SQL
Structured Query Language

Danielle KEPSEU

Plan

Introduction

- **Base de données (BDD)**
- **Système de Gestion de Base de Données (SGBD)**

SQL

- **Définition**
- **Commandes de base**
- **Premières requêtes (Create, Drop, Insert, Select, Update, etc.)**
- **Jointures**

Introduction

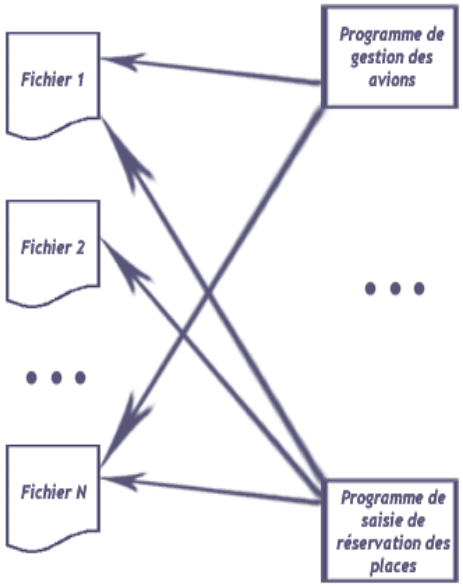
Définition d'une base de données

- Une **base de données** (BDD ou DB), **database en anglais** (*DB*) est une collection informatisée de données **stockées, électroniquement** en vue d'être exploitées et partagées par différents utilisateurs.
- Le terme base de données voit le jour dans les années 1960.
- Les données peuvent être stockées sous une forme très **structurée** (base de données relationnelles par exemple), ou bien sous la forme de données brutes peu structurées (avec les bases de données **NoSQL** par exemple).
- Une base de données peut être localisée dans un même lieu et sur un même support informatisé, ou répartie sur plusieurs machines à plusieurs endroits.
- La notion de base de données est généralement couplée à la notion de réseau.

Des fichiers aux Bases de Données

FICHIERS + PROGRAMMES D'APPLICATION

- Adoptés jusqu'à la fin des années 1970
- Les données des fichiers sont décrites dans les programmes



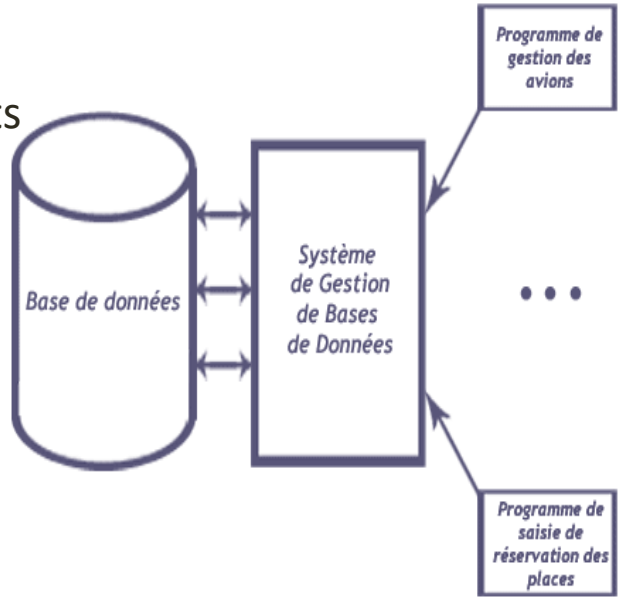
- Contrôle de la validité des données très complexe
- Langage de programmation de bas niveau lourd à manier. Il faut connaître les opérations de base (ouvrir, fermer, lire tant que, etc.)
- Un changement dans la structure d'un enregistrement nécessite la réécriture de tous les programmes qui l'utilisent
- Applications difficilement portables

Des fichiers aux Bases de Données

Solution : BASES DE DONNÉES + SYSTÈME de GESTION DE BASE DE DONNÉES

Adoptée à partir des années 1980

- Les données et les programmes sont indépendants
- Description unique des données
- Gestion centralisée des données



Utilité d'une base de données ?

- **Stocker et organiser** de grands volumes de données
- **Faciliter l'exploitation**
  - Ajout/Suppression,
  - Mise à jour,
  - Recherche de données
- Possibilité de pouvoir **accéder à la BDD par plusieurs utilisateurs simultanément**

Exemples :

La plupart des entreprises possèdent des bases de données informatiques contenant des informations essentielles à leur fonctionnement :

- carnet d'adresses, fichier clients
- gestion de projet : découpage en tâches, plan financé
- catalogue des produits de l'entreprise avec photos
- carnet de commandes

Utilité d'une base de données ?

Covid : le Royaume-Uni passe à côté de milliers de cas à cause… d'un fichier Excel arrivé à saturation

Les autorités sanitaires britanniques ont reconnu que près de 16.000 cas de coronavirus en Angleterre sont passés sous le radar au cours de la semaine écoulée à cause d'un problème dans le chargement des données.

[Covid : le Royaume-Uni passe à côté de milliers de cas à cause… d'un fichier Excel arrivé à saturation | Les Echos]

Fonctions d'un SGBD ?

Un **Système de Gestion de Base de Données**, abréviation **SGBD**, en anglais **DBMS** pour Data Base Management System), est un logiciel qui permet d'interagir avec la base de données.

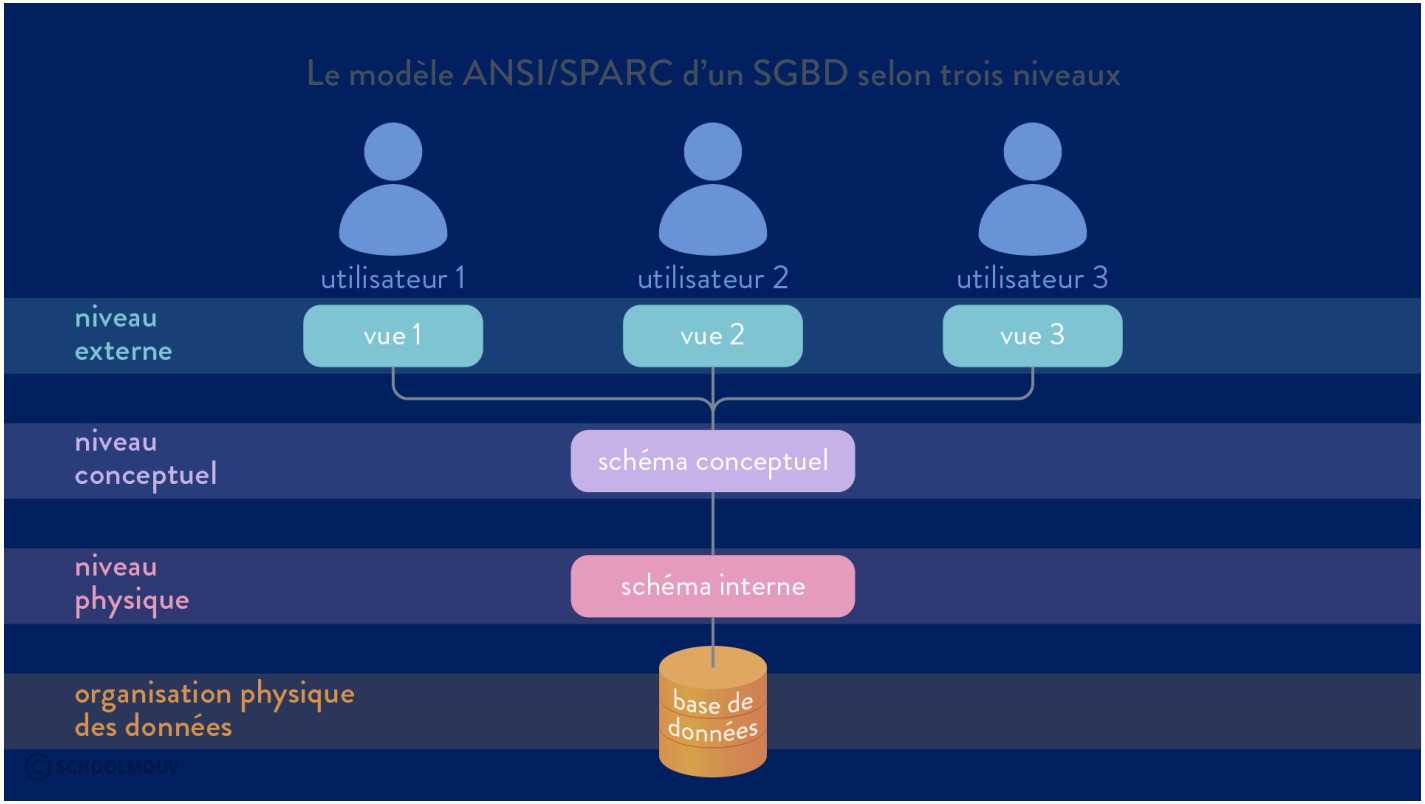
Les rôles d'un SGBD sont :

- **Description et structuration des données**
  - Décrire les données et leurs types
  - Définir des règles pour garantir l'intégrité des données (comme les contraintes de domaines et d'existence, etc.)
- **Manipulation des données**
  - Interroger et mettre à jour les données sans préciser d'algorithme d'accès (sélectionner, modifier, rechercher, regrouper, trier, etc.)
- **Persistance des données**
  - Assurer que les données sont stockées de manière permanente, même après la fin des programmes qui les utilisent.

Fonctions d’un SGBD ?

- **Fiabilité des données :** Assurer que les données sont correctes et fiables.
  - S’assurer que les données respectent certaines règles.
  - Utiliser des mots de passe et des autorisations pour protéger les données.
  - Assurer que les opérations sur les données sont sécurisées.
  - Prévoir des copies de secours en cas de problème.
  - Gérer les accès de plusieurs utilisateurs en même temps.
- **Accès fiable :**
  - Assurer un accès constant et sécurisé aux données.
  - Utiliser des techniques comme l’indexation pour rendre le système plus rapide.
- **Indépendance des données :**
  - Pouvoir changer l’organisation des données sans affecter les applications.
  - Montrer aux utilisateurs seulement les données qui les concernent.

Architecture d'un SGBD



En 1965, Charles Bachman conçoit l'architecture ANSI/SPARC encore utilisée de nos jours.

Types de SGBD

Hiérarchique

- les premiers, avec gestion de pointeurs arborescente

Réseau (graphe)

- les plus rapides, la navigation est entre pointeurs

Relationnel (le plus utilisé)

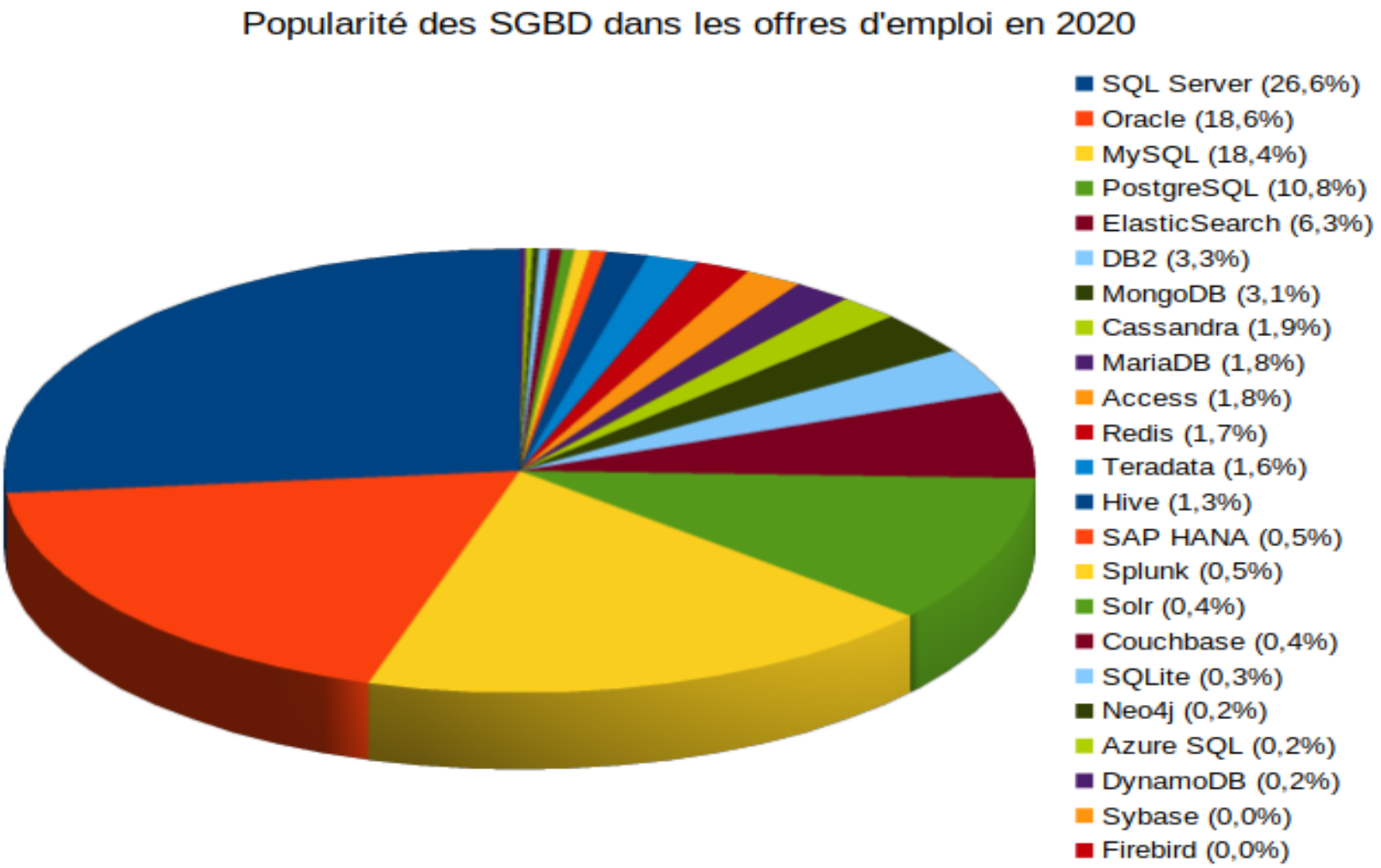
- les plus utilisés, basés sur l'algèbre relationnelle
- représentation des données sous forme de tables

Déductif (faits + règles)

- utilisés pour les systèmes experts (calcul de prédicats)

Objet

- instances de classes hiérarchisées



https://emploi.developpez.com/actu/315884/Emploi-informatique-2020-les-bases-de-donnees-les-plus-demandees-et-les-mieux-payees-edition-etendue-avec-23-SGBD/

SQL

STRUCTURED **Q**UERY **L**ANGUAGE

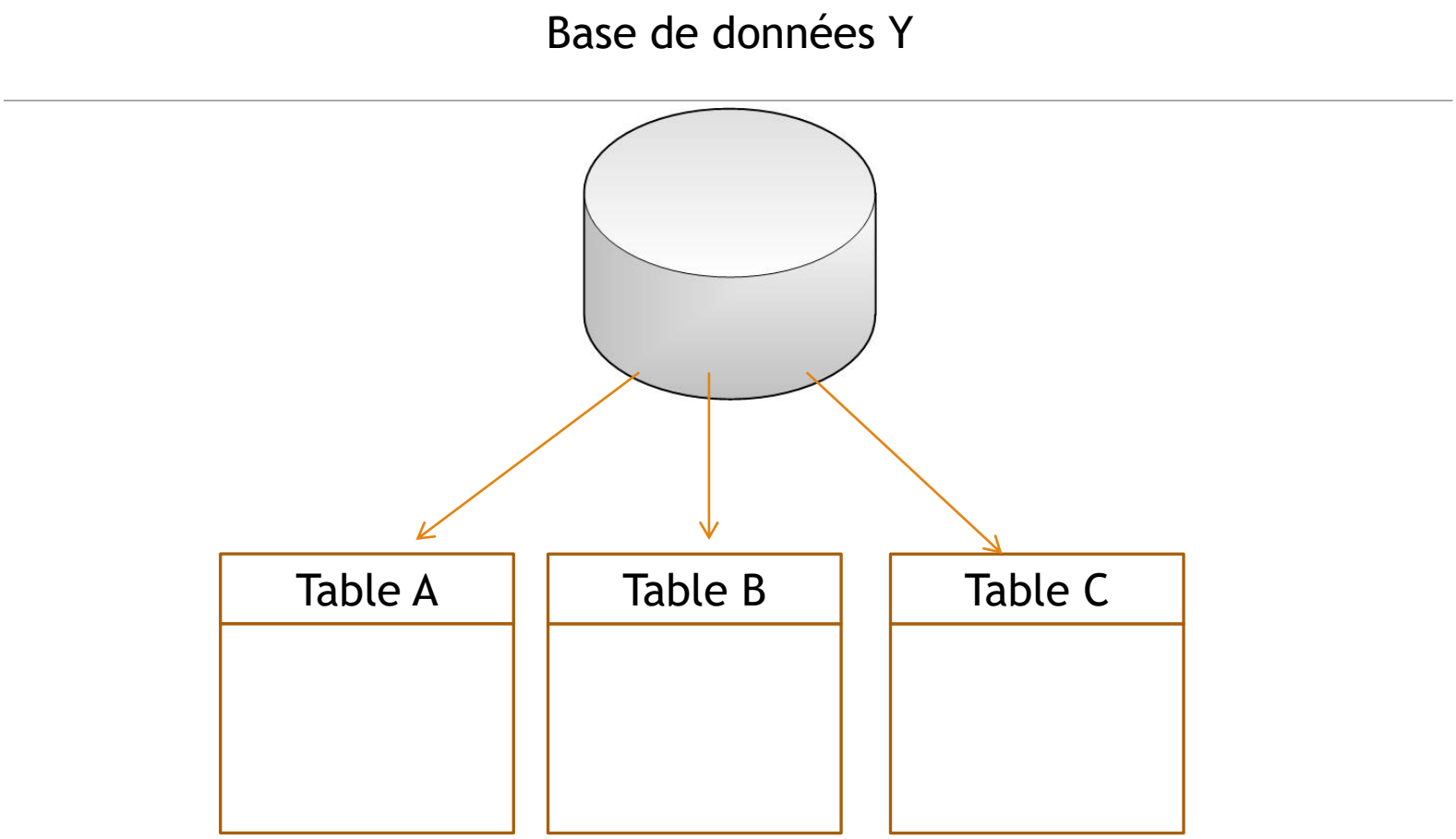
Kesako ?

- SQL (**S**tructured **Q**uery **L**anguage) ou Langage structuré de requêtes
- C'est un langage de requête, **pas un langage de programmation**
- Nécessite
  - un moteur de base de données :
    - *MySql, Maria DB, Sql Serveur, Oracle,…*
  - Un langage applicatif pour s'y connecter :
    - Php, C#, Vb, Java, Python, Ruby, …

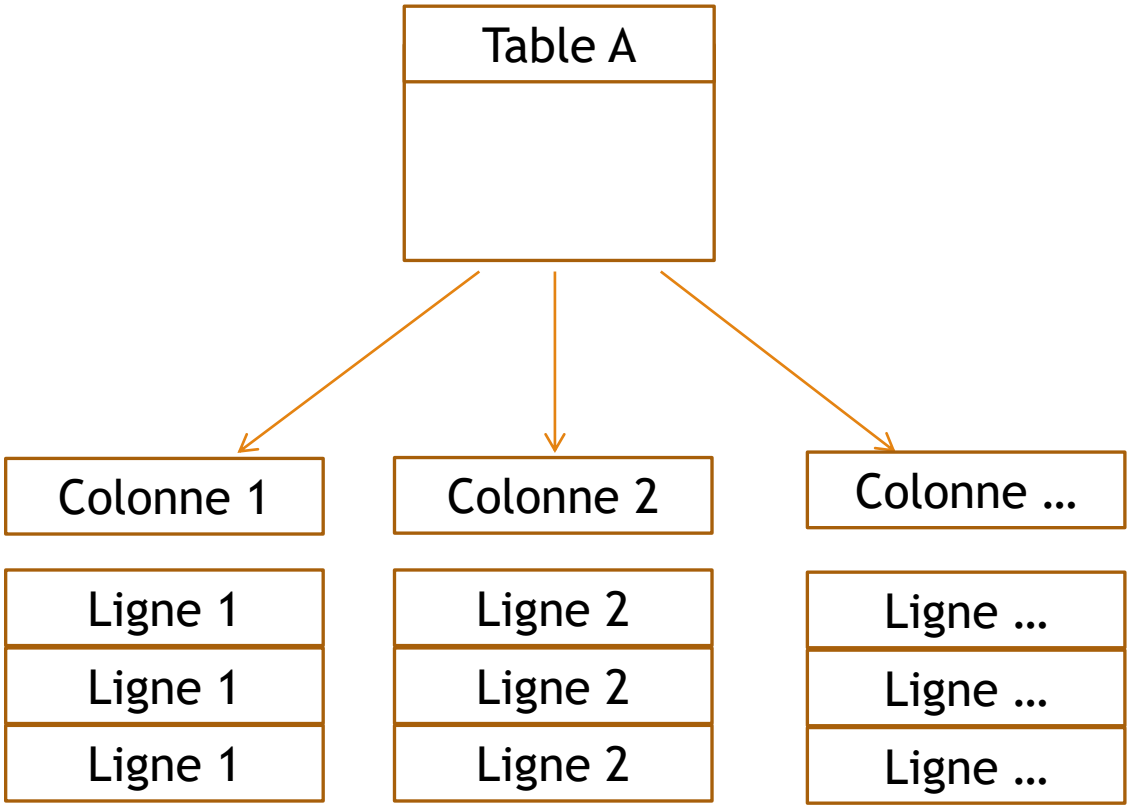
Pour quelle application ?

- SQL est un langage basé sur l'algèbre relationnelle
  - Gestion des bases de données, et des données : des tables, des enregistrements
  - Contrôle les accès aux informations d'une base de données
- Il utilise les termes **table, ligne (ou enregistrement**) et **colonne (ou champ**) au lieu des termes relationnels relation, tuple et attribut
- Un script peut être enregistré dans un fichier avec l'extension **.sql**
- Une instruction SQL peut s'écrire sur plusieurs lignes pour être exécutée, l'instruction doit se terminer par un point-virgule (;)

Structure de la base de données



Structure de la base de données

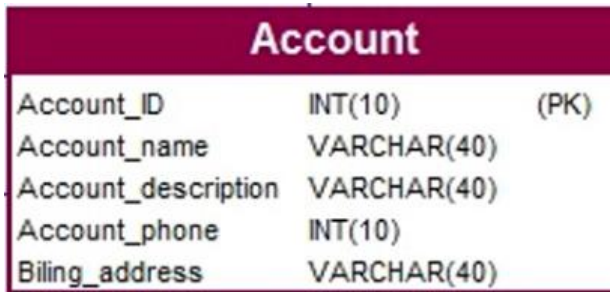


Structure de la base de données

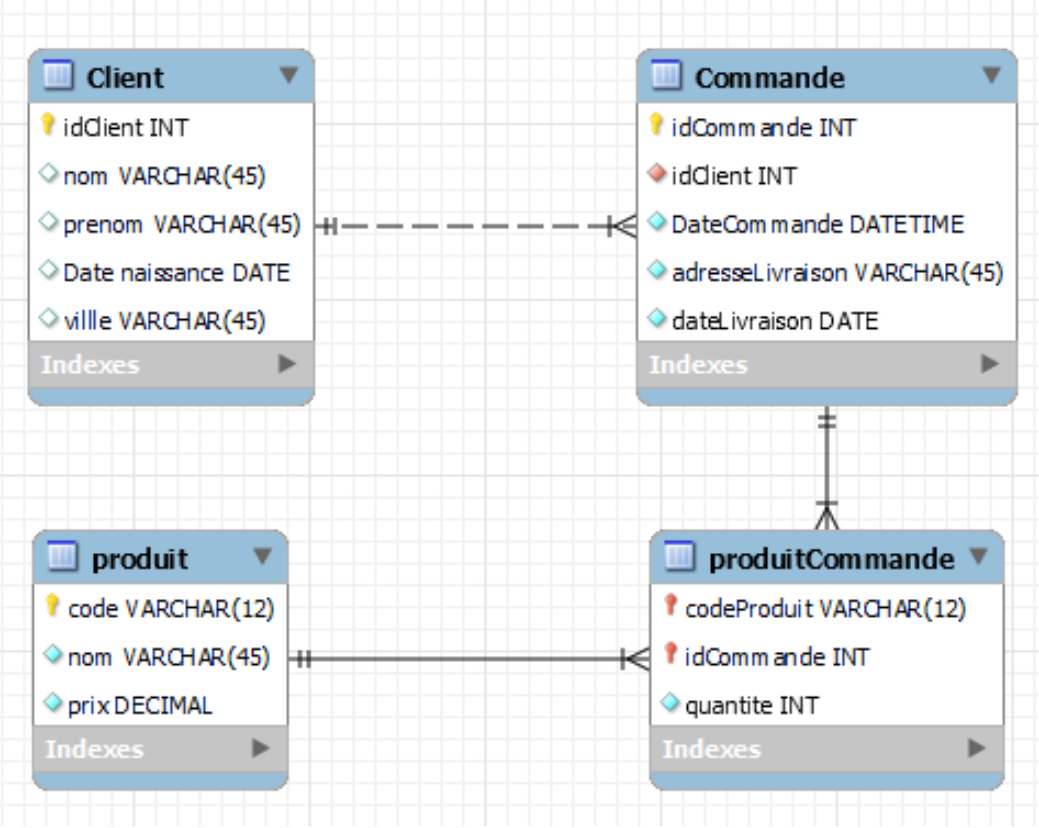
Colonne 1 VARCHAR (100)	Colonne 2 INT	Colonne … BIT (bool)
Je suis une chaine	10	True

Une table a
- 1 nom
- X colonnes nommées et typées
- 1 colonne est marquée comme **clef primaire**

Les DATAS ont des contraintes
- Typées : char, int, varchar, float, bit, ….
- Longueur : à définir



Exemple
Schéma d'une base de données



CREATE

Créer une base de données

CREATE DATABASE nom_base

ou

CREATE DATABASE **IF NOT EXISTS** nom_base

CREATE

Créer votre 1ère TABLE avec SQL

```
CREATE TABLE NOM_DE_MA-TABLE
(
      colonne1 type_donnees,
      colonne2 type_donnees,
      colonne3 type_donnees,
      colonne4 type_donnees
)
```

```
CREATE TABLE IF NOT EXISTS CLIENT
(
      ID INT PRIMARY KEY NOT NULL,
      NOM VARCHAR(100),
      PRENOM VARCHAR(100),
      VILLE VARCHAR(255),
      DATENAISS DATE
)
```

Lorsqu'on créé une table on doit donner un nom à chaque colonne et préciser le typage des données des colonnes. On peut également rajouter les contraintes.

Clé primaire

PRIMARY KEY *(nomColonne1)*

Une clé primaire **permet d'identifier de façon unique chaque enregistrement d'une table**.

Exemples de déclaration d'une clé primaire :

```
CREATE TABLE IF NOT EXISTS CLIENT
(
    ID INT  NOT NULL,
    NOM VARCHAR(100),
    PRENOM VARCHAR(100),
    VILLE VARCHAR(255),
    DATENAISS DATE,
    PRIMARY KEY(ID)
)
```

```
CREATE TABLE IF NOT EXISTS CLIENT
(
    ID INT PRIMARY KEY NOT NULL,
    NOM VARCHAR(100),
    PRENOM VARCHAR(100),
    VILLE VARCHAR(255),
    DATENAISS DATE
)
```

Cas d'une clé primaire composée

PRIMARY KEY *(nomColonne1, nomColonne2)*

Une concaténation des valeurs de deux colonnes

Exemple de la déclaration d'une clé primaire composée

```
CREATE TABLE IF NOT EXISTS contenu(
codeProduit VARCHAR(10),
idCommande INTEGER,
quantite INTEGER,
PRIMARY KEY(codeProduit, idCommande)
);
```

Déclarer une clé étrangère

FOREIGN KEY (nomColonne) **REFERENCES** (nomColonneDansLaTableOrigine)

Une *clé étrangère est* une colonne (ou plusieurs colonnes) d'une table qui *est* reliée à la colonne clé primaire d'une autre table.

Exemple de la déclaration d'une clé étrangère

```
CREATE TABLE IF NOT EXISTS contenu (
codeProduit VARCHAR(10),
idCommande INTEGER,
quantite INTEGER,
PRIMARY KEY(codeProduit, idCommande),
FOREIGN KEY(codeProduit) REFERENCES produit (code),
FOREIGN KEY(idCommande) REFERENCES commande (id)
);
```

ALTER

Modifier une TABLE avec SQL

Les principales modifications sur la structure d'une table sont : **l'ajout d'un attribut , la modification d'un attribut ou la suppression d'un attribut**

```
ALTER TABLE  NOM_DE_MA-TABLE
ADD nom_colonne type_donnees,
```

```
ALTER TABLE Client
ADD Email varchar(255);
```

```
ALTER TABLE  NOM_DE_MA-TABLE
DROP nom_colonne;
```

ET AUSSI

```
MODIFY nom_colonne type_donnees,
CHANGE colonne_ancien_nom colonne_nouveau_nom type_donnees
```

DROP

Supprimer votre une table ou une base de données

DROP TABLE NOM_DE_MA-TABLE

```
DROP TABLE client;
```

DROP DATABASE NOM_DE_MA-TABLE

```
DROP DATABASE testDB;
```

Actions sur la base de données

- Insérer
- Modifier
- Lire et Manipuler

Actions sur la base de données

Insérer des données

```
INSERT INTO TABLE
(nom_colonne_1,
nom_colonne_2, ...)
VALUES
('valeur 1', 'valeur 2', ...)
```

```
INSERT INTO CLIENT
(PRENOM, NOM, VILLE,
DATENAISS)
VALUES
('Pierre', 'Laporte', 'Agen', '1990-01-01'
),
('Sophie', 'Durand', 'Boé', '1980-10-03'),
('Lucy', 'Yong', 'Layrac', '2000-03-11')
;
```

Actions sur la base de données

Mettre à jour des données

UPDATE **TABLE**
SET
colonne_1 = 'valeur 1',
colonne_2 = 'valeur 2',
colonne_3 = 'valeur 3'

WHERE condition

```
UPDATE CLIENT
SET
email = 'toto@aol.com'
WHERE VILLE = 'Agen',
```

Actions sur la base de données

Supprimer des données

```
DELETE FROM TABLE
WHERE condition;
```

```
DELETE FROM Client
WHERE nom='Doe';
```

Actions sur la base de données

Lire des données

Se compose de 3 parties

SELECT
nom_du_champ1,
nom_du_champ2

Liste des champs
à afficher

FROM

Tables cibles

Liste des tables
où allez chercher
les données

WHERE

Conditionnelle

Condition
sur les données
des tables cibles

Actions sur la base de données

Lire des données

Se compose de 3 parties

SELECT	FROM	WHERE
NOM, PRENOM	CLIENT	ID> 2 AND NOM LIKE ‘J%’

SELECT	FROM	WHERE
NOM, PRENOM	CLIENT	VILLE = ‘Agen’

Actions sur la base de données

Lire des données

SELECT * FROM TABLE

Affiche l'ensemble des données d'une table

* signifie all

SELECT NOM, PRENOM FROM TABLE

Affiche uniquement les noms et prénoms d'une table

Actions sur la base de données

La clause WHERE

Nom de champ	Opérateur	Valeur de champs
AGE	=	Égale
	<>	Pas égale
	>	Supérieur à
	<	Inférieur à
	>=	Supérieur ou égale à
	<=	Inférieur ou égale à
	IN	Liste de plusieurs valeurs possibles
	BETWEEN … AND …	Valeur comprise dans un intervalle
NOM	LIKE	Recherche en spécifiant le début, milieu ou fin d'un mot.
	IS NULL	Valeur est nulle
	IS NOT NULL	Valeur n'est pas nulle

Actions sur la base de données

La clause WHERE

Se couple grâce à des AND ou/et OR

Nom de champ = Valeur de champs

AND

Nom2 de champ = Valeur de champs

Valeur n'est pas nulle

Exemple

Lire des données

```
SELECT NOM, PRENOM
FROM TABLE
WHERE
AGE > 50
```

Affiche uniquement les noms et prénoms des personnes de la table dont l'AGE est supérieur à 50 ans.

```
SELECT NOM, PRENOM
FROM TABLE
WHERE
((AGE <= 50 AND AGE > 12)
OR
(NOM LIKE 'J%' AND VILLE
LIKE '%c'))
```

Affiche uniquement les noms et prénoms des personnes dont l'AGE est supérieur à 12 ans et inférieur ou égal à 50 ans OU dont les noms commencent par 'J' et les villes finissent par 'c'.

Exemple

Pour lire des données

```
SELECT NOM, PRENOM FROM TABLE
WHERE
(AGE <= 50 AND AGE > 12)
OR
(NOM LIKE 'J%' AND VILLE LIKE '%c'))
```

Affiche uniquement les noms et prénoms
pour les AGES > 12 ans et <=à 50 ans OU les noms commencent par 'J' et les villes finissent par 'c'
d'une table

Actions sur la base de données

La clause ORDER BY

```
SELECT colonne1, colonne2, colonne3
FROM table
ORDER BY colonne1 DESC, colonne2 ASC
```

Fonctions d'agrégation

Opérations statistiques

- **COUNT(column_name | *)** : renvoie le nombre de lignes d'une table.
- **SUM(column_name)** renvoie la somme des valeurs d'une colonne numérique.
- **AVG(column_name)** renvoie la valeur moyenne d'une colonne numérique.
- **MIN(column_name)** renvoie la valeur minimale d'une colonne sélectionnée.
- **MAX(column_name)** renvoie la valeur maximale d'une colonne sélectionnée.

Utilisées conjointement avec GROUP BY et HAVING permettent de faire des regroupements de lignes pour faire une somme par exemple

Fonctions d'agrégation

Opérations statistiques

```
SELECT Author, COUNT(*) AS NumberOfBooks
FROM Books
GROUP BY Author;
```

Author	NumberOfBooks
Anthony Molinaro	1
Alan Beaulieu	1
Donald Knuth	2

```
SELECT SUM(DISTINCT Price)
AS TotalDistinctPrice
FROM Books;
```

TotalDistinctPrice
72

Utilisée conjointement avec GROUP BY et HAVING permet de faire des regroupements de lignes pour faire une somme par exemple

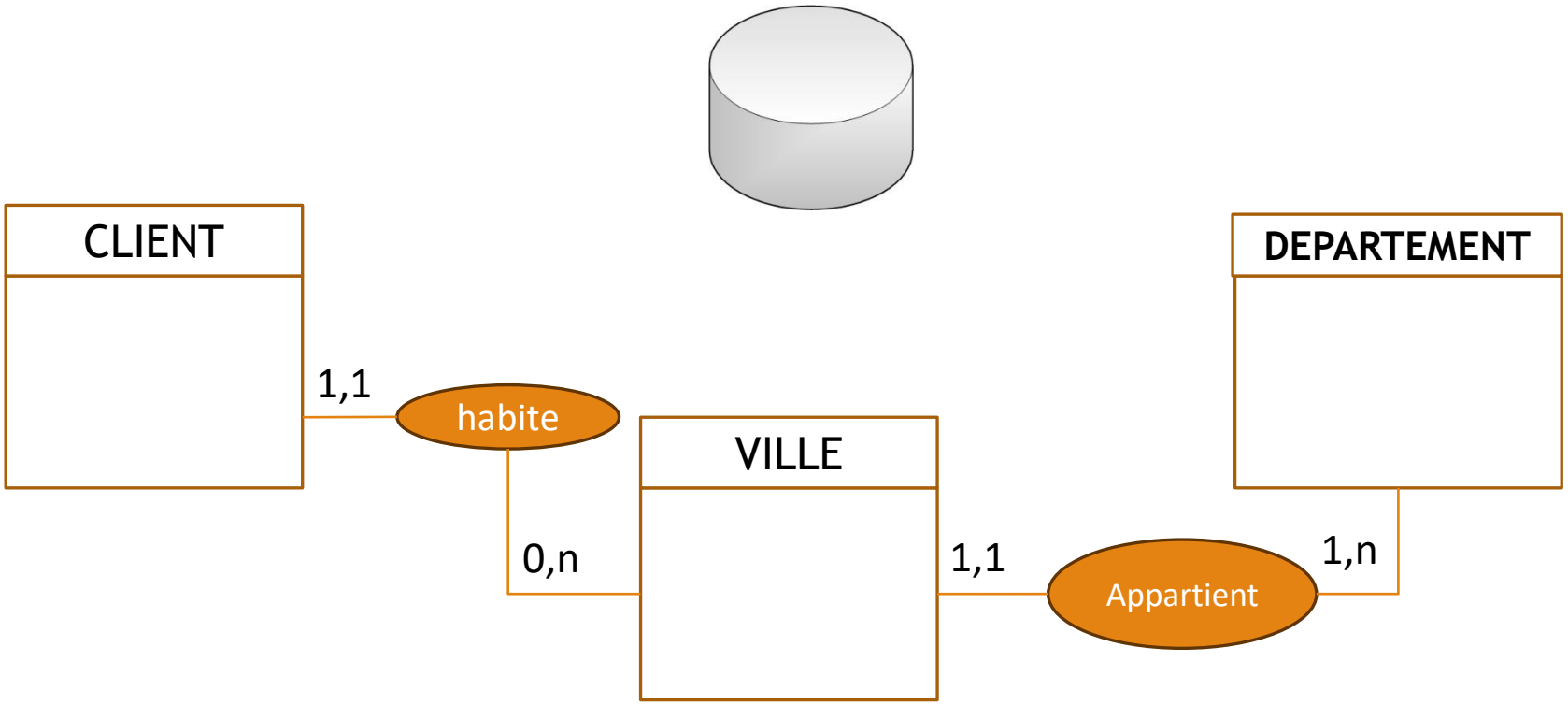
Structure de la base de données

Pourquoi relationnelle?

- Les bases de données relationnelles sont basées sur le modèle relationnel, un moyen intuitif et simple de représenter des données dans des tables.
- Les colonnes de la table contiennent les attributs des données, et chaque enregistrement a généralement une valeur pour chaque attribut, ce qui facilite l'établissement des relations entre les points de données.

Structure de la base de données

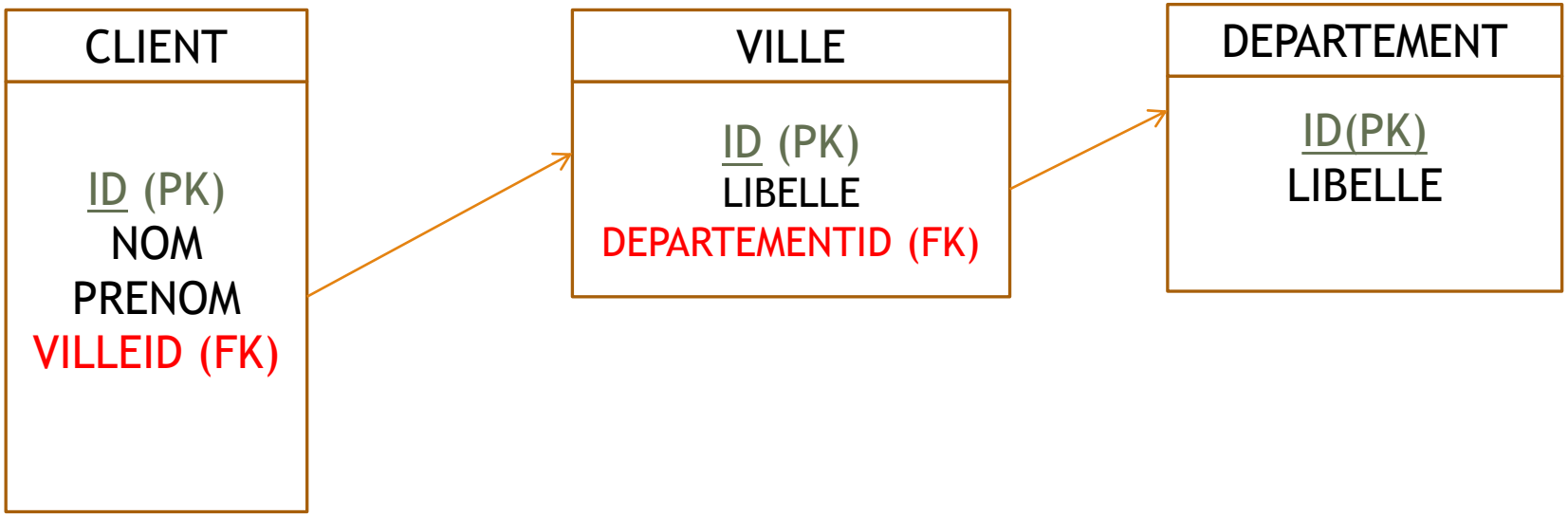
Base de données Y



Modèle conceptuel de données (MCD)

Exemple

Liaison 1 à n



Modèle logique de données (MLD)

Les jointures

Comment relier 2 tables

SELECT colonneA3, colonneA2, colonneB4
FROM tableA, tableB
WHERE **tableA** .colonneA1 = **tableB**.colonneB1
ORDER BY colonne1 DESC, colonne2 ASC

colonneA1 = clef primaire table 1

colonneB1 = clef secondaire ou étrangère table 2

```
SELECT LIBELLE, PRENOM,NOM
FROM Client,  Ville
WHERE Ville.ID = Client.VILLEID;
```

Les jointures

Comment relier 2 tables

```
SELECT *
FROM table1
INNER JOIN table2
ON table1.colonneA1 =
table2.colonneB1
```

```
SELECT LIBELLE, NOM, PRENOM
FROM Client
INNER JOIN Ville
ON Ville.ID = Client.VILLEID;
```

colonneA1 = clef primaire

colonneB1 = clef secondaire ou étrangère

Les jointures

Autres types de jointures

LEFT JOIN : liste tous les résultats de la table de gauche
même s'il n'y a pas de résultats dans la seconde table

RIGHT JOIN : liste tous les résultats de la table de droite
même s'il n'y a pas de résultats dans la seconde table

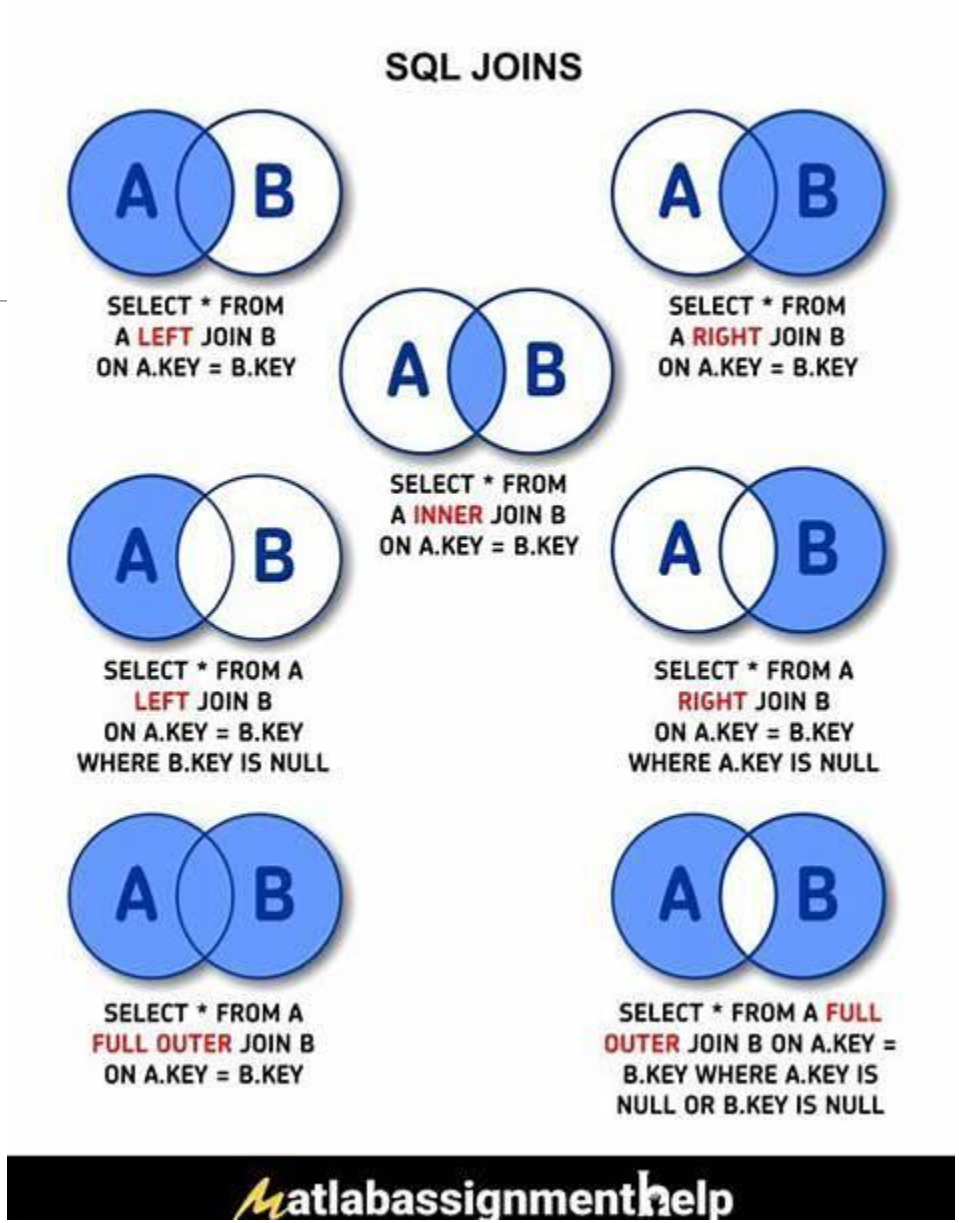
FULL JOIN : combine les résultats des 2 tables

SELF JOIN : crée une relation d'une table sur elle même

CROSS JOIN : retourne chaque ligne d'une table avec chaque ligne d'une autre table

Les jointures

Exemples



Liens utiles

Les commandes SQL les plus importantes | LearnSQL.fr

Cours et Tutoriels sur le Langage SQL

SQL Tutorial (w3schools.com)