Base de données

1. Exemple

On considère le tableau suivant qui donne divers renseignements sur les abonnés d'une bibliothèque.

ABONNE				
Id	Nom	Prenom	Telephone	Code_postal
1	Dupont	Eric	06456789	29300
2	Bertin	Agathe	06123456	29200
3	Ferte	Solen	06234567	35000
4	Dupuy	Paul	06345678	29000

On peut considérer les lignes à partir de la 3$^{\text{ème}}$ de ce tableau comme des quintuplets {1,Dupont,Eric,06456789,29300}, ...

Cela correspond au produit cartésien de différents ensembles $Id \times Nom \times Prenom \times Telephone \times Code\_postal$ où Id est un sous ensemble de $\mathbb{N}$ et les autres des chaînes de caractères

On appelle **relation** cette partie du produit cartésien. Le tableau illustre donc cette "relation".

Si on note $\mathcal{R}$ cette relation, on pourrait écrire par exemple $\mathcal{R}$(1,Dupont,Eric,06456789,29300) pour signifier que ce quintuplet appartient à cette relation (à cette partie du produit cartésien).

Terminologie.

- On dit qu'un tel tableau est une **relation**. Exceptée la première ligne correspondant au nom du tableau et la deuxième, c'est donc en fait une partie d'un produit cartésien.
- La deuxème ligne donne le nom des ensembles qui permet de définir le produit cartésien. On dit que c'est la ligne des **attributs**.
- L'ensemble des cinq attributs (ensembles) {Id,Nom,Prenom,Telephone,Code_postal} s'appelle **schéma relationnel** de la relation (du tableau) ci-dessus.
- Chacun des 4 quintuplets qui apparait dans les trois lignes 3, 4, 5 et 6 s'appelle **enregistrement**. Leurs cases s'appellent **champs**.
- Le **domaine** d'un attribut c'est un ensemble auquel appartient l'attribut. Par exemple, pour l'attribut Date dans ce tableau, c'est l'ensemble des entiers naturels et, pour les trois autres attributs, c'est l'ensemble des chaînes de caractères.

On suppose que le tableau ci-dessus donne les abonnés d'une bibliothèque contenant des livres.

Le tableau ci-dessous donne les livres de la bibliothèque.

LIVRE					
Id	isbn	titre	auteur	annee_edition	prix
1	1-123-46578-1	Le petit prince	Saint Exupéry	1920	123,00
2	1-123-46578-1	Le petit prince	Saint Exupéry	1920	123,00
3	2-124-46678-1	Le petit prince	Saint Exupéry	1975	12,00
4	3-345-23455-1	Les Misérables	Victor Hugo	1990	5,00
5	1-145-26795-1	Les Misérables	Victor Hugo	1990	5,00
6	2-143-23715-1	Les Misérables	Victor Hugo	1953	35,00
7	2-645-67845-1	L'Assommoire	Emile Zola	1877	2500,00

- Les **attributs** ici sont : Id, isbn, titre, auteur, annee_edition, prix.
- Le sixuplet $\{\}$ est un **enregistrement** de la relation.

Et la bibliothèque gère les emprunts des livres par les abonnés par le tableau suivant

EMPRUNT		
Id_livre	Id_abonnee	date_emprunt
123	32	11/11/17
28	24	12/7/17
32	24	12/7/17

- L'ensemlbe des trois tableaux (ou table) LIVRE, ABONNE et EMPRUNT forment la base de données de la bibliothèque.
- Cette base de données est représenté de manière simplifiée par sa structure :
  - ABONNE(Id, nom, prenom, telephone, code_postal)
  - LIVRE (Id, isbn, titre, auteur, annee_edition,prix)
  - EMPRUNT(id_livre, id_abonne, date_emprunt)

2. Notion de clé

Il est souhaitable que dans une relation (un tableau), on puisse trouver un moyen de sélectionner sans ambiguité un enregistrement (un p-uplet). Il est clair que si dans une relation (tableau) on a deux enregistrements identiques à deux lignes distinctes, cela ne sera pas possible.

Faire référence à l'attribut isbn de LIVRE ne permet pas d'isoler un seul enregistrement, puisqu'apparaît plusieurs fois. En effet, une bibliothèque peut avoir plusieurs livres de la même édition. De même faire référence à l'attribut titre ou à l'attribut auteur ne permet pas d'isoler un seul enregistrement.
Si on fait référence ensemble aux plusieurs attributs non plus, car on peut avoir deux livres de même référence.
Par contre, dans le tableau EMPRUNT, si on fait référence ensemble aux deux attributs id_livre et id_abonne, on peut isoler un seul enregistrement. C'est la notion de **clé**.
On appelle clé d'une relation, un sous-ensemble d'attributs (A_1,A_2,$\cdots$,A_p) tels que, pour tout p-uplets $(x_1, x_2, \cdots, x_p)$ de valeurs, avec $x_i \in A_i$, on a dans la relation (tableau) un unique enregistrement.
Une **clé primaire** est une clé choisie parmi toutes les clés possibles.
Dans le premier LIVRE, l'attribut Id est une clé primaire.
Dans le second tableau ABONNE l'attribut Id a été créé pour faire une clé primaire. On aurait pu s'en passer en utilisant les attributs Nom, Prenom et Telephone. Mais il est plus pratique de créer une clé formée par un seul attribut.
On appelle clé secondaire toute autre clé que la clé primaire.
On appelle clé étrangère toute clé qui fait référence à une clé primaire d'une autre table.

Les clefs id_livre et id_abonne sont des clefs étrangères faisant le lien entre les tables. Il est important de repérer ces liens pour pouvoir manipuler une base de données correctement.

3. Algèbre relationnelle

Remarque. Les instructions proposées pour faire des manipulations sur les relations (tableaux) utilisent le langage de requête **SQL** (Structured Query Language ou, en français, langage de requête structurée).
On présente ci-dessous quelques **requêtes** simples. Pour pouvoir faire une requête sur une relation (tableau), il faut pouvoir y accéder, il faut donc utiliser son nom.

a. Accéder à une table

Pour accéder aux enregistrements d'une relation (tableau), on utilisera l'instruction :

SELECT * FROM <nom_du_tableau>

b. Sélection

Il s'agit d'extraire d'une relation (un tableau) des enregistrements (des p-uplets ou des lignes) qui vérifient une condition.

c. Projection

Exemple de projection mathématique sur un produit cartésien :

$$\begin{array}{ccc} \mathbb{R}^3 & \to & \mathbb{R}^2 \\ (x_1, x_2, x_3) & \mapsto & (x_1, x_3) \end{array}$$

La projection dans les bases de données correspond à extraire d'un tableau un certain nombre de colonnes pour faire une autre tableau. Il s'agit donc de faire une sélection suivant des attributs. On utilisera l'instruction :

SELECT <attribut_1,attribut_2 ,...,attribut_p> FROM <nom_du_tableau>

d. Requêtes sur une relation

i. Requêtes simples

Il s'agit de sélectionner des champs suivants un critère de sélection dans une relation (tableau).

SELECT <attribut_1,attribut_2 ,...,attribut_p> FROM <nom_du_tableau> WHERE <condition>

Il est possible que plusieurs relations aient des attributs identiques. Pour éviter cette répétition on écrira

SELECT DISTINCT <attribut_1,attribut_2 ,...,attribut_p> FROM <nom_du_tableau> WHERE <condition>

Pour la condition, les opérateurs ou commandes possibles sont :

Opérateur ou commande	Signification
$=$	est égal à
$!=$ ou $<>$	est différent de
$>$ ou $<$	est supérieur (ou inférieur) à
$>=$ ou $<=$	est supérieur (ou inférieur) ou égale à
IS NULL	n'a pas de valeur
IS NOT NULL	possède une valeur
LIKE 'a%'	commence par la lettre 'a'
LIKE '%a'	finit par la lettre 'a'
LIKE '%a%'	contient la lettre 'a'
NOT LIKE '%a%'	contient la lettre 'a'

Pour les requêtes combinées, on utilise les opérateurs logiques utilisées en théorie des ensembles : **AND, OR, NOT IN**.

ii. Requêtes combinées

Il est possible que l'on souhaîte que la condition soit plusieurs valeurs d'une table. Pour cela, on peut utiliser les opérateurs **IN** et **NOT IN** et les valeurs seront alors une table d'un attribut défini par un **SELECT**, c'est-à-dire :

SELECT <attribut_1,attribut_2 ,...,attribut_p> FROM <nom_du_tableau> WHERE <attribut_i IN (SELECT . . .)>

iii. **Requêtes avec jointures**

Il s'agit dans ce type de requêtes de mettre plusieurs tableaux en relation, et d'extraire des données.
Pour mettre en relation ces tableaux, on utilise l'instruction **JOIN ON**.
Il est nécessaire d'avoir un critère de combinaison : c'est la nécessité d'identifier les clefs étrangères faisant le lien entre les tables

SELECT ... FROM <table1> **JOIN** <table2> **ON** <table1>.<clefprimaire>=<table1>.<clefetrangere>

Bien sûr, on peut faire plusieurs jointures pour combiner toutes les tables nécessaires.

SELECT ... FROM (<table1> **JOIN** <table2> **ON** <table1>.<CP>=<table1>.<CE>) JOIN <table3> **ON** <table3>.<CP>=<table?>.<CE>)

e. Fonctions d'agrégation

On s'intéresse à des requêtes concernant un sous-ensemble de données ('données agrégées') et portant sur des fonctions comme le maximum, le minimum, la moyenne ...

- **Compter : COUNT()**
- **Moyenne : AVG()**
- **Somme : SUM()**
- **Maximum : MAX()**
  Si il y a plusieurs enregistrements dont l'attribut est le maximum, il n'en retourne qu'un seul.
- **Minimum : MIN()**

- **Agrégation : GROUP BY**
  La commande GROUP BY permet de partitionner une relation suivant un attribut, afin d'appliquer une des fonctions précédentes sur les différentes parties.

- **Agrégation : ORDER BY**
  La commande ORDER BY permet de trier les résultats suivant un attribut.

- **Agrégation : HAVING()**
  Cette instruction permet d'exprimer des conditions pendant l'instruction GROUP BY. On ne l'utilise que si on a une fonction d'agrégation qui ne doit pas prendre en compte que les éléments vérifiant la condition. Autrement dit, elle a la même fonctionnalité que l'instruction WHERE après SELECT ... FROM.

f. Renommer

On a besoin parfois de renommer une colonne (un attribut). C'est parfois le cas par exemple quand on utilise les fonction COUNT, SUM, AVG, MAX ou MIN qui sont définies ci-dessous, car elles créent des colonnes dont le nom n'est pas exploitable pour faire des sélections.
Pour renommer un attribut, on utilise l'instruction **AS**.

SELECT MAX(moyenne),auteur FROM (SELECT auteur, AVG(prix) AS moyenne FORM LIVRE GROUP BY auteur)