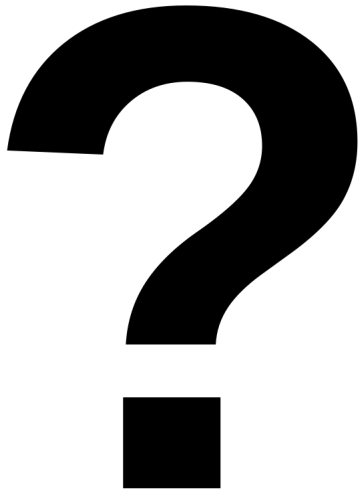

# Bases de données

## Vocabulaires, requêtes et applications

Anicet E. T. Ebou, ediman.ebou@inphb.ci

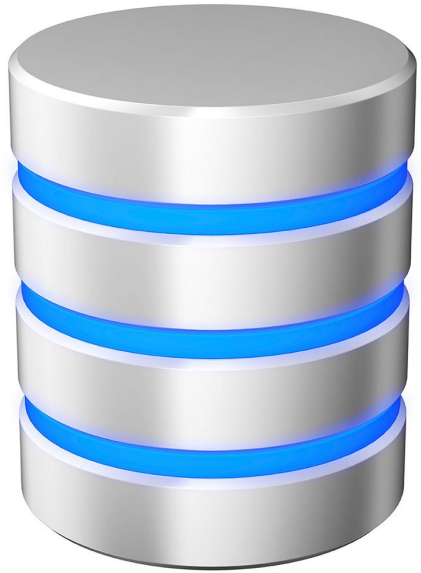
# Le problème du stockage des données

- Ou stocker les données dont j’ai besoin pour mon programme?
- Ou stocker les résultats de mes programmes?
- D’où la nécessité des systèmes de gestion de base de données (SGBD).



# Les SGBD à la rescousse!

- Indépendance physique: Possibilité de modifier les structures de stockage sans modifier les programmes.
- Indépendance logique: Possibilité d'ignorer les données d'autres applications.
- Manipulation aisée par le biais d'un langage déclaratif (SQL, Structured Query Language).
- Conception aisée par le biais de méthodes utilisant des modèles simples à manipuler (Modèle Entité/Association).



# 01

# Vocabulaire

# Définitions

- **Domaine**: type (c'est-à-dire entier, flottant, chaîne de caractère ou autre) des données qui seront stockées dans une colonne de la table considérée.
- **Attribut**: Couple nom/domaine qui permet de définir une colonne. Par exemple Nom: String ou Age: Int, etc.
- **Variable de relation**: Ensemble ordonné d'attributs (sert d'entête à une table ou « relation »). Par exemple (Nom: String, Age: Int, Adresse: String).
- **Tuple**: Ensemble ordonné de couples attribut/valeur (ligne). Par exemple (Nom: Trump, Age: 70, Adresse: WhiteHouse).

# Définitions

- **Relation**: Ensemble de tuples, soit une table en SQL

Orders

| order_id | item | amount | customer_id |
|---|---|---|---|
| 1 | Keyboard | 400 | 4 |
| 2 | Mouse | 300 | 4 |
| 3 | Monitor | 12000 | 3 |
| 4 | Keyboard | 400 | 1 |
| 5 | Mousepad | 250 | 2 |

# 02

# Clé primaire, clé étrangère

# Notion de clé primaire

Une clé de relation est un groupe d'attributs qui détermine un tuple unique dans une relation. Dans un SGBD, pour chaque relation, une clé (la plus simple possible) est choisie et est appelée **clé primaire** de la relation.

Notation: en général, on souligne l'attribut choisi pour clé primaire lors de la déclaration d'une table dans un schéma relationnel.

```
COURS    (NC:int, CODE_COURS:string, INTITULE:string)
ETUDIANT (NE:int, NOM:string, PRENOM:string)
INSCRIT  (NE:int, NC:int, ANNEE:int)
```

# Notion de clé primaire

Dans l’exemple précédent, NC est une clé pour la table COURS (représente le « numéro » du cours) alors que NE en est une pour la table ETUDIANT.

En revanche, il n’existe pas de clé mono-attribut pour la table INSCRIT.

C’est particulièrement utile pour toutes les problématiques d’indexation (pour retrouver facilement une ligne donnée dans toute la table) et pour les jointures (la jonction de deux tables qui partagent un même attribut).

# Notion de clé primaire

À noter qu'en SQL, on peut imposer qu'un attribut soit considéré comme clé primaire de la table. De la sorte, la base de données refusera tout simplement qu'une nouvelle ligne soit créée avec une valeur redondante de la colonne (par exemple si on essaie de créer un étudiant qui aurait le même « numéro étudiant » qu'un autre).

# Notion de clé étrangère

```
COURS    (NC:int, CODE_COURS:string, INTITULE:string)
ETUDIANT (NE:int, NOM:string, PRENOM:string)
INSCRIT  (NE:int, NC:int, ANNEE:int)
```

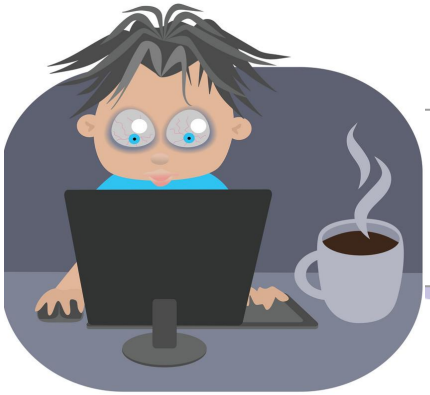
Dans notre exemple, ni NE, ni NC ne sont des clés primaires pour la table INSCRIT car un même étudiant peut être inscrit à plusieurs cours et un même cours rassemble (généralement !) plusieurs étudiants. En revanche, elles sont ce qu'on appelle des **clés étrangères**, c'est-à-dire qu'elles pointent sur la clé primaire d'une autre table, ce qui permet d'établir des jointures entre les différentes tables.

# 03

# Structured Query Language (SQL)

# SQL: Structured Language Query

- Le langage SQL, contrairement à Python, n'est pas sensible à la casse (c'est-à-dire aux majuscules et aux minuscules), mais par convention, on choisit de donner les instructions du langage en majuscules (comme `SELECT`, `FROM`, `WHERE`, etc.) alors que ce qui est du ressort des tables (autant les attributs que les noms des tables) en minuscules.
- SQL est un langage déclaratif, c'est-à-dire que l'on dit au SGBD ce que l'on veut faire, mais on ne lui dit pas (vraiment) comment il doit le faire.



# Travaux Pratiques

https://www.programiz.com/sql/online-compiler

Compilateur en ligne de SQL. Il contient 3 tables: Customers, Orders et Shippings. Nous utiliserons ces tables dans le reste du cours.

Online SQL Editor



**Customers [-]**
- customer_id [int]
- first_name [varchar(100)]
- last_name [varchar(100)]
- age [int]
- country [varchar(100)]

**Orders [-]**
- order_id [integer]
- item [varchar(100)]
- amount [integer]
- customer_id [integer]

**Shippings [-]**
- shipping_id [integer]
- status [integer]
- customer [integer]

Input

Run SQL

```
-- Online SQL Editor to Run SQL Online.
-- Use the editor to create new tables, insert data and all other SQL
operations.

SELECT * FROM Customers;
```

Output

| customer_id | first_name | last_name | age | country |
|---|---|---|---|---|
| 1 | John | Doe | 31 | USA |
| 2 | Robert | Luna | 22 | USA |
| 3 | David | Robinson | 22 | UK |
| 4 | John | Reinhardt | 25 | UK |

Available Tables

Customers

| customer_id | first_name | last_name | age |
|---|---|---|---|
| 1 | John | Doe | 31 |
| 2 | Robert | Luna | 22 |
| 3 | David | Robinson | 22 |
| 4 | John | Reinhardt | 25 |
| 5 | Betty | Doe | 28 |

Orders

| order_id | item | amount | customer_id |
|---|---|---|---|
| 1 | Keyboard | 400 | 4 |
| 2 | Mouse | 300 | 4 |
| 3 | Monitor | 12000 | 3 |
| 4 | Keyboard | 400 | 1 |
| 5 | Mousepad | 250 | 2 |

# 04

# Les opérations de bases

# Les bases

La forme générale d'une requête en SQL est :

```
SELECT ... FROM table;
```

Toutes les requêtes commencent par SELECT, se terminent par un point-virgule. Les mots-clés SELECT ... FROM réalisent l'interrogation de la table. On peut demander :

- Un attribut de la table désignée en le nommant;
- Plusieurs en les séparant par une virgule;
- Tout en les désignant par une étoile.

# Projection

La projection dans le concept de bases de données consiste à extraire des colonnes d'une table.

Exemple: `SELECT first_name, last_name, age FROM Customers;`

On qualifie cette dernière requête de projection, on dit qu'on projette la table Customers sur les colonnes first_name, last_name et age.

# Sélection

La sélection dans le concept de bases de données consiste à extraire des lignes d'une table. On utilise le mot WHERE suivi du critère (ou des critères) de sélection.

Exemple de requête:

```
SELECT first_name, last_name, age FROM Customers WHERE country = "UK";
```

# Sélection: les opérateurs de comparaison

Les opérateurs de comparaison utilisables sont :

- **=** ou **!=** qui peuvent être utilisés avec tout type de données ;
- **>**, **<**, **>=**, **<=** qui sont utilisables uniquement avec des données numériques ;
- On peut aussi utiliser LIKE (Comme), BETWEEN (Entre), IN, AND, OR, NOT. Exemple de requête:

```
SELECT * FROM Orders WHERE item = 'Keyboard' OR amount = 400;
```

# Sélection: les opérateurs de comparaison

Comparer la condition à quelque chose qui est une chaîne de caractères: LIKE.

Dans la comparaison, on utilise les jokers ‘%’ (pour remplacer un nombre quelconque de caractères) et ‘_’ (pour remplacer un seul caractère).

Exemple de requête:

```
SELECT * FROM Customers WHERE first_name LIKE '_ohn';
```

# Sélection: les tris

Trier les résultats par ordre croissant / décroissant: ORDER BY ... ASC / DESC

Exemple de requête:

```
SELECT DISTINCT customer_id FROM Orders WHERE amount > 400 ORDER BY customer_id DESC;
```

# Sélection

Projection sans doublon: SELECT DISTINCT

Exemple de requête:

```
SELECT DISTINCT customer_id FROM Orders WHERE amount > 400 ORDER
BY customer_id ASC;
```

# Sélection: autres opérateurs

- **LIMIT**: limiter à un nombre d'enregistrements que l'on va donner.
- **OFFSET**: débuter à partir d'un nombre d'enregistrements que l'on va donner.
- **UNION**, I**NTERSECT**, **EXCEPT**: opérations ensemblistes sur les tables.

# 05

# Les jointures

# Jointures

On peut, dans une base de données, croiser des informations présentes dans plusieurs tables par l'intermédiaire d'une jointure (c'est d'ailleurs le grand intérêt d'une base de données). On peut joindre deux tables quand elles ont un attribut commun.

# Jointures: exemples

Il faut bien penser à lever toute ambiguïté sur les noms d'attribut dans le cas où deux tables possèdent des colonnes de même nom, on utilise alors un préfixe.

```
SELECT DISTINCT Customers.first_name, Customers.last_name FROM
Customers JOIN Orders ON Orders.customer_id = Customers.customer_id;
```

# Jointures: exemples

On peut aussi alléger la syntaxe à l'aide d'alias avec AS:

```
SELECT Cu.first_name, Cu.last_name FROM Customers AS Cu JOIN Orders
AS Ord ON Or.customer_id = Cu.customer_id WHERE Ord.item =‘Keyboard’;
```

# 06

# Fonctions d’agrégations

# Fonctions d’agrégations

Ces fonctions permettent d’effectuer des opérations mathématiques ou des calculs statistiques sur un ensemble d’enregistrements sélectionnés.

# Fonctions d’agrégations: COUNT

**Compter les enregistrements:** On utilise la fonction COUNT. Le résultat de l’opération sera affiché dans un nouveau champ.

Exemple: Compter le nombre de clients qui ont un âge supérieur à 22 ans.

```
SELECT COUNT (Customers.customer_id) AS Nombre_clients FROM
Customers WHERE Customers.age > 22;
```

# Fonctions d’agrégations: SUM

**Additionner les valeurs d’un champ numérique:** On utilise la fonction SUM comme la fonction COUNT.

Exemple: Consulter le total des commandes.

```
SELECT SUM (Orders.amount) AS Total_commandes FROM Orders;
```

# Fonctions d’agrégations: AVG

**Calculer la moyenne des valeurs d’un champ numérique:** On utilise la fonction AVG (average = moyenne).

Exemple: Consulter la moyenne d’âge des clients.

```
SELECT AVG (Customers.age) AS Moyenne_age FROM Customers;
```

# Fonctions d’agrégations: MIN

**Afficher la valeur minimale d’un champ numérique:** On utilise la fonction MIN.

Exemple: Consulter l’âge minimum des clients.

```
SELECT MIN (Customers.age) AS Minimun_age FROM Customers;
```

# Fonctions d’agrégations: MAX

**Afficher la valeur maximale d’un champ numérique:** On utilise la fonction MAX.

Exemple: Consulter l’âge maximum des clients.

```
SELECT MAX (Customers.age) AS Maximum_age FROM Customers;
```

# 07

# Clauses de regroupement

# Clauses de regroupement

Les clauses de regroupement permettent de réaliser des opérations sur des groupes d’enregistrements.

# Clauses de regroupement: GROUP BY

La clause GROUP BY permet de créer des groupes d'enregistrements sur lesquels pourront être utilisées les fonctions d'agrégation. Elle est nécessaire dès lors que l'on souhaite afficher des données issues des tables et des données issues de fonctions d'agrégation.

Les champs de l'instruction SELECT doivent être repris dans la clause GROUP BY.

# Clauses de regroupement: GROUP BY

La syntaxe est : GROUP BY Champ1, Champ2, ...

Exemple : afficher le client avec l'âge maximal dans chaque pays.

```
SELECT country, MAX (age) FROM Customers GROUP BY country;
```