*Introduction aux bases de données*

*Notes de cours*



*Johnny Piette*

Table des matières

[I. Introduction 3](#_Toc70488780)

[1. Mise en situation 3](#_Toc70488781)

[2. Brainstorming : Qu’auriez-vous pu faire pour garder les résultats ? 3](#_Toc70488782)

[3. Approche intuitive des SGBD 3](#_Toc70488783)

[II. Base de données 4](#_Toc70488784)

[1. Définition 4](#_Toc70488785)

[2. Système de gestion de base de données (SGBD) 4](#_Toc70488786)

[3. Base de données relationnelles 4](#_Toc70488787)

[4. SQL 5](#_Toc70488788)

[III. Modèle conceptuel de données (MCD) 5](#_Toc70488789)

[1. Entité 5](#_Toc70488790)

[2. Clef d’identité 6](#_Toc70488791)

[3. Relation/association 7](#_Toc70488792)

[4. Cardinalités/Multiplicité ou combien ? 7](#_Toc70488793)

[5. Passage du MCD au MLD 7](#_Toc70488794)

[Règle n°1 7](#_Toc70488795)

[Règle n°2 8](#_Toc70488796)

# Introduction

## Mise en situation

Vous avez fait un super programme en Python qui va faire la moyenne des points obtenus pour le cours de Python. Le résultat est stocké en mémoire. Vous avez encodé les résultats de 4 classes composées de 30 élèves chacune. C’est cool. Vous avez le résultat désiré : la moyenne de la classe en python de chaque classe et la moyenne totale de toutes classes confondues. Bien joué, vous assurez ! 😊

Vous éteignez votre ordinateur. Vous allez en terrasse car elles ont réouvert le 8 mai. Et boum, le lendemain, mal de crâne et le trou noir de la journée précédente. Votre directeur vous téléphone et vous demande la moyenne/classe et la moyenne générale. Mais impossible de vous en souvenir… Aie aie aie cette journée commence très mal…

## Brainstorming : Qu’auriez-vous pu faire pour garder les résultats ?

A partir de l’exemple précédent que peut-on retirer comme conclusion ?

## Approche intuitive des SGBD

De la faiblesse de notre programme stocké en mémoire. On constate que l’on aurait pu stocker notre résultat dans un fichier. Et le relire par après. Ce qui est déjà une très grande évolution par rapport à un stockage en mémoire vive.

Cependant, notre directeur a dû nous téléphoner pour avoir le résultat. Il est déjà dommage d’avoir eu besoin de transmettre ce fichier.

Dans une infrastructure de type entreprise, ça ne posait pas de problème car notre directeur aurait eu accès à ce fichier via le réseau d’entreprise : Domaine, système de fichiers, partages, etc…

Imaginons maintenant que le conseil de classe de ces 4 classes se déroulent en même temps. Le directeur fournit le fichier aux 4 titulaires de classe.

Chaque titulaire manipule un fichier car il est possible que l’on donne la moyenne aux élèves proches de la moyenne. C’est tout le débat d’un conseil de classe.

Chaque titulaire devra envoyer son fichier au directeur. Et le directeur devra remettre le tout dans un fichier reprenant les modifications de chaque titulaire : aie aie aie sur autant d’élèves les risques d’erreurs commencent à augmenter avec autant de manipulations manuelles.

Idéalement il aurait été très intéressant que chaque titulaire puisse encoder ses modifications et que ces modifications soient prises en compte. Avec un programme, il est plus compliqué de faire la gestion concurrentielle d’un même fichier. Mais ça reste faisable. Mais de nos jours nous avons recours à ce qu’on appelle des bases de données. Le programme se connecte sur une base de données où seraient stockés les résultats de tous les élèves de toutes les classes. L’accès concurrent est généré nativement : chaque titulaire peut modifier en même temps la base de données.

# Base de données

## Définition

Une base de données est un outil informatique qui permet d’organiser des informations de façon sécurisée, hiérarchisée et sans doublon. Appelée Database en anglais (on voit souvent l’abréviation db), les bases de données sont des logiciels qui permettent surtout de mieux travailler.

C’est donc une collection structurée de données cohérentes, intègres, protégées et accessibles simultanément aux utilisateurs

Concrètement, les informaticiens se sont rapidement retrouvés face à des problèmes difficiles à résoudre en termes de performance et d’intégrité. Comment s’assurer qu’une information saisie dans un système informatique est unique, toujours bien rangée et correctement protégée contre les mauvaises manipulations ?

Un SGBD peut gérer plusieurs bases. En effet, on pourrait très bien avoir les données des cours d’une école comme base et une autre base pour les données d’un garage de voitures Tesla. C’est tout à fait possible et même utile pour sécuriser et séparer des données différentes.

## Système de gestion de base de données (SGBD)

Le SGBD n’est que l’application concrète de la base de données. Sans SGBD, la BD reste un outil théorique « sur papier ». Le SGBD permet concrètement de mettre en place le travail de modélisation et de se servir de la base de données imaginée.

Implémenter une base de données dans un SGBD impose d’arrêter son choix sur un outil. Pour le choisir, il faut avoir réfléchi aux contraintes et caractéristiques de la base de données (volume d’information, accès depuis un même lieu ou pas, droits et accès simultanés) … Baser son choix de SGBD uniquement sur les outils disponibles (par exemple Ms Access parce qu’il est installé avec la suite Ms Office) est à coup sûr une mauvaise idée.

Un Système de Gestion de Base de Données (SGBD) est un logiciel qui permet de stocker des informations dans une base de données. Un tel système permet de lire, écrire, modifier, trier, transformer ou même imprimer les données qui sont contenus dans la base de données.

De plus, un SGBD est sécurisé via l’utilisation d’utilisateurs, de mots de passe. On peut définir les objets que l’utilisateur pourra utiliser dans notre SGBD : bases, tables, procédures stockées, etc.

Dans le cadre de ce cours, nous utiliserons MySQL comme SGBD. Car elle c’est un logiciel libre. Elle existe en version libre et propriétaire. En version propriétaire et donc payante vous aurez un support de la part de Sun qui a racheté MySQL en 2010.

## Base de données relationnelles

La révolution apportée par le modèle relationnel réside dans une indépendance totale par rapport au modèle physique. Défini par Codd en 1970 sur des bases purement mathématiques, ce modèle s’affranchit résolument de toute contrainte matérielle. Cela explique qu’il ait démarré assez lentement, parce qu’il exigeait les machines puissantes dont nous disposons seulement aujourd’hui.

## SQL

Les SGBD offrent un langage d’interrogation des données qui s’appelle le SQL. Le SQL a connu plusieurs normes. On pourrait penser que le SQL est un langage universel et identique à tous SGBD. Et bien non ! En gros oui mais avec parfois des adaptations mineures mais posant un problème majeur : mes commandes SQL ne seront pas exactement les mêmes si je veux changer de base de données. Exemple : passer d’Oracle à MySQL. La volonté de ne pas uniformiser vient du fait qu’un fabricant de SGBD n’a pas envie que vous quittiez son SGBD très facilement. De plus, certains fabricants offrent de belles fonctionnalités par rapport à la concurrence. Donc si vous utilisez des fonctionnalités très spécifiques et propriétaires à un SGBD. Il vous sera difficile de migrer vers un autre SGBD ou au prix de beaucoup d’efforts. Mais pour des requêtes classiques la syntaxe de votre commande SQL sera la même partout.

# Modèle conceptuel de données (MCD)

Dans la méthodologie Merise destinée à créer des bases de données, il y a des outils dédiés aux traitements et aux données. Le MCD (Modèle Conceptuel des Données) est un des outils majeurs concernant les données. Ce modèle décrit une situation à analyser à l’aide d’entités et de relations. Des entités peuvent être liées à une autre entité par l’intermédiaire de ce qu’on appelle une relation.

Une fois notre modèle fait, on convertira nos entités en tables et nos relations en clés primaires et clés étrangères. Nous verrons plus loin cela.

## Entité

Une entité a un nom unique afin de la manipuler facilement. Plus tard dans l’analyse, l’entité se transforme en table et devient concrètement une table lors de la réalisation effective de la base de données.

Cet ensemble d’informations, l’entité, partage les mêmes caractéristiques et peut être manipulé au sein du système d’information mais aussi en discutant entre informaticiens et personnes du métier.

Reprenons notre exemple sur les résultats de nos élèves et les notes des élèves pour le cours de python.

Si on devait essayer de rassembler les informations de base d’un élève : de l’entité élève. Quelles sont les propriétés qui le caractérisent et permettent de l’identifier au mieux ?

* Un nom
* Un prénom
* Une date de naissance
* Sexe (F/M/X) : prendre en compte les personnes transgenres. J’ai déjà vu une personne qui avait mis en DB un booléen pour le sexe : EstHomme. On verra plus loin pourquoi ce n’est pas à faire.)
* Un téléphone fixe (ou pas)
* Un gsm (ou pas)
* Email (ou pas)
* Rue
* Numéro d’habitation
* Boîte
* CP
* Ville

Que manque-t-il pour permettre d’identifier de manière unique votre élève ? En effet, certaines données ne sont pas obligatoires : Gsm, téléphone, email.

Pour répondre à cette question. Que demandent les services auxquels vous êtes clients quand vous les appelez ? Ce qu’ils demandent c’est un numéro de client qui vous est propre, unique. Ce numéro vous est attribué. Et ne peut être en double. Dans le cas contraire, ça m’arrangerait qu’une personne ayant le même numéro de client paie mes factures. 😊

Ici dans le cas de notre élève, on pourrait ajouter une propriété pour l’identifier de manière unique : un numéro d’inscription, un matricule, etc. Ce numéro unique peut être un simple numéro incrémenté (incrémenté = augmenté) : 47. Ou bien suivre des règles bien précisent : 1/2021 (Premier étudiant inscrit de l’année 2021).

Chaque propriété de notre élève comporte un type. Reprenons notre entité élève et pour chaque propriété on va typer celle-ci :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nom du champ** | **Type** | **Obligatoire**  **(= PAS NULL)** | **Unique** | **Remarque** |
| Matricule | Entier | Oui | Oui | C’est le seul champ obligatoire et unique. |
| Nom | Chaîne de caractères | Oui | Non |  |
| Prénom | Chaîne de caractères | Oui | Non |  |
| Date de naissance | Date | Oui | Non |  |
| Sexe | Caractère | Oui | Non |  |
| Téléphone fixe | Chaîne de caractères | Non | Oui |  |
| Gsm | Chaîne de caractères | Non | Oui |  |
| Email | Chaîne de caractères | Non | Oui |  |
| Rue | Chaîne de caractères | Oui | Non |  |
| Numéro d’habitation | Chaîne de caractères | Oui | Non |  |
| Boîte | Chaîne de caractères | Non | Non |  |
| CP | Entier | Oui | Non |  |
| Ville | Chaîne de caractères | Oui | Non |  |

La propriété Matricule va s’appeler « clé d’identité ».

## Clef d’identité

La clef d’identité permet d’identifier de manière sûre et fiable notre élève. Cette clé doit être pensée pour qu’il ne puisse JAMAIS y avoir de doublons. La clef peut être composée d’une ou plusieurs propriétés. Les valeurs de clefs d’identités sont uniques et non nulles.

Dans l’entité, cette clef est soulignée pour marquer justement que c’est une clef.

Il arrive souvent qu’on ajoute le préfixe « Id » (pour identifiant) à une clef d’identité. Exemple : Id\_Client, Id\_Etudiant. Cette clef est souvent écrite Id. Rajouter Id\_Eleve dans l’entité Elève est un peu redondant : c’est évident.

Dans 90% des cas, c’est souvent un numéro automatique incrémenté de 1.

## Relation/association

La relation ou association relie plusieurs entités.

Notre entité élève fait partie d’une classe. Une classe est une entité.

Ce qui relie l’entité élève et l’entité classe c’est la relation : « fait partie de » dans le sens élève vers classe.

## Cardinalités/Multiplicité ou combien ?

Dans la théorie des ensembles, la cardinalité est une propriété des ensembles, y compris infinis, qui généralise la notion de nombre d’éléments aux ensembles finis.

Les cardinalités sont des couples de valeurs que l’on retrouve entre chaque entité et ses relations. La première valeur est la valeur minimale et la seconde est la valeur maximale.

Il existe trois valeurs : 0, 1 ou N (>1 : plusieurs)

Exemple :

* Entité Élève
* Relation : Fait partie de
* Entité Classe

Les cardinalités traduisent des règles de gestion.

Un élève fait partie d’une et une seule classe.

Sens Elève vers Classe : cardinalité 1 (minimum) et 1 (maximum)

Et une classe a un ou plusieurs élèves.

Sens Classe vers Elève : cardinalité 1 (minimum) et N (maximum).

## Passage du MCD au MLD

Le passage du Modèle Conceptuel de Données (MCD) au Modèle Logique de données (MLD).

### Règle n°1

#### Une entité devient une table

Une entité devient une table. Dans un SGBD, une table est une structure composée de colonnes. Ces colonnes sont typées et peuvent avoir des contraintes : unique, non nulle, etc. Ces colonnes correspondent aux propriétés de l’entité. Une colonne porte le nom de champ dans une table. Par exemple le champ « prénom » de la table Elève.

Par convention, on n’utilise pas de caractère accentué pour le nom des champs.

Dans l’affichage de l’ensemble des données de notre table. Par exemple la table Élève qui contient tous les élèves, chaque ligne de cette table correspond à ce qu’on appelle un enregistrement qui correspond à un élève.

La valeur prise par un champ pour un enregistrement donné se situe à l’intersection entre l’enregistrement et le nom du champ.

#### La clef d’identité devient la clef primaire

La clef primaire (Primary key en anglais) permet d’identifier de manière unique un enregistrement d’une table. Si on liste tous les enregistrements de la table élèves = si on liste tous les élèves de la table élèves, nous n’aurons pas deux élèves avec la même clef primaire. Le SGBD ne le permettrait pas et provoquerait une erreur si on essayait de le faire. La création d’une clef primaire donne lieu dans les SGBD la création d’un index qui permet aux SGBD de traiter plus rapidement les recherches, les tris. C’est très intéressant quand on brasse une très grosse quantité de données.

#### Une propriété devient un champ

Une propriété d’une entité devient un champ. Ce champ a un type : integer, float, boolean, varchar (chaîne à taille variable), char (chaîne à taille fixe), date (date, datetime, timespamp).

### Règle n°2

Une association de type 1:N (c’est à dire qui a les cardinalités maximales positionnées à « 1 » d’une côté de l’association et à « n » de l’autre côté) se traduit par la création d’une clé étrangère dans la relation correspondante à l’entité côté « 1 ». Cette clé étrangère référence la clé primaire de la relation correspondant à l’autre entité.

Exemple :

Elève Fait partie d’une classe : cardinalité 1:1

Et une classe a un ou plusieurs élèves : 1:N

On ajoutera dans la table Elève la clef primaire de la table Classe. En effet, cette clé permettra d’identifier la classe dont fait partie un élève. Quand on ajoute comme champ la clef primaire d’une autre table, cette clef porte le nom de clef étrangère (Foreign Key en anglais). Contrairement à la clef primaire qui doit être unique dans une table, une même valeur de clef primaire peut y figurer plusieurs fois. Ce qui est logique : plusieurs élèves font partie d’une même classe.