

Introduction aux concepts des bases de données

Qu'est-ce qu'une donnée ?

En informatique, les **données** (**data** en anglais) sont des **représentations d'informations**

Nous pouvons les **stocker**, les **traiter** et les **manipuler**

Elles peuvent prendre des formes ou des **types** variés

Les données (data)

Je m'appelle Jean Dupont, je suis garagiste et j'ai 25ans

Je m'appelle Marie Dupuis, je suis chauffagiste et j'ai 33ans

Je m'appelle Eric François, je suis contrôleur et j'ai 51ans

Les données (data)

Je m'appelle Jean Dupont, je suis garagiste et j'ai 25ans

Je m'appelle Marie Dupuis, je suis chauffagiste et j'ai 33ans

Je m'appelle Eric François, je suis contrôleur et j'ai 51ans

Données structurées, semi-structurées, non structurées

Les données structurées : organisées et facilement exploitables

Elles peuvent être représentées dans des **tableaux**

Leur **format** est prédéfini

Elles sont **faciles à manipuler**

Les données (data)

Je m'appelle Jean Dupont, je suis garagiste et j'ai 25ans

Je m'appelle Marie Dupuis, je suis chauffagiste et j'ai 33ans

Je m'appelle Eric François, je suis contrôleur et j'ai 51ans

Prénom	Nom	Métier	Age
Jean	Dupont	garagiste	25
Marie	Dupuis	chauffagiste	33
Eric	François	controleur	51

Quelques exemples de données structurées :

Le format CSV (comma-separated values)

Le fichier représente un **tableau**

Les données sont séparées par des **virgules**, des points-virgules ou des tabulations

La première ligne du fichier donne le nom des **colonnes**

```
year,position,artist,song,indicativerevenue,us,uk,de,fr,ca,au
"2000","1","Faith Hill","Breathe","24030.051","2","33","-","-","-","-","1"
"2000","2","Santana & The Product G","Maria Maria","23320.084","1","1"
"2000","3","Joe Thomas","I Wanna Know","21516.777","4","-","-","-","-","1"
"2000","4","Aaliyah","Try Again","21099.824","1","5","5","26","-","1"
"2000","5","Toni Braxton","He Wasn't Man Enough","21023.066","2","5"
"2000","6","Rob Thomas & Santana","Smooth","20735.418","1","3","21"
"2000","7","Vertical Horizon","Everything You Want","20402.965","1"
"2000","8","Destiny's Child","Say My Name","19489.657","1","3","14"
"2000","9","Lonestar","Amazed","19138.169","1","21","91","-","-","1"
"2000","10","Matchbox Twenty","Bent","18997.978","1","-","-","-","-","1"
"2000","11","Madonna","Music","18983.471","1","1","2","8","1","1"
"2000","12","Sisqo","Thong Song","18403.832","3","3","15","15","-","1"
"2000","13","Three Doors Down","Kryptonite","18341.509","3","-","85"
"2000","14","Destiny's Child","Jumpin' Jumpin'","18020.444","3","5"
"2000","15","Creed","Higher","17082.223","7","47","-","-","-","-","1"
```

source : Chart2000, version 0.3.0067

Quelques exemples de données structurées :

Le tableur

Un programme qui manipule des **feuilles de calcul**, et qui représente également les données en **tableaux**

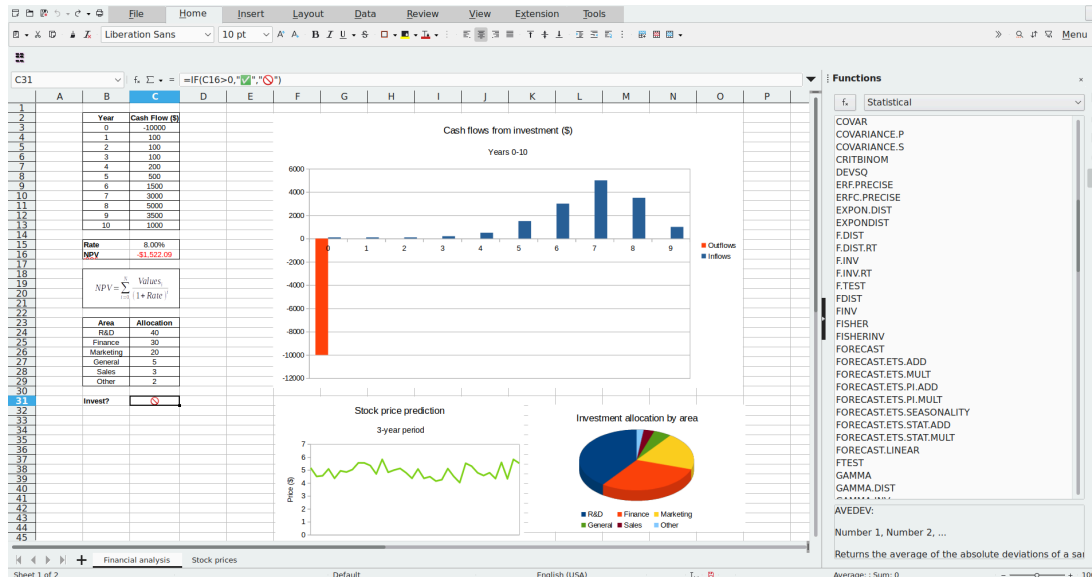
C11 <L> TOTAL C1
25

	A	B	C	D
1	ITEM	NO.	UNIT	COST
2				
3	MUCK RAKE	43	12.95	556.85
4	BUNZ CUT	15	6.75	101.25
5	TOE TONER	250	49.95	12487.50
6	EYE SNUFF	2	4.95	9.90
7				
8			SUBTOTAL	13155.50
9			9.75% TAX	1282.66
10			TOTAL	14438.16

VisiCalc, le premier
tableur (1979)

Quelques exemples de données structurées :

Le tableur




Des fonctionnalités diverses selon les programmes :
formules et fonctions,
graphiques et visualisations,
tableau croisé dynamique,
macros et scripts...


LibreOffice Calc

Les tableurs sont généralement utilisés par des **individus** ou des **petites équipes** pour des tâches **spécifiques**

Ils sont peu pratiques pour de gros volumes de données


Les **bases de données** offrent des fonctionnalités plus avancées pour le stockage et la gestion de données à **grande échelle**, et sont utilisées pour des applications **plus larges** et **plus complexes**


Aide • 



Insee

English
Presse


MENU




RECHERCHE

Le fichier des données harmonisées des recensements de la population de 1968 à 2019


Le fichier contient 18 variables et 51 531 264 observations.

FICHIER DONNÉES HARMONISÉES DES RECENSEMENTS DE LA POPULATION DE 1968 À 2019

(dbase, 308 Mo)



(csv, 329 Mo)



> [Accéder à la liste des variables du fichier \(pdf\)](#)

Spécifications et limites relatives à Excel

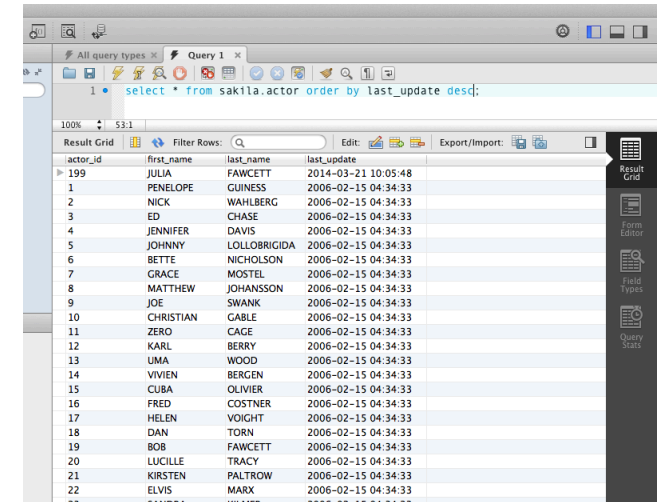
Fonctionnalité	Limite maximale
Ouverture des classeurs	Limité par la quantité de mémoire disponible et les ressources système
Nombre total de lignes et de colonnes dans une feuille de calcul	1 048 576 lignes et 16 384 colonnes
Largeur des colonnes	255 caractères
Hauteur des lignes	409 points

Quelques exemples de données structurées :

La base de données relationnelle

Un modèle développé dès le début des années 1970, mais qui demeure l'un des plus utilisés aujourd'hui

Elle représente les données sous forme de **tables**



The screenshot shows the MySQL Workbench interface with a query window titled 'Query 1'. The query is: `select * from sakila.actor order by last_update desc;`. The result grid displays the following data:

actor_id	first_name	last_name	last_update
199	JULIA	FAWCETT	2014-03-21 10:05:48
1	PENELOPE	GUINNESS	2006-02-15 04:34:33
2	NICK	WAHLBERG	2006-02-15 04:34:33
3	ED	CHASE	2006-02-15 04:34:33
4	JENNIFER	DAVIS	2006-02-15 04:34:33
5	JOHNNY	LOLLOBRIGIDA	2006-02-15 04:34:33
6	BETTE	NICHOLSON	2006-02-15 04:34:33
7	GRACE	MOSTEL	2006-02-15 04:34:33
8	MATTHEW	JOHANSSON	2006-02-15 04:34:33
9	JOE	SWANK	2006-02-15 04:34:33
10	CHRISTIAN	GABLE	2006-02-15 04:34:33
11	ZERO	CAGE	2006-02-15 04:34:33
12	KARL	BERRY	2006-02-15 04:34:33
13	UMA	WOOD	2006-02-15 04:34:33
14	VIVIEN	BERGEN	2006-02-15 04:34:33
15	CUBA	OLIVIER	2006-02-15 04:34:33
16	FRED	COSTNER	2006-02-15 04:34:33
17	HELEN	VOIGHT	2006-02-15 04:34:33
18	DAN	TORN	2006-02-15 04:34:33
19	BOB	FAWCETT	2006-02-15 04:34:33
20	LUCILLE	TRACY	2006-02-15 04:34:33
21	KIRSTEN	PALTROW	2006-02-15 04:34:33
22	ELVIS	MARX	2006-02-15 04:34:33

Un aperçu de base de données dans le logiciel MySQL Workbench

Les données non structurées : riches en informations, mais complexes à traiter

Abondantes dans le **Big Data**, elles sont faciles à accumuler

Exemples : e-mails, photos, fichiers audio...

Elles ne peuvent **pas** être correctement représentées sous la forme d'un tableau

Éric François <eric.francois@mail... 09:30 (il y a 6 minutes) ☆ ↶ ⋮

À Marie Dupuis ▼

Joyeux anniversaire Marie ! J'espère que tu fêtes ça dignement, 33 ans c'est pas rien ! Comment tu vas ?

Ça fait un bail, j'espère te recroiser bientôt. T'as des nouvelles de Jean ? Je me demandais si il était toujours garagiste.

Si ça t'intéresse, je connais une pâtisserie qui fait des fraisiers super bons, celle qui est rue Fragaria. Pas d'anniversaire sans gâteau !

À bientôt,

Éric

Éric François <eric.francois@mail... 09:30 (il y a 6 minutes) ☆ ↶ ⋮

À Marie Dupuis ▼

Joyeux anniversaire Marie ! J'espère que tu fêtes ça dignement, 33 ans c'est pas rien ! Comment tu vas ?

Ça fait un bail, j'espère te recroiser bientôt. T'as des nouvelles de Jean ? Je me demandais si il était toujours garagiste.

Si ça t'intéresse, je connais une pâtisserie qui fait des fraisiers super bons, celle qui est rue Fragaria. Pas d'anniversaire sans gâteau !

À bientôt,

Éric

Leur **traitement** est plus complexe : data science, intelligence artificielle, machine learning...

Nom	Prénom	Métier	Âge	Va peut-être s'acheter un fraisier
François	Éric	?	33	?
Dupuis	Marie	?	?	oui ?
?	Jean	Garagiste ?	?	?

Les bases de données

Qu'est-ce qu'une base de données ?

Une base de données (BDD), ou database (DB), est **un ensemble organisé de données stockées** électroniquement dans un système informatique

L'histoire des bases de données remonte aux **années 1960** : techniquement, on commence à pouvoir stocker **de grandes quantités de données**

Développement des premiers systèmes informatiques, données des entreprises, informations militaires...

Le besoin de gérer efficacement ces **grands volumes** de données a conduit à la création des BDD

Elles sont accessibles simultanément par différents programmes ou différents utilisateurs

Elles peuvent être stockées dans un ou plusieurs fichiers, locales ou distantes, sur une ou plusieurs machines

Les avantages du **cloud** : maintenance, réduction des coûts, scalabilité

Différents types de BDD

Il existe **différents modèles** de base de données, adaptées à **différents types de données**, et répondant à des exigences différentes : orientée texte, hiérarchique, réseau, relationnelle, orientée objet...

Les toutes premières BDD étaient des bases de données hiérarchiques

De nos jours, les deux modèles dominants sont le modèle **relationnel** et le modèle **non-relationnel**

Interagir avec la base : le SGBD

Le **système de gestion de base de données** (SGBD), ou database management system (DBMS), est un **logiciel** qui permet de manipuler la structure et le contenu des bases de données

Le SGBD sert d'**intermédiaire** entre l'utilisateur/l'application et les données

Il a été inventé dans les années 1960 pour gérer les bases de données hiérarchiques

Il permet de réaliser les quatre opérations de base du **CRUD** : créer, lire, mettre à jour et supprimer

C



create

R



read

U



update

D



delete

ainsi que des actions **plus avancées** : les fonctions d'agrégation, les jointures, les requêtes complexes, etc.

Il gère la **sécurité** et les droits d'accès

Il assure la **sauvegarde** et la récupération des données

Il utilise des techniques d'**optimisation des performances** :
indexation, mise en cache, parallélisation des requêtes...

Il assure la réussite des transactions, et permet d'éviter des
problèmes dus aux **accès concurrents**

Les bases de données relationnelles : les plus largement utilisées

La BDD relationnelle est un modèle simple et **puissant**, qui demeure la façon la plus **populaire** de gérer une BDD aujourd'hui

Développée dès les années 1970 par les travaux d'Edgar F. Codd, elle s'appuie sur le **modèle relationnel**, et sur des théories mathématiques solides

Les données sont organisées en **relations**, représentées par des **tables**

Transactions et principe ACID

Une transaction est une **opération sur les données**

Elle est souvent composée de **plusieurs instructions**

Dans une base de données relationnelle, une transaction doit respecter le **principe ACID** et ses quatre propriétés :

Atomicité, Cohérence, Isolation, Durabilité

Le principe ACID

Atomicité

Une transaction doit être soit totalement réalisée, soit annulée

Elle aboutit **entièrement ou pas du tout**

Si une partie de la transaction échoue, les modifications doivent être annulées, et les données restaurées à leur état initial

Le principe ACID

Cohérence

Une transaction ne peut faire passer la base de données que d'un état cohérent à un autre

Elle doit **conserver l'intégrité** de la base de données, et en respecter toutes les **contraintes** et les **règles**

Cela empêche la corruption de la base de données par une transaction illégale

Le principe ACID

Isolation

Chaque transaction doit être exécutée de manière **isolée**

Même quand des transactions sont exécutées simultanément, il n'y a pas d'interférences entre elles

Cela permet d'éviter les conflits d'**accès concurrents** à la base de données

Le principe ACID

Durabilité

Lorsqu'une transaction a été validée, les modifications doivent être **enregistrées** de manière permanente dans la base de données, même en cas de défaillance du système

Le principe ACID garantit la **fiabilité** et l'**intégrité** des transactions

Il existe toutefois des modèles de bases de données qui s'en écartent

Des exemples de SGBD **relationnels** (SGBDR) populaires:



Les bases de données non-relationnelles : une approche alternative

Les bases de données **non-relationnelles**, ou **NoSQL**, s'écartent du modèle relationnel

Conçues pour traiter des données **non structurées** ou semi-structurées, elles sont de plus en plus utilisées avec l'essor du **Big Data**

Il existe une variété de types de SGBD non-relationnels (orientés clé/valeur, documents, colonnes, graphes...), qui correspondent à **différents usages spécifiques**

Des exemples de SGBD **non-relationnels** populaires :



Les bases de données non-relationnelles relâchent certaines contraintes du modèle relationnel pour répondre à d'autres priorités : gérer des données **massives** et **distribuées**, avec des exigences de **haute disponibilité** et de **scalabilité**

Elles supportent en général un modèle de **cohérence éventuelle**, ou **BASE** : Basically Available, Soft-state, Eventually consistent

Des avantages et des limites propres à chaque modèle

Les bases relationnelles et non-relationnelles traitent les informations d'une façon différente, et répondent à des **besoins différents**

Il faut choisir le type de base de données adéquat **en fonction du cas d'usage**

Le théorème CAP

Consistency, Availability, Partition Tolerance

Cohérence, disponibilité, distribution

D'après le théorème CAP, entre ces trois aspects, la base de données doit faire des **compromis**

Dans toute base de données, on ne peut respecter **au maximum** que **deux propriétés** parmi les trois

CA

Consistency + Availability : **cohérence + disponibilité**

En général on retrouve ce modèle dans les systèmes de gestion de bases de données **relationnelles**

Un système **centralisé** : les données ne sont pas répliquées à plusieurs même endroits

➡ Peut être problématique pour la scalabilité, ou en cas de panne

CP

Consistency + Partition Tolerance : **cohérence + distribution**

Les données sont **distribuées** et **répliquées sur plusieurs serveurs**

Avec une cohérence **forte** : tous les nœuds voient **exactement** les mêmes données au même moment

➔ Nécessite de **synchroniser** les serveurs entre eux, implique un temps de latence, des **temps de réponse** plus longs

AP

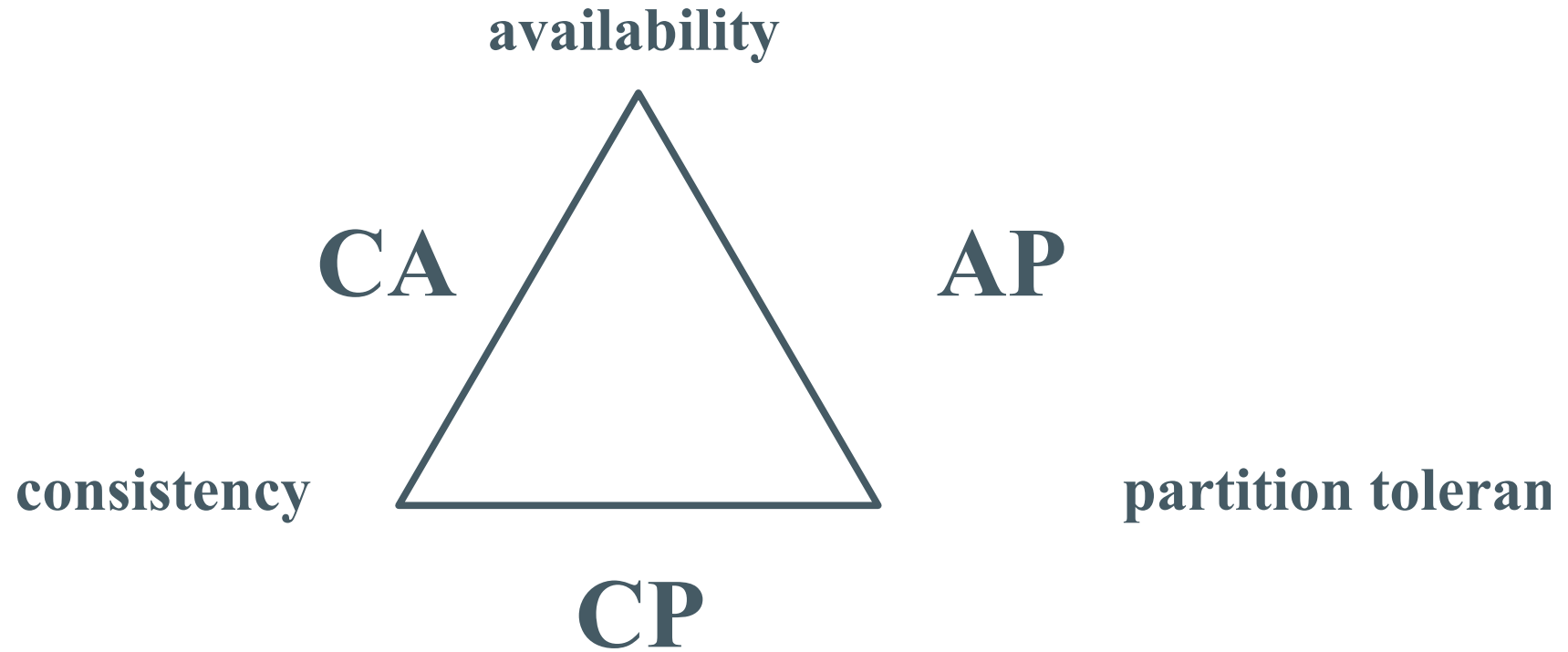
Availability + Partition Tolerance : **disponibilité + distribution**

Des **données distribuées** qui offrent un **temps de réponse** rapide

On retrouve l'idée de **cohérence éventuelle** (eventually consistent)

➡ Des temps de réponses très rapides, mais une possibilité de **divergence** des données, des **conflits** qu'il va falloir gérer et résoudre

Besoin de **monitoring** et de stratégies de gestion des pannes



➔ D'où l'intérêt de **bien choisir** sa base de données en fonction de ses **besoins**

Modélisation des bases de données

Structurer et relier les données de manière logique

La modélisation consiste à concevoir :

- la **structure** des données qui vont être stockées dans la base de données
- les **relations** entre les données

Ceci dans le but de répondre aux besoins fonctionnels d'une application

La modélisation garantit l'intégrité et la cohérence des données : on va définir des **contraintes d'intégrité**, des **règles** que doivent respecter les données lors de leur création et de leur manipulation

La modélisation des bases de données permet de **structurer les données** de manière **logique et cohérente**

Il existe **différentes étapes** dans la modélisation de bases de données, l'identification et la définition :

- des **entités** et de leurs attributs
- des **relations**, des **associations** entre ces entités
- des **contraintes**

La structure

Quels sont les concepts présents dans vos données ?

Formulez les concepts principaux, et leurs caractéristiques

Exemple : un employé

n° employé : 12345

nom : Untel

qualification : ingénieur

lieu de travail : Utopios

➔ Ces concepts seront représentés par des tables

Qu'est-ce qu'une table ?

Une **table**, une **relation**, est un tableau à deux dimensions qui contient des données

Elle se compose de lignes et de colonnes.

Une table porte un nom

Les tables

Colonne

Nom de la table

entête 1	entête 2	entête 3	entête 4
	uplet 1 (ligne)		
	uplet 2 (ligne)		
	uplet 3 (ligne)		

DATABASE (Base de données)

Table 1

C1	C2	C3

Table 2

C1	C2	C3

Table 3

C1	C2	C3

Les lignes

enregistrements, lignes, tuples, n-uplets

Elles doivent toutes être différentes, on ne peut pas avoir deux fois la même ligne

➔ **Clé primaire** (Primary Key, **PK**)

C'est une colonne (ou parfois une combinaison de colonnes), souvent un nombre entier, qui **identifie** chaque ligne de manière **unique**, qui permet de distinguer les lignes les unes des autres

Les colonnes

Les colonnes de la table sont les **attributs**

Un attribut a :

- un nom
- une valeur
- un type
- un domaine (= l'ensemble des valeurs possibles, autorisées, que peut prendre un attribut)

Un attribut est **dérivé** lorsqu'on peut le déterminer à partir d'autres attributs

Exemples :

- prix HT et TVA → prix TTC
- date de naissance et date d'aujourd'hui → âge

➡ On ne va pas les stocker dans la base de données, ce serait redondant

Les associations

On doit chercher les liens logiques qui existent entre ces concepts, les **associations** entre les tables

Relations **one to one**, un-à-un : une entité est associée à une autre entité

Relations **one to many**, un-à-plusieurs : une entité peut être associée à un nombre quelconque d'autres entités

Relations **many to many**, plusieurs-à-plusieurs : un nombre quelconque d'entités qui sont associées à un nombre quelconque d'entités

Comment on va inscrire ces associations dans la base de données ?

Clé étrangère

(Foreign Key, **FK**)

Elle sert à lier plusieurs tables entre elles, à établir une **relation** entre deux tables

L'attribut clé étrangère doit renvoyer à la **clé primaire** d'une autre table

Une clé étrangère doit **obligatoirement** faire référence à une ligne existant dans une autre table (sinon les données sont incohérentes)

Des tables qui représentent des liens entre d'autres tables :

Les tables d'association

ou tables de jointure

Elles sont utilisées dans les relations many-to-many

On obtient ainsi le **schéma** de la table : l'ensemble des colonnes de cette table :

- nom de la relation
- attributs
- clé primaire
- clé étrangère

et le schéma relationnel : l'ensemble de **tous les schémas des tables** de la base de données

Implémentation informatique de la base de données

Pour créer la structure de la base de données, le schéma sera traduit en **langage SQL**, en langage de définition de données

```
CREATE TABLE Employe (  
  id INT,  
  nom VARCHAR(32) NOT NULL,  
  prenom VARCHAR(32),  
  date_naissance DATE,  
  poste VARCHAR(100),  
);
```

Beaucoup de méthodes différentes pour concevoir une base de données

La modélisation de bases de données se fait généralement en utilisant un **langage de modélisation**

Ils permettent de représenter les différentes entités (des objets, concepts, personnes...) et les **relations** entre ces entités

Il en existe différents types, dont voici les deux plus connus :

- Le **modèle Entité-Association** de la méthode **Merise**
- Le **diagramme de classes**, qui fait partie de l'**UML**

UML : Unified Modeling Language

Le langage de modélisation UML est né en 1994, mais **UML v2.0** date de 2005. Il s'agit d'une version majeure apportant des innovations radicales et étendant largement le champ d'application d'UML

L'avantage : c'est un langage commun aux développeurs

Il existe de **nombreux** types de modèles UML, dans ce cas précis on peut utiliser le **diagramme de classes**

Merise : le modèle entité-association

ou entity-relationship model (ER)

Le diagramme **entité-association** permet d'avoir une représentation graphique de la base de données, ce qui simplifie la compréhension

Nous l'aborderons plus en détail par la suite

SQL

Le langage SQL

Le langage SQL (*Structured Query Language*) est le langage de **requêtes** qui va nous permettre d'**interagir** avec notre base de données via le système de gestion de base de données.

Conçu au début des années 70 par IBM, il a été normé par l'American National Standards Institute (ANSI) **en 1986** puis mondialement par l'International Organization for Standardization (ISO) **en 1989**.

Il reçoit encore fréquemment des mises à jour (sa dernière date de 2023) et est le langage de requêtes le plus utilisé au monde.

Le langage SQL

Quelques mises à jours majeures:

- **SQL-92** (1992)
Ajout des requêtes récursives et des déclencheurs
- **SQL:1999** (1999)
Support des objets et des procédures stockées
- **SQL:2011** (1992)
Ajout des tables temporaires et des opérations de fusion.
- **SQL:2023** (2023)
Support des fichiers JSON

Les extensions de langage

Le choix de SGBDR n'est pas anodin pour l'apprentissage du SQL, car de nombreux SGBDR ont leur propre **extension du langage** SQL qui apportent des **fonctionnalités supplémentaires** et peuvent parfois modifier la syntaxe.

Attention donc, certaines syntaxes de requêtes qui sont valables dans un SGBDR auront des syntaxes différentes dans d'autres

Cependant, la logique derrière ces différentes syntaxes reste la plupart du temps très similaire.

Exemples d'extensions de langages

- **PL/SQL** (*Procedural Language*)

C'est l'extension créée par Oracle. Elle ajoute des fonctionnalités de programmation telles que les boucles, les fonctions...

- **PL/pgSQL**

C'est l'extension utilisée par le SGBD PostgreSQL. Sa logique est proche du PL/SQL.

- **T-SQL** (*Transact-SQL*)

L'extension spécifique de Microsoft SQL Server et Sybase.

Les sous-langages

Le SQL est un langage très étendu qui s'est vu augmenté de nombreuses fonctionnalités au fil des années. Il permet de faire diverses manipulations avec les bases de données relationnelles.

Par souci de clarté, il est souvent découpé en 5 grandes sous-catégories de **fonctionnalités spécifiques**, elles sont appelées les **sous-langages SQL**.

Définition des sous-langages

- **Data Query Language (DQL):** Le langage de requête de données est utilisé pour interroger une BDD et en lire ses données.
- **Data Manipulation Language (DML):** Le langage de manipulation de données est utilisé pour modifier, ajouter ou supprimer des données au sein d'une base de données.
- **Data Definition Language (DDL):** Le langage de définition de données est utilisé pour définir la structure des objets de la base de données

Définition des sous-langages

- **Data Control Language (DCL):** Le langage de contrôle de données est utilisé pour gérer les autorisations et les privilèges d'accès à la base de données.
- **Transaction Control Language (TCL):** Le langage de contrôle des transactions est utilisé pour gérer spécifiquement les transactions dans une base de données.

MySQL

Pour ce module, notre choix va se porter sur l'un des SGBDR les plus utilisés: **MySQL**

- **Cross-platform** : Compatible avec la plupart des systèmes d'exploitation: Windows, MacOS, Linux...
- **Populaire** : MySQL dispose d'une vaste communauté d'utilisateurs qui fournissent un support, partagent des connaissances et contribuent à son amélioration continue.
- **Gratuit** : MySQL est un logiciel open source, il est disponible gratuitement sous la licence publique générale GNU (GPL).

Pour travailler sur nos bases de données depuis notre ordinateur, plusieurs possibilités d'interfaces s'offrent à nous

Les deux plus accessibles étant:

- L'interface en **ligne de commande**, ou CLI (Command Line Interface) est disponible depuis n'importe quel ordinateur via le terminal, plus flexible, mais plus difficile à appréhender

- L'interface **graphique**, ou GUI (Graphical User Interface) est plus accessible pour les débutants et les non-développeurs, mais la surcouche graphique engendre une consommation de ressources supérieure

Pour l'instant, nous allons travailler depuis un GUI.

MySQL est compatible avec plusieurs logiciels, les plus populaires étant:



Dans le cadre de ce cours, nous allons choisir **MySQL Workbench**

MySQL Workbench

C'est l'outil officiel fourni par **Oracle**, le propriétaire de **MySQL**

Il fournit donc une intégration complète de MySQL et est régulièrement mis à jour pour être compatible avec les dernières versions

Il offre également un support multi-plateformes

