨ FK : pour Foreign Key signifie que c'est une clé étrangère. La clé primaire associée à id_classe se trouve dans une autre table (table classe)

N. Création de table

Par exemple, si je souhaite créer la table « eleve ».

```
CREATE TABLE eleve(
      id_eleve INT AUTO_INCREMENT PRIMARY KEY,
      nom VARCHAR(50) NOT NULL,
      prenom VARCHAR(50) NOT NULL,
      annee_naissance INT,
      id_classe INT NOT NULL,
      FOREIGN KEY(id_classe) REFERENCES classe(id_classe)
);
```

Une fois la table créée on peut venir y ajouter par la suite des colonnes :

```
ALTER TABLE MaTable
ADD COLUMN mapremierecolonne;
```

O. Voir la liste des tables

Pour voir la liste des tables d'une base de données sélectionnée, la commande est :

```
SHOW TABLES;
```

```
MariaDB [LYCEE]> SHOW TABLES;
+----------------+
| Tables_in_LYCEE |
+----------------+
| classe          |
| eleve           |
| matieres        |
| notes           |
+----------------+
```

P. Supprimer un table

Pour supprimer une table, c'est la commande DROP :

```
DROP TABLES classe ;
```

Dans cet exemple, la table « classe » a été supprimée.

Q. Voir la liste des colonnes d'une table

La commande DESCRIBE, permet de voir la liste des colonnes d'une table.

Exemple avec la base de données LYCEE :

```
DESCRIBE classe;
```

```
+-----------+-----------------------------------+------+-----+---------+----------------+
| Field     | Type                              | Null | Key | Default | Extra          |
+-----------+-----------------------------------+------+-----+---------+----------------+
| id_classe | int(11)                           | NO   | PRI | NULL    | auto_increment |
| nom_classe| varchar(50)                       | NO   |     | NULL    |                |
| niveau    | enum('Seconde','Premiere','Terminal') | NO |     | NULL    |                |
+-----------+-----------------------------------+------+-----+---------+----------------+
```

```
DESCRIBE eleve;
```

```
+-----------------+-------------+------+-----+---------+----------------+
| Field           | Type        | Null | Key | Default | Extra          |
+-----------------+-------------+------+-----+---------+----------------+
| id_eleve        | int(11)     | NO   | PRI | NULL    | auto_increment |
| nom             | varchar(50) | NO   |     | NULL    |                |
| prenom          | varchar(50) | NO   |     | NULL    |                |
| annee_naissance | int(11)     | NO   |     | NULL    |                |
| id_classe       | int(11)     | NO   | MUL | NULL    |                |
+-----------------+-------------+------+-----+---------+----------------+
```

```
DESCRIBE matieres;
```

```
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| id_matiere  | int(11)     | NO   | PRI | NULL    | auto_increment |
| nom_matiere | varchar(50) | NO   |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
```

```
DESCRIBE notes;
```

```
+------------+---------+------+-----+---------+----------------+
| Field      | Type    | Null | Key | Default | Extra          |
+------------+---------+------+-----+---------+----------------+
| id_note    | int(11) | NO   | PRI | NULL    | auto_increment |
| id_eleve   | int(11) | NO   | MUL | NULL    |                |
| id_matiere | int(11) | NO   | MUL | NULL    |                |
| note       | int(11) | NO   |     | NULL    |                |
+------------+---------+------+-----+---------+----------------+
```

R. Insérer des données dans la base de données

On ajoute les données dans la table « classe ». La particularité de la colonne « niveau » est qu'elle ne peut prendre que 3 valeurs qui sont :

- « Seconde »
- « Premiere »
- « Terminal »

Si je souhaite ajouter les données dans cette table classe les 3 classes du BAC Pro CIEL par exemple :

```
INSERT INTO classe(nom_classe,niveau) VALUES
      ('535','Seconde'),
      ('635','Première'),
      ('735','Terminal');
```

S. Visualiser les données d'une table

Pour visualiser (lire) des données, la commande utilisée est SELECT.

Par exemple, pour visualiser la totalité des données de la table « classe » de la base de données LYCEE :

```
SELECT * FROM classe;
```

id_classe	nom_classe	niveau
1	535	Seconde
2	635	Premiere
3	735	Terminal

```
SELECT * FROM eleve;
```

id_eleve	nom	prenom	annee_naissance	id_classe
1	MICHEL	Nathan	2006	3
2	DUPONT	Camille	2007	2
3	LEFEVRE	Alice	2006	3
4	GIRARD	Chloé	2009	1
5	LEROY	Léa	2007	2
6	LAMBERT	Gabriel	2009	1
7	VINCENT	Hugo	2008	2
8	MOREAU	Emma	2009	1
9	OLIVIER	Arthur	2007	2
10	FOURNIER	Mathieu	2007	3
11	ROUX	Raphaël	2009	1
12	BERTRAND	Manon	2008	2
13	LEFEVRE	Sophie	2009	1
14	LAURENT	Théo	2006	3
15	CHEVALIER	Juliette	2007	2
16	SIMON	Clara	2007	3
17	GARCIA	Lucas	2009	1
18	MARCEL	Louis	2007	2
19	DUBOIS	Antoine	2008	1
20	LEMAIRE	Jade	2006	3

Si je souhaite ne visualiser que les élèves de seconde (id_classe=1), je le précise avec la commande WHERE :

```
SELECT * FROM eleve WHERE id_classe=1;
```

id_eleve	nom	prenom	annee_naissance	id_classe
4	GIRARD	Chloé	2009	1
6	LAMBERT	Gabriel	2009	1
8	MOREAU	Emma	2009	1
11	ROUX	Raphaël	2009	1
13	LEFEVRE	Sophie	2009	1
17	GARCIA	Lucas	2009	1
19	DUBOIS	Antoine	2008	1

Je peux également préciser que je veux afficher la liste de tous les élèves avec leur nom de classe associée sous forme de tableau à 3 colonnes (nom, prénom, nom de la classe) :

```
SELECT
      eleve.nom,
      eleve.prenom,
      classe.nom_classe
FROM
      eleve
JOIN
      classe ON eleve.id_classe = classe.id_classe;
```

nom	prenom	nom_classe
GIRARD	Chloé	535
LAMBERT	Gabriel	535
MOREAU	Emma	535
ROUX	Raphaël	535
LEFEVRE	Sophie	535
GARCIA	Lucas	535
DUBOIS	Antoine	535
DUPONT	Camille	635
LEROY	Léa	635
VINCENT	Hugo	635
OLIVIER	Arthur	635
BERTRAND	Manon	635
CHEVALIER	Juliette	635
MARCEL	Louis	635
MICHEL	Nathan	735
LEFEVRE	Alice	735
FOURNIER	Mathieu	735
LAURENT	Théo	735
SIMON	Clara	735
LEMAIRE	Jade	735

Explication de la commande précédente :

```
SELECT
      eleve.nom,
      eleve.prenom,
      classe.nom_classe
```

Cette partie indique que je veux afficher 3 colonnes :

- ⇨ eleve.nom qui correspond à la colonne « nom » de la table « eleve »
- ⇨ eleve.prenom qui correspond à la colonne « prenom » de la table « eleve »
- ⇨ classe.nom_classe qui correspond à la colonne « nom_classe » de la table « classe » dont on va spécifier les paramètres plus loin dans « JOIN ».

```
FROM
      eleve
```

Cette partie indique que je récupère ces données dans la table eleve.

```
JOIN
      classe ON eleve.id_classe = classe.id_classe;
```

Cette partie indique que l'on va joindre (liaison) la table « classe » à la table « eleve ». Et que l'on va indiquer que les valeurs eleve.id_classe seront égales aux valeurs de classe.id_classe.

Afficher la liste des élèves par ordre alphabétique :

```
SELECT
      nom,
      prenom
FROM
      eleve
ORDER BY
      nom ASC;
```

nom	prenom
BERTRAND	Manon
CHEVALIER	Juliette
DUBOIS	Antoine
DUPONT	Camille
FOURNIER	Mathieu
GARCIA	Lucas
GIRARD	Chloé
LAMBERT	Gabriel
LAURENT	Théo
LEFEVRE	Sophie
LEFEVRE	Alice
LEMAIRE	Jade
LEROY	Léa
MARCEL	Louis
MICHEL	Nathan
MOREAU	Emma
OLIVIER	Arthur
ROUX	Raphaël
SIMON	Clara
VINCENT	Hugo

Dans notre cas, le tri a bien été réalisé mais on a Alice et Sophie LEFEVRE qui ont le même nom et ne sont pas bien triés. On va donc rajouter le tri par prénom.

```
SELECT
      nom,
      prenom
FROM
      eleve
ORDER BY
      nom ASC,
      prenom ASC;
```

nom	prenom
BERTRAND	Manon
CHEVALIER	Juliette
DUBOIS	Antoine
DUPONT	Camille
FOURNIER	Mathieu
GARCIA	Lucas
GIRARD	Chloé
LAMBERT	Gabriel
LAURENT	Théo
LEFEVRE	Alice
LEFEVRE	Sophie
LEMAIRE	Jade
LEROY	Léa
MARCEL	Louis
MICHEL	Nathan
MOREAU	Emma
OLIVIER	Arthur
ROUX	Raphaël
SIMON	Clara
VINCENT	Hugo

T. Supprimer des données

La suppression d'une ou plusieurs données dans une base de données s'effectuent grâce à la commande DELETE.

```
DELETE FROM eleve WHERE id_eleve = 5;
```

Dans l'exemple de cette commande, j'ai supprimé un élève de la table « eleve » qui était rattaché à id_eleve de 5.

```
DELETE FROM eleve WHERE nom = "DUBOIS";
```

Dans cet exemple, j'ai supprimé l'élève portant le nom « DUBOIS » de la table « eleve ». L'inconvénient de cette commande, c'est que si plusieurs élèves portent le même nom, ils seront tous supprimer. Par exemple dans notre table élève nous avons deux élèves « LEFEVRE », il y a Sophie et Alice. Si nous faisons la commande suivante :

```
DELETE FROM eleve WHERE nom = "LEFEVRE";
```

Nous supprimons les élèves Sophie et Alice.

Si nous souhaitions supprimer seulement Alice LEFEVRE, on entrerait la commande suivante :

```
DELETE FROM eleve WHERE nom = "LEFEVRE" AND prenom = "Alice";
```

Si nous souhaitons supprimer tous les élèves d'une classe on peut également :

```
DELETE FROM eleve WHERE id_classe = 1;
```

Ou encore, si on ne connait pas forcément le id_classe et seulement le nom de la classe :

```
DELETE FROM eleve
     WHERE id_classe
     IN (SELECT id_classe FROM classe WHERE nom_classe="735");
```

Si l'on souhaite supprimer la totalité des données de la table « eleve » (NON RECOMMANDÉ) :

```
DELETE FROM eleve;
```

U. Modifier des données

Lorsque j'ai rentré l'élève Gabriel LAMBERT, je me suis trompé, il s'appelle Jack. Je dois donc changer son prénom de Gabriel en Jack.

id_eleve	nom	prenom	annee_naissance	id_classe
1	MICHEL	Nathan	2006	3
2	DUPONT	Camille	2007	2
3	LEFEVRE	Alice	2006	3
4	GIRARD	Chloé	2009	1
5	LEROY	Léa	2007	2
6	LAMBERT	Gabriel	2009	1
7	VINCENT	Hugo	2008	2
8	MOREAU	Emma	2009	1
9	OLIVIER	Arthur	2007	2
10	FOURNIER	Mathieu	2007	3
11	ROUX	Raphaël	2009	1
12	BERTRAND	Manon	2008	2
13	LEFEVRE	Sophie	2009	1
14	LAURENT	Théo	2006	3
15	CHEVALIER	Juliette	2007	2
16	SIMON	Clara	2007	3
17	GARCIA	Lucas	2009	1
18	MARCEL	Louis	2007	2
19	DUBOIS	Antoine	2008	1
20	LEMAIRE	Jade	2006	3

```
UPDATE eleve
      SET prenom = "Jack"
      WHERE id_eleve=6;
```

On a désormais le résultat suivant :

id_eleve	nom	prenom	annee_naissance	id_classe
1	MICHEL	Nathan	2006	3
2	DUPONT	Camille	2007	2
3	LEFEVRE	Alice	2006	3
4	GIRARD	Chloé	2009	1
5	LEROY	Léa	2007	2
6	LAMBERT	Jack	2009	1
7	VINCENT	Hugo	2008	2
8	MOREAU	Emma	2009	1
9	OLIVIER	Arthur	2007	2
10	FOURNIER	Mathieu	2007	3
11	ROUX	Raphaël	2009	1
12	BERTRAND	Manon	2008	2
13	LEFEVRE	Sophie	2009	1
14	LAURENT	Théo	2006	3
15	CHEVALIER	Juliette	2007	2
16	SIMON	Clara	2007	3
17	GARCIA	Lucas	2009	1
18	MARCEL	Louis	2007	2
19	DUBOIS	Antoine	2008	1
20	LEMAIRE	Jade	2006	3