On organise ses données en base de données relationnelle lorsque la structure "plate" d'un tableau traditionnel ne suffit pas à faire des recherches d'informations de manière efficace. En effet, la présentation des données en tableau est intimement liée à un "point de vue", un type de requête particulier.

Par exemple, quel défaut voyez-vous à la table ci-dessous ?

nom	prenom	moyenne	maison	referent	dortoir	effectif
Granger	Hermione	19	Gryffondor	MacGonagall	tour	28
Diggory	Cedric	17	Poufsouffle	Chourave	logis	31
Malefoy	Drago	13	Serpentard	Rogue	donjon	25
Potter	Harry	17	Gryffondor	MacGonagall	tour	28
Weasley	Ron	16	Gryffondor	MacGonagall	tour	28

Il faut séparer les entités.

I Rappel du vocabulaire

1 Vocabulaire

Le principe : une base de données est un ensemble de "tableaux" appelés relations ayant des liens entre eux.
Exemples :

relation maisons

nom	referent	dortoir	effectif
Gryffondor	MacGonagall	tour	28
Serdaigle	Flitwick	tour	35
Poufsouffle	Chourave	logis	31
Serpentard	Rogue	donjon	25

relation eleves

nom	prenom	moyenne	maison
Granger	Hermione	19	Gryffondor
Diggory	Cedric	17	Poufsouffle
Malefoy	Drago	13	Serpentard
Potter	Harry	17	Gryffondor
Weasley	Ron	16	Gryffondor

quatre attributs

- nom, de domaine { Gryffondor, Serdaigle, Pourfsouffle, Serpentard } ou chaines de caractères
- ...
- dortoir, de domaine {tour, logis, donjon} ou chaines de caractères
- effectif de domaine $\mathbb{N}$

cette relation est associée au schéma relationnel (nom, prenom, moyenne, maison)

Chaque **enregistrement** (ou valeur) doit être un 4-uplet différent.

relation : table
attribut : variable, champs (field)
domaine : type des attributs, valeurs possibles
schéma relationnel : structure (liste des attributs)
enregistrement : une "ligne" de la table

Le programme n'insiste pas sur la création de base de données (leur conception est hors programme) mais sur leur manipulation en langage SQL. SQL est l'abréviation de Structured Query Language.

Exemples en SQL de ce qu'il serait bon de savoir écrire (tombé surtout en TSI) :

```
CREATE   TABLE IF NOT EXISTS eleves (
             nom TEXT, prenom TEXT, moyenne INTEGER, maison TEXT,
             PRIMARY KEY(nom)    );
INSERT INTO eleves (nom,prenom,moyenne,maison) VALUES ('Lovegood','Luna',15,'Serdaigle');
UPDATE eleves SET moyenne=20 WHERE nom='Granger' AND prenom='Hermione';
```

2 Notion de clé primaire

Une **clé** est un ensemble d'attributs sur lequel la relation est « injective ». Tous les enregistrements sont distincts sur ces attributs. Il est très pratique (mais pas obligatoire) d'avoir un unique attribut comme clé. Tous les enregistrements prennent une valeur différente pour cet attribut, il peut donc servir à lui tout seul à les « identifier ». Parfois, il en faut plusieurs. Une clé est dite **primaire** lorsqu'elle utilise le nombre minimum d'attributs possible.
On souligne le/les attributs choisi comme clé primaire.
On choisit généralement quel(s) attribut(s) servira(ont) de clé primaire avant de créer la table (lors de la conception de la base).

Exemple : Donner une clé primaire pour chacune des relations.

Parfois un attribut n'est pas une clé dans la relation observée mais l'est dans une autre relation : on parle alors de **clé étrangère** (particulièrement utile pour faire des jointures sans redondance).

Exemple : nom est une clé primaire dans la relation maison, cet attribut est une clé étrangère dans eleves sous l'appellation maison.

II Opérations sur une table

Les deux opérations au programme sont

- **projection :** pour une table on décide de ne sélectionner que certains attributs. On obtient une nouvelle table temporaire.
- **sélection :** on ne sélectionne dans une table que les enregistrements dont certains attributs correspondent à des valeurs choisies. On obtient une nouvelle table temporaire.

Toutes ces opérations peuvent se combiner entre elles, dans un ordre si possible optimal pour effectuer des recherches.

Et comment ça s'écrit en SQL ? Toutes les requêtes (sélection et projection) commencent par SELECT et se terminent par un point-virgule.

Pour effectuer la projection :

```
SELECT A1, .. An FROM R;      # on sélectionne les attributs A1,...An dans la table R
SELECT * FROM R;      # on sélectionne tous les attributs dans la table R
```

Attention, en faisant une projection, des enregistrements qui ne différaient que par leurs valeurs pour des attributs désormais non sélectionnés deviennent identiques. Dans la théorie des bases de données, ils sont « instantanément fusionnés » mais dans l'implémentation en SQL c'est à nous de préciser si on veut qu'ils soient fusionnés ou non grâce au mot clé **DISTINCT**.

Exemples :

relation maisons

nom	referent	dortoir	effectif
Gryffondor	MacGonagall	tour	28
Serdaigle	Flitwick	tour	35
Poufsouffle	Chourave	logis	31
Serpentard	Rogue	donjon	25

```
SELECT dortoir
FROM maisons;
```

dortoir
tour
tour
logis
donjon

```
SELECT DISTINCT dortoir
FROM maisons;
```

dortoir
tour
logis
donjon

pour renommer un attribut (ou une table), on effectue une projection en indiquant tous les attributs et en précisant avec le mot clef **AS** le nouveau nom.

Exemple : SELECT nom , referent AS prof_principal , effectif FROM eleves;

Pour effectuer une sélection :

```
SELECT * FROM R WHERE A=a;      # on sélectionne tous les enregistrements de la table R pour lesquels
l'attribue A prend la valeur a.
```

On peut effectuer des sélections composées en utilisant les opérateurs booléens OR, AND...

Exemple :

```
SELECT nom,prenom
FROM eleves
WHERE maison='Gryffondor' AND moyenne>=17;
```

On peut utiliser des sous requêtes et les opérateurs IN ou NOT IN

Exemple :

```
SELECT referent
FROM maisons
WHERE nom IN(   SELECT DISTINCT maison
                FROM eleves
                WHERE moyenne>=16);
```

III Avec plusieurs tables

- entre tables **de même schéma** : On peut réaliser des unions, des intersections, des différences on obtient une « nouvelle » table.
  pour les opérateurs ensemblistes UNION , INTERSECT, et EXCEPT.

  Exemple de syntaxe : SELECT * FROM table1 INTERSECT SELECT * FROM table2 ;

- produit cartésien
  Le produit cartésien de deux relations R et R' , noté $R \times R'$ est l'ensemble des possibilités d'association des valeurs de R et de R'

relation noms

Nom_Famille
Durand
Dupont

relation prenoms

Prenom
Alice
Bob

noms×prenoms

Nom_Famille	Prenom
Durand	Alice
Durand	Bob
Dupont	Alice
Dupont	Bob

  Pour effectuer en SQL le produit cartésien, on effectue un SELECT de deux tables (ou plus)
  SELECT * FROM table1, table2 ;

- La jointure symétrique permet de recoller deux relations de schémas différents du moment que deux de leurs attributs prennent les mêmes valeurs.

Relation Livres

titre	nomauteur
Les trois mousquetaires	Dumas
Notre Dame de Paris	Hugo

Relation Auteurs

nom	prenom
Hugo	Victor
Dumas	Alexandre Fils
Dumas	Alexandre Père

en BDD théorique : jointure Livres[nomauteur=nom]Auteurs

```
SELECT *
FROM Livres
JOIN Auteurs
ON Livres.nomauteur = Auteurs.nom
```

titre	nomauteur	nom	prenom
Les trois mousquetaires	Dumas	Dumas	Alexandre père
Les trois mousquetaires	Dumas	Dumas	Alexandre fils
Notre Dame de Paris	Hugo	Hugo	Victor

Remarque 1 : Il y a une redondance avec les deux attributs sur lesquels a été faite la jointure (il faut donc effectuer une projection)

```
SELECT Livres.titre, Livres.nomauteur, Auteurs.prenom
FROM Livres
JOIN Auteurs
ON Livres.nomauteur=Auteurs.nom ;
```

Remarque 2 : Comme nom n'était pas une clé primaire dans la table auteur (ce qui est normalement le cas le plus fréquent), l'enregistrement Les trois mousquetaire a été "dupliqué" (d'où l'intérêt d'utiliser comme critère de jointure une clé étrangère).

Remarque 3 : On peut obtenir le même résultat avec un produit cartésien suivi d'une sélection mais ce n'est pas ce que recommande le programme d'Informatique pour tous.

```
SELECT Livres.titre, Livres.nomauteur, Auteurs.prenom
FROM Livres, Auteurs
WHERE Livres.nomauteur=Auteurs.nom ;
```

Remarque 4 : On peut faire une jointure sur plusieurs attributs, voire trois tables :
Exemple : Livres[nomauteur=nom, prenomauteur=prenom]Auteurs

```
SELECT Livres.titre, Livres.nomauteur, Auteurs.prenom
FROM Livres
JOIN Auteurs
ON Livres.nomauteur=Auteurs.nom AND Livres.prenomauteur=Auteurs.prenom ;
```

Exercice : On revient à la base HarryPotter

relation maisons

nom	referent	dortoir	effectif
Gryffondor	MacGonagall	tour	28
Serdaigle	Flitwick	tour	35
Poufsouffle	Chourave	logis	31
Serpentard	Rogue	donjon	25

relation eleves

nom	prenom	moyenne	maison
Granger	Hermione	19	Gryffondor
Diggory	Cedric	17	Poufsouffle
Malefoy	Drago	13	Serpentard
Potter	Harry	17	Gryffondor
Weasley	Ron	16	Gryffondor

relation professeurs

nom	prenom	mail
Dumbledore	Albus	a.dumbledore@ac-poudlard.uk
MacGonagall	Minerva	minervaMG@ac-poudlard.uk
Filtwick	Filius	flitwick@gmail.com
Chourave	Pomona	chourave@ac-poudlard.uk
Rogue	Severus	severus.rogue@ ac-poudlard.uk
Hagrid	Rubeus	hagrid@ac-poudlard.uk

a) Combien d'enregistrements comptabiliserait le résultat de la requête :
SELECT * FROM maisons,eleves,professeurs ;

b) Donner une requête qui donne l'adresse du professeur de Cedric Diggory.

c) Quels sont les prénoms des professeurs responsables de maison ?

d) *pour aller plus loin* Quelles sont les maisons dont le referent a une adresse qui ne finit pas par @ac-poudlard.uk

IV Fonctions d'agrégation

Les fonctions d'agrégation dans le langage SQL permettent d'effectuer des opérations statistiques sur un ensemble d'enregistrements.

Les fonctions d'agrégation sont des fonctions idéales pour effectuer quelques statistiques de bases sur des tables. Les principales fonctions sont les suivantes :

- **AVG()** pour calculer la moyenne sur un ensemble d'enregistrements
- **COUNT()** pour compter le nombre d'enregistrements sur une table ou une colonne distincte
- **MAX()** pour récupérer la valeur maximum d'une colonne sur un ensemble de ligne. Cela s'applique à la fois pour des données numériques ou alphanumérique
- **MIN()** pour récupérer la valeur minimum de la même manière que MAX()
- **SUM()** pour calculer la somme sur un ensemble d'enregistrement

Exemple : :
```
SELECT AVG(Moyenne)
FROM eleves ;
```

Toutes ces fonctions prennent tout leur sens lorsqu'elles sont utilisée avec la commande **GROUP BY** qui permet de filtrer les données sur une ou plusieurs colonnes.

Exemple : :
```
SELECT maison, AVG(moyenne)
FROM eleves
GROUP BY maison ;
```

Attention, le résultat d'une fonction d'agrégat est le résultat d'une requête. Il ne peut être utilisé en cours de requête.

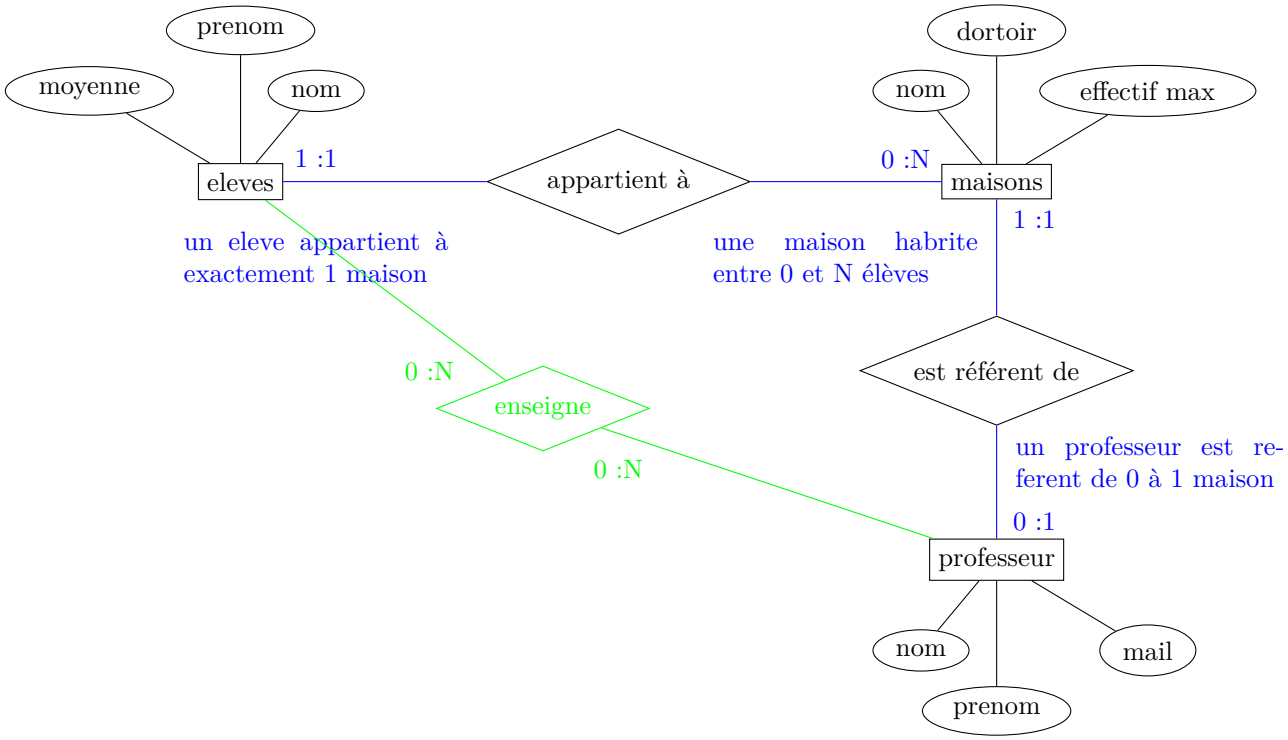
Exemple :
```
SELECT nom
FROM eleves
WHERE moyenne<AVG(moyenne) ;
# ne fonctionne pas !
```

```
SELECT nom
FROM eleves
WHERE moyenne<(  SELECT AVG(moyenne)
                 FROM eleves) ;
```

V Pour aller plus loin : Le modèle Entité association

Le programme d'informatique pour tous se concentre sur l'exploitation d'une base de données déjà créée et remplie. Néanmoins, il est intéressant de réaliser que les bases de données s'appuient sur un modèle Entité-Association.

Lors de la conception des bases de données, on doit identifier les entités avec leurs attributs propres, et les associations qui les relient.



Lorsque l'on passe du modèle Entité association au modèle Relationnel (création de la base de données), les associations (comme appartient ou est référent de) se traduisent :

- soit par l'ajout d'un attribut dans une des tables (possible dans une table dans laquelle la cardinalité de l'association est au plus 1).
  Par exemple, dans la table eleves, on a ajouté un attribut maison. On ne pourrait pas dans la table maisons ajouter un attribut eleves car il y a plusieurs élèves...
  dans la table maisons on a ajouté un attribut referent, on aurait pu faire l'inverse, c'est à dire ajouter un attribut maisons dans la table professeurs.
- soit par la création d'une table d'association si la cardinalité est supérieure à 1. Si l'on rajoutait une relation « est le prof de » : chaque prof peut avoir plusieurs élèves et chaque élève a plusieurs professeurs. Le simple ajout d'un attribut dans une table ne permettrait pas de stocker cette information. On créerait donc une table supplémentaire, voir plusieurs (liste des professeurs par élève, ou des élèves par professeurs), voir l'amélioration du modèle par l'ajout d'une nouvelle entité classes ....