# Introduction aux Bases de Données

©Sovanna Tan
sovanna.tan@u-pec.fr

Septembre 2021

# Organisation du cours

- 5 séances de Cours Magistraux (CM) en distanciel ou en amphi
- 5 séances de Travaux sur Machines : `MySQL` et `phpMyAdmin`
- Évaluation en Contrôle Continu Intégral :
  - 2 contrôles continus : la meilleur note compte (mécanisme de deuxième chance)
  - 1 mini-projet à faire à la maison avec une soutenance

# Contenu du cours

## Le langage SQL (Structured Query Language)

- Créer, détuire une base de données, une table.
- Insérer, mettre à jours des éléments
- Rechercher des éléments dans une ou plusieurs tables

# Bibliographie


- *SQL 5$^{e}$ édition*, Ryan Stephens, Ronald Plew, Arie Jones, Person, 2012
  titre original *Sams Teach Yourself SQL in 24 Hours*
- `https://dev.mysql.com/doc/`

# Problématique

## Définition

Une base de données est un ensemble structuré de données.

Objectifs

- Assurer la persistance des données.
- Éliminer la redondance des données.
- Partager les données entre tous les utilisateurs.
- Incorporer les modifications facilement et rapidement.
- Simplifier l'utilisation de fichiers de données (ex : fichier généré par un tableur).
- Minimiser le coût du stockage et de l'accès aux données.
- Améliorer l'intégrité, la cohérence et la précision des données.
- Assurer la confidentialité des données : sécuriser l'accès et empêcher les accès non autorisés.

# Historique

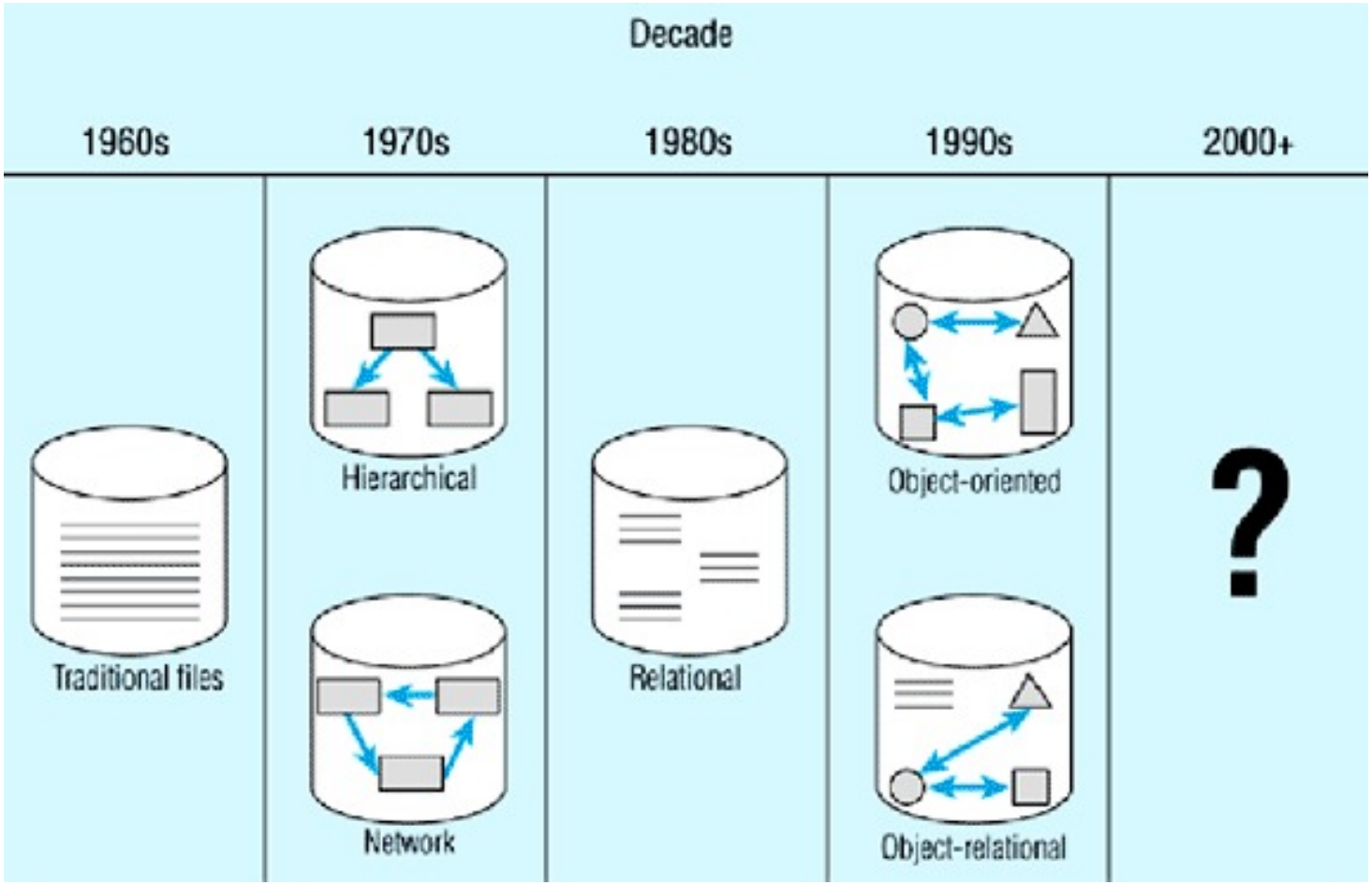



image provenant de `http://handledata.blogspot.fr/2012/07/dbms-types-illustration.html`

# Base de données réseau

- 1963 : Integrated Data Store de Charles W. Bachman (Turing Award en 1973)
- 1968 : COBOL

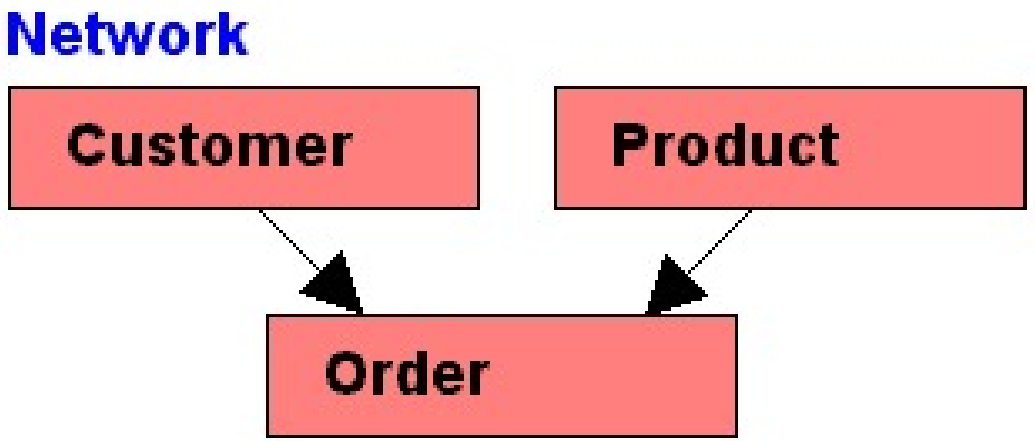



image provenant de `http://encyclopedia2.thefreedictionary.com/DBMS`

# Base de données hiérachique

- 1966 : Information Management System d'IBM crée pour Apollo

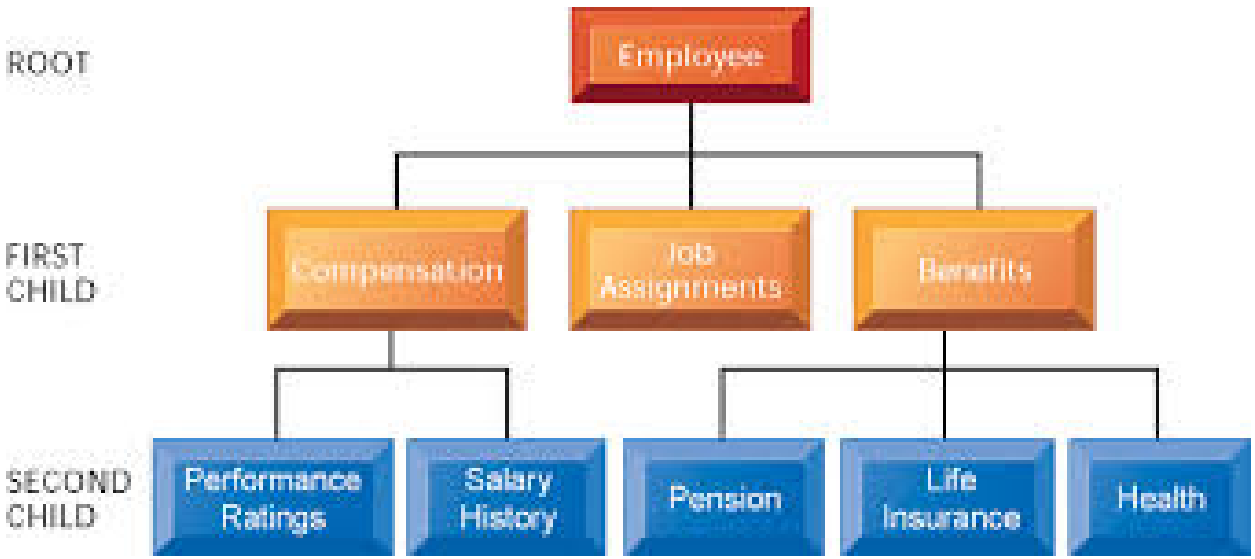



image provenant de `http://wps.pearsoned.co.uk/wps/media/objects/14157/14497116/Learning_Tracks/Ess10_CH05_LT3_Hierarchical_and_Network_Data_Models.pdf`

# Base de données relationnelle

- 1970 : Edgar F. Codd (IBM, Turing Award en 1981) publie le modèle relationnel qui utilise la théorie des ensembles.
- IBM développe le langage SEQUEL (Structured English Query Language) qui deviendra SQL.
- 1979 : Oracle sort la première version commerciale de SQL.
- 1982 : IBM sort DB2.

**School Table**

| ID | Name |
|---|---|
| S001 | University of Technology |
| S002 | University of Applied Science |

**Student Table**

| School ID | ID | Name | DOB |
|---|---|---|---|
| S001 | UT-1000 | Tommy | 05/06/1995 |
| S001 | UT-1000 | Better | 16/04/1995 |
| S002 | UAS-1000 | Linda | 02/09/1995 |
| S002 | UAS-1000 | Jonathan | 22/06/1995 |

image provenant de https://www.visual-paradigm.com/tutorials/databasedesign.jsp

# La notion d'ensemble

- Un **ensemble** est une collection d'objets distincts qui satisfont certaines propriétés. Chacun de ces objets est un **élément** de l'ensemble.
  - ex : $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$
- La définition de l'ensemble doit nous permettre de répondre à la question : "est-ce que cet objet appartient à l'ensemble ?".
- Définition d'un ensemble **en extension** : on donne la liste des éléments entre $\{$ et $\}$.
  - ex : $\{0, 1, 2, 3\}$
- Définition d'un ensemble **en compréhension** : on donne un propriété telle que tout objet qui la vérifie appartient à l'ensemble et tout objet qui ne la vérifie pas n'appartient pas à l'ensemble.
  - ex : $\{x \in \mathbb{N} \mid x \leq 4\}$

# La notion de relation

- Une **relation** décrit un lien entre plusieurs objets.
- Une **relation binaire** est définie par son ensemble de départ, son ensemble d'arrivée et une règle qui associe certains éléments de l'ensemble de départ à l'ensemble d'arrivée.

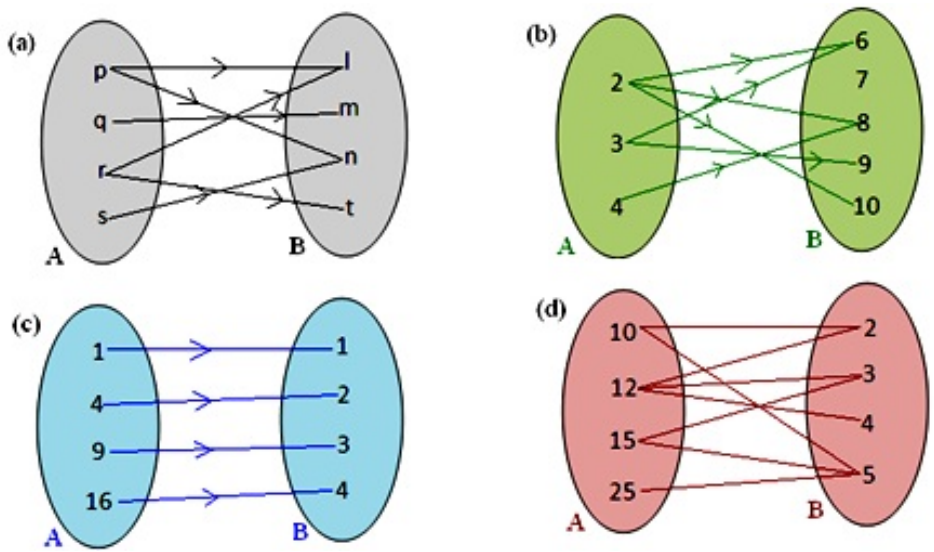
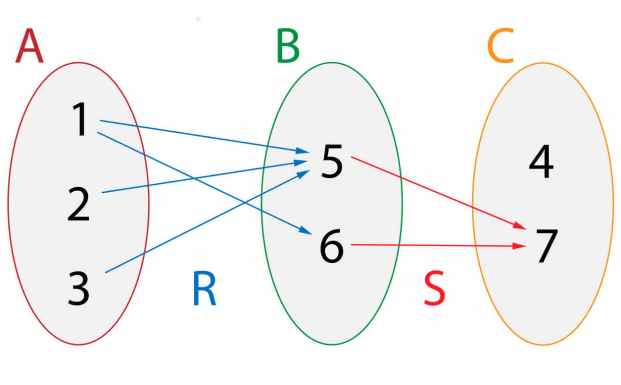



image provenant de `https://www.math-only-math.com/worksheet-on-math-relation.html`

# Représentation d'une relation sous forme de table

- Une **relation** *n*-**aire** met en relation les éléments de *n* ensembles.




| A | B | C |
|---|---|---|
| 1 | 5 | 7 |
| 1 | 6 | 7 |
| 2 | 5 | 7 |
| 3 | 5 | 7 |

- Chaque ligne de la table est unique.
- Les lignes peuvent être données dans n'importe quel ordre.

image provenant de `https://en.wikipedia.org/wiki/Book:Introduction_to_Order_Theory`

# Structure d'une base de données relationnelle

- Une base de données relationnelle est constituée de **tables** ou **relations**.
- Les colonnes d'une table sont appelées **attributs** ou **champs** de la relation.
- Les valeurs des attributs sont des *valeurs atomiques*.
- Le **schéma de la relation** $R$ est défini par le nom de la relation $R$ et de ses attributs $A_1$, $A_2$, ..., $A_n$ et est noté $R(A_1, A_2, ..., A_n)$.
- Le **schéma d'une base de données** est l'ensemble des schémas des relations qui la compose.
- On appelle **domaine** de l'attribut l'ensemble des valeurs que peut prendre un attribut.

# Éléments d'une base de données relationnelle

- Une ligne ou entrée d'une table est un n-uplet appelé **élément** de la relation et est une **donnée** ou **enregistement** de la base de données.
- Une **relation** est un ensemble de n-uplets. Il n'y a pas d'éléments en double.
- Exemple : la table Students

| id | firstName | dateOfBirth |
|---|---|---|
| 1 | John | 1996-03-09 |
| 2 | Mary | 1996-04-15 |
| 3 | Linda | 1996-08-22 |
| 4 | Jonathan | 1996-09-17 |
| 5 | Mary | 1996-10-07 |
| 6 | Linda | 1996-08-22 |

# Valeur NULL

- La valeur NULL en base de données représente une valeur inconnue ou manquante.
- On la teste avec un opérateur spécifique (différent des opérateur de comparaison $=$ et $<>$) : **IS NULL** ou **IS NOT NULL**.

# Créer la table avec SQL

```
CREATE TABLE Students (
   id   INT NOT NULL AUTO_INCREMENT,
   firstName VARCHAR(20),
   dateOfBirth DATE,
  PRIMARY KEY(id)
);
```

- Les commandes **DESC** ou **SHOW CREATE TABLE** permettent de voir la structure d'une table.

```
mysql> DESC Students;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| id          | int(11)     | NO   | PRI | NULL    | auto_increment |
| firstName   | varchar(20) | YES  |     | NULL    |                |
| dateOfBirth | date        | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
mysql> SHOW CREATE TABLE Students;
+----------+-------------------------------------------------------------+
| Students | CREATE TABLE `Students` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `firstName` varchar(20) DEFAULT NULL,
  `dateOfBirth` date DEFAULT NULL,
  PRIMARY KEY (`id`)
+----------+-------------------------------------------------------------+
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8 |
```

# Le codage des caractères en informatique

- Dans un ordinateur, les informations sont stokées en base 2 avec des 0 et des 1 (bit pour BInary digiT).
- Chaque caractère est représenté par un nombre entier.
- Code ASCII sur 7 bits
  - A-Z : 65 à 90
  - a-z : 97 à 122
- Codes ISO sur 8 bits (ISO/CEI 8859-1 ou ISO latin-1, ISO/CEI 8859-15 ou ISO latin-9, ...)
- UNICODE et ISO/CEI 10646 : systèmes de codage universel pour tout système d'écriture. Ils donnent des codes de longueur variable (ex : UTF-8 sur 8 ou 16 bits) ou des codes de longueur fixe jusqu'à 32 bits.

# Les types des attributs chaînes de caractères

| Type SQL | Description |
|---|---|
| **CHAR** ou CHARACTER | chaîne de caractères de longueur fixe |
| **VARCHAR** ou CHARACTER VARYING | chaîne de caractères de longueur variable |
| **NCHAR** ou NATIONAL CHARACTER | chaîne de caractères UNICODE de longueur fixe |
| **NVARCHAR** ou NATIONAL CHARACTER VARYING | chaîne de caractères UNICODE de longueur variable |
| **TEXT** | chaîne de caractères longue de longueur indéterminée |

- On précise la taille des données en nombre d'octets sauf pour le type TEXT.
  - ex :**VARCHAR(20)**

# Le type ENUM

- Avec le type ENUM, on restreint les valeurs d'un attribut à une liste de valeurs autorisées.
- Si on insère, une valeur non autorisée, la chaîne vide '' est insérée.

```
CREATE TABLE shirts (
    name VARCHAR(40),
    size ENUM('x-small', 'small', 'medium', 'large', 'x-large')
);
```

# Le codage des entiers en informatique

- Les valeurs des entiers codables dépendent du processeur. Un entier est codé sur 64 bits pour un processeur 64 bits, 32 bits pour un processeur 32 bits, etc. Le processeur et le langage de programmation influent sur les valeurs des plus petit et plus grand entiers représentables.
- Les entiers positifs sont codés avec leur valeur en base 2 :

| | |
|---|---|
| 0 | 0...000 |
| 1 | 0...001 |
| 2 | 0...010 |
| 3 | 0...011 |

- Les entiers négatifs sont représentés en complément à deux :

| | |
|---|---|
| -1 | 1...111 |
| -2 | 1...110 |
| -3 | 1...101 |
| -4 | 1...100 |

# Les types des attributs numériques à valeur exacte

| Type SQL | Description | Étendue |
|---|---|---|
| **BIT** | bit | de 1 à 64 bits (BIT(1) à BIT(64)) |
| **TINYINT** | entier codé sur 1 octet | $[-128, 127]$ ou $[0, 255]$ |
| **SMALLINT** | entier codé sur 2 octets | $[-2^{15}, 2^{15}-1]$ ou $[0, 2^{16}-1]$ |
| **MEDIUMINT** | entier codé sur 3 octets | $[-2^{23}, 2^{23}-1]$ ou $[0, 2^{23}-1]$ |
| **INT** | entier codé sur 4 octets | $[-2^{31}, 2^{31}-1]$ ou $[0, 2^{31}-1]$ |
| **BIGINT** | entier codé sur 8 octets | $[-2^{63}, 2^{63}-1]$ ou $[0, 2^{63}-1]$ |
| **DECIMAL** ou **NUMERIC** | nombre décimal avec représentation exacte | DECIMAL(5,2) $[-999.99, 999.99]$ précision : 5, échelle : 2 |

- Pour MySQL, le type **BOOLEAN** correspond à TINYINT(1). Il prend deux valeurs :
  - 0 correspond à false ;
  - une valeur non nulle correspond à true.

# Le codage des nombres flottants en informatique

- Nombre normalisé : $\pm 0.\text{Mantisse}\ 10^{\text{Exposant}}$
  - ex :5420.11 $= +0.542011\ 10^4$
- On représente le signe, la mantisse et l'exposant an base 2 sur 32, 64, 128 ou 256 bits suivant les processeurs et les langages de programmation.

# Les types des attributs numériques à valeur approchée

| Type SQL | Description |
|---|---|
| **FLOAT** | nombre flottant codé sur 4 octets |
| **DOUBLE** | nombre flottant codé sur 8 octets |

- Les valeurs utilisées sont des valeurs approchées. Il faut faire attention lorsqu'on fait des comparaisons avec ces types de valeurs.

# Les types des attributs temporels

| Type SQL | Description | Étendue |
|---|---|---|
| **DATE** | date | de '1000-01-01' à '9999-12-31' |
| **DATETIME** | date et heure | de '1000-01-01 00 :00 :00' à '9999-12-31 23 :59 :59' |
| **TIMESTAMP** | horodatage UTC | de '1970-01-01 00 :00 :01' à '2038-01-19 03 :14 :07' |
| **TIME** | durée | de '-838 :59 :59.000000' à '838 :59 :59.000000' |

- Si on insére la valeur NULL pour un attribut de type DATETIME ou TIMESTAMP, la date et l'heure courante ou l'horodatage courant est inséré.
- Un attribut de type TIMESTAMP peut avoir la propriété d'être mis à jour à chaque modification de l'enregistrement.

# Insérer des éléments

- en spécifiant toutes les valeurs des attributs ;

```
INSERT INTO Students VALUES
   (NULL, 'John', '1996-03-09'),
   (NULL, 'Mary', '1996-04-15'),
   (NULL, 'Mary', '1996-10-07');
INSERT INTO Students VALUES (14, 'Linda', '1996-08-22');
```

- en ne donnant que les valeurs connues.

```
INSERT INTO Students(firstName) VALUES ('Marc');
```

```
mysql> SELECT * from Students;
+----+-----------+-------------+
| id | firstName | dateOfBirth |
+----+-----------+-------------+
|  1 | John      | 1996-03-09  |
|  2 | Mary      | 1996-04-15  |
|  3 | Mary      | 1996-10-07  |
| 14 | Linda     | 1996-08-22  |
| 15 | Marc      | NULL        |
+----+-----------+-------------+
```

# Effacer un élément dont on connaît la clé

```
DELETE FROM Students WHERE id=2;
```

```
mysql> SELECT * from Students;
+----+-----------+-------------+
| id | firstName | dateOfBirth |
+----+-----------+-------------+
|  1 | John      | 1996-03-09  |
|  3 | Mary      | 1996-10-07  |
| 14 | Linda     | 1996-08-22  |
| 15 | Marc      | NULL        |
+----+-----------+-------------+
```

# Manipuler une base de données MySQL

- Créer une base :

  **CREATE DATABASE** Courses ;

- Utiliser la base (spécifique MySQL et SQL Server) :

  **USE** Courses ;

- Effacer la base :

  **DROP DATABASE** Courses ;

# Voir la liste des tables

**SHOW TABLES**;

```
mysql> SHOW TABLES;
+--------------------+
| Tables_in_courses  |
+--------------------+
| Students           |
+--------------------+
```

# Effacer une table et son contenu

```
DROP TABLE Students;
```

```
mysql> show TABLES;
Empty set (0,00 sec)
```