Bases de données

En classe de seconde et de première, on a vu que pour stocker, traiter et manipuler des données on utilisait des structures de données comme les tableaux, dictionnaires ainsi que des fichiers textes tels que CSV ou JSON.
On atteint rapidement les limites de cette façon de faire lorsque :

- on a une grande quantité de données,
- on doit régulièrement mettre à jour les données,
- on a des données complexes,
- on croise les données avec d'autres données
- on doit partager des données...

3.1 Notion de base de données

Une base de données, database en anglais, est une collection de données organisées pour pouvoir être utilisées directement via ce qu'on appelle un Système de Gestion de Base de Donnée (SGBD). Cette collection peut être de très grande taille et contenir des millions d'enregistrements !

Quelques repères historiques :

- **1960** idée des bases de données (dans le cadre du programme Apollo)
- 1970 Edgar Franck Codd (informaticien britannique employé chez IBM, 1923-2003) définit les bases du modèle relationnel.
- 1974 langage SQL (normalisé en 1986)

Définition 1
Une base de données est un ensemble contenant et permettant de retrouver des données structurées ou brutes. Cette base de données possède une organisation des données fixée par ce que l'on appelle un modèle de données.

On retient qu'une base de données est un ensemble structuré et organisé permettant le stockage de grandes quantités d'informations afin d'en faciliter l'exploitation.

Les bases de données peuvent être centralisées, distribuées ou répliquées. Elles peuvent être accessibles par plusieurs utilisateurs simultanément.

a. Les différents modèles de base de données

La cohérence des données est la problématique fondamentale des bases de données, elle peut être mise à rude épreuve lors d'insertion, de modification, de suppression, de pannes. La résolution de ces problèmes est le rôle du « Système de Gestion de Base de Données » (SGBD).

Un SGBD est caractérisé par le modèle de description de données qui le supporte. Ce modèle peut être hiérarchique, réseau, relationnel, objet, noSql...
Le modèle le plus répandu est le modèle relationnel, cependant en fonction du domaine, certains modèles sont plus utilisés que d'autres.

b. Base de données relationnelles

Le modèle relationnel a été conceptualisé et développé par Ted CODD. Il décide d'appliquer des principes mathématiques de la théorie des ensembles. Le modèle relationnel représente la base de données comme un ensemble de tables que l'on appelle relations. Au niveau physique, le système est libre d'utiliser n'importe quelle technique de stockage du moment qu'il est possible de relier ces structures à des tables.
Ce modèle relationnel est caractérisé ainsi :

- les données sont organisées sous forme de tables à deux dimensions, relations, dont les lignes sont des n-uplets ou tuples ;
- les données sont manipulées par des opérateurs de l'algèbre relationnel ;
- l'état de cohérence de la base est défini par un ensemble de contraintes d'intégrité.

Éléments constitutifs du modèle relationnel

Un élément d'une table ou relation est appelé une **entité**. Chaque entité possède des **attributs**.

Nom de la Table ou Relation

Attribut 1	Attribut 2	Attribut 2	← entête
...	...	...	← entité 1
...	...	...	← entité 2

Un attribut est un identifiant décrivant une information stockée dans une base. Exemple, l'âge d'une personne, le nom d'une personne, le numéro de sécurité sociale.

Le domaine d'un attribut est l'ensemble de ses valeurs possibles. Par exemple, l'âge d'une personne est un entier.

Une clé primaire est un attribut qui identifie chaque entité de manière unique.

Une clé étrangère est un attribut d'une table qui est une clé primaire dans une autre table.

Le schéma d'une relation précise le nom de la relation ainsi que la liste de ses attributs avec leurs domaines.

Le schéma relationnel est constitué par l'ensemble des schémas de relation avec mention des clés étrangères.

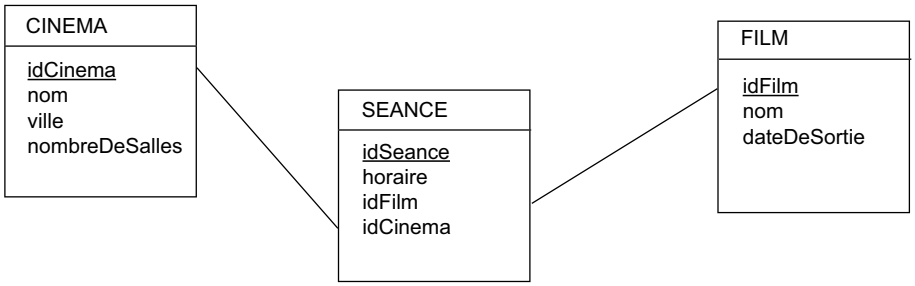
c. Exemple complet

L'exemple ci-dessous reprend celui de l'activité cinemas constitutée de trois fichiers JSON `cinemas.json`, `films.json` et `seances.json`. On l'adapte au modèle relationnel.

CINEMA(idCinema, nom, ville, nombreDeSalles) idCinema est la clé primaire de la relation CINEMA

FILM(idFilm, nom, dateDeSortie) idFilm est la clé primaire de la relation FILM

SEANCE(idSeance, Horaire, idFilm, idCinema) id seance est la clé primaire de la relation SEANCE, idFilm et idCinema sont des clés étrangères.



Il reste à préciser le domaine de chaque attribut, `idCinema`, `idFilm`, `idSeance` seront de type nombre entier `int` ainsi que `nombreDeSalles`, tandis que les autres attributs seront des chaînes de caratères `string` sauf peut-être `dateDeSortie` qui pourrait être, si le SGBD le permet, de type `Date`.

d. Exercices

▶ Exercice 1

On donne la relation suivante :
`LIVRES(numISBN : string, titre : string, numpage : int, annee_edition : int, idAuteur:int)`
Donner les attributs de cette relation et les domaines d'intégrité de ceux-ci. Quelle pourrait être la clé primaire ? Y aurait-il une clé étrangère ?

▶ Exercice 2

On donne un extrait de la relation VOITURE

numImmatriculation	marque	modele	couleur	puissance	dateMiseEnService
ZE-312-KL	Renault	Modus	bleu ciel	6	20/12/2018
TS-205-GH	BMV	M3	gris	10	15/08/2019
BB-404-TF	DACIA	Logan	bleu	5	01/01/2016

1. Quels sont les attributs utilisés ?

2. Quelles sont les règles de domaines de ces attributs ?

3. Les enregistrements sont-ils redondants ?

▶ Exercice 3

Nolwenn possède une playlist musicale constituée de nombreux titres. Chaque morceau de musique a un numéro ID3TAG unique, un titre de la chanson, le nom de l'interprète, le nom de l'album, l'année de parution, le genre musical Elle décide de créer une relation MYMUSIC pour pouvoir gérer plus facilement sa playlist. Indiquer quels attributs et domaines d'intégrité associés à ceux-ci elle devrait utiliser.

3.2 Le langage SQL

SQL (Structured Query Language) est un langage de gestion des bases de données relationnelles pour interroger, mettre à jour, définir les données, contrôler l'accès aux données. C'est le langage standard utilisé par presque tous les SGBDR depuis 1985. Une interrogation de base de données se fait par l'intermédiaire de ce que l'on appelle une requête.

a. Le langage de définition des données

Cette partie de SQL est utilisée pour spécifier le schéma de la base de données, à savoir le type des données, les tables qui permettront de les stocker ainsi que les contraintes. Ce langage permet de créer le schéma d'une base de données ou le modifier, mais il ne peut pas être utilisé pour modifier les données de la base de données (DML). Une base de données est composée de différents types d'objets : les tables, les clés, les index, les liens.
Nous utiliserons comme SGBD MYSQL d'abord en ligne de commande puis par l'intermédiare d'un connecteur python. Dans une console Mysql saisir

```
mysql -u root -p
```

puis saisir le mot passe défini lors de l'installation :

```
Enter password:
```

Suite à quoi nous allons lister les bases de données existantes :

```
 mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0,00 sec)
```

Créons une base de données TEST

```
 mysql> CREATE TEST;
```

Nous allons nous placer et travailler dans la base TEST.

```
 mysql> USE TEST;
 Database changed
```

Examinons le contenu de notre BDD

```
mysql> SHOW TABLES;
Empty set (0,00 sec)
```

Elle est bien entendu vide. Nous allons créer notre première table ou relation en langage SQL. Pour créer une table on utilise le mot clé `CREATE`. Il y a de nombreuses options, une lecture de la documentation est indispensable.

```
CREATE TABLE CINEMA (
  idCinema int(11) NOT NULL  AUTO_INCREMENT PRIMARY KEY,
  nom varchar(50) NOT NULL,
  ville varchar(50) NOT NULL,
  nombreDeSalles int(11) NOT NULL);
```

Les domaines des entités ainsi que des contraintes d'intégrité sont définis.
Par exemple `NOT NULL` signifie que l'attribut concerné ne peut être vide.
La contrainte `PRIMARY KEY` permet de déclarer qu'un ensemble d'attributs est la clé primaire de la table. Il n'y a qu'une clause de ce genre par table, chaque valeur de la clé doit être unique.
De même la contrainte `FOREIGN KEY` permet de déclarer qu'un ensemble d'attributs est une clé étrangère.
Pour finir signalons qu'il existe une commande pour modifier `ALTER TABLE nomdelatable` et une autre pour supprimer `DROP TABLE nomdela table`. Il va de soi que ces deux commandes sont à utiliser avec précaution.

▶ Exercice 4

Créer dans la base de données `TEST` les tables `FILM` et `SEANCE`.

b. Les requêtes de mise à jour des données

Nous verrons dans le paragraphe suivant quelles requêtes permettent d'extraire de l'information d'une base de données. Mais avant cela voyons comment ajouter, modifier, supprimer des données dans une base de données.

Pour ajouter :

```
INSERT INTO matable (attribut1,attribut2, ..., attributn) VALUES
(valeur1, valeur2, ..., valeurn);
```

Ce qui donne pour la table `CINEMA`

```
 INSERT INTO CINEMA (idCinema, nom, ville, nombreDeSalles) VALUES
(0, 'UGC Ciné cité', 'Ludres', 14),
(0,'CinéLun', 'Lunéville',5),
(0,'Kinépolis','Nancy',18),
(0,'Caméo','Nancy',7),
(0,"L'Impérial",'MAxeville',5);
```

Notez que la valeur 0 est automatiquement remplacée par l'entier adéquate car la contrainte d'intégrité `AUTO_INCREMENT` a été saisie lors de la création de cette table.

Pour modifier : Ci-dessus le nom de la ville Maxéville a été mal orthographié `MAxeville` au lieu de `Maxeville`. On peut le modifier avec le mot-clé `UPDATE` :

```
UPDATE CINEMA
SET ville = "Maxeville"
WHERE nom = "L'Impérial";
```

Pour supprimer : on utilise le mot-clé `DELETE`, ici aussi il faut manipuler cette commande avec précaution.

```
 DELETE FROM TABLE
WHERE Condition;
```

▶ Exercice 5

Saisir les données des contenus dans les fichiers `films.json` et `seances.json` dans les tables `FILM` et `SEANCE`

c. Le langage de manipulation des données

Les requêtes de sélection simples : l'obtention de données se fait exclusivement avec lacommande `SELECT`. Cet ordre possède 6 clauses différentes dont seules les deux premières sont obligatoires. Syntaxe de la commande `SELECT`

```
SELECT Liste-Attributs
FROM Liste-tables
WHERE critère de sélection
GROUP BY liste-attributs-groupe
HAVING critère-groupe
ORDER BY Liste-attributs
```

▶ Exercice 6

Tester les requêtes ci-dessous :
`SELECT * FROM CINEMA;` Qu'obtient-on ?
`SELECT nom, ville FROM CINEMA WHERE nombreDeSalles > 5;` Qu'obtient-on ?
`SELECT nom, ville, nombreDeSalles FROM CINEMA ORDER BY nombreDeSalles;` Qu'obtient-on ?
Quelle requête écrire pour obtenir les infos concernant les cinémas de Nancy ?
`mysql>.....`
Requêtes sur plusieurs tables
Pour croiser les informations de plusieurs tables, on dispose du produit cartésien et de la jointure.
On joint les deux tables par l'égalité d'un attribut d'une table avec un attribut d'une autre table.
Afin de ne pas confondre des attributs de même nom de deux tables différentes, on ajoute le nom de la table en préfixe (`MATABLE.nomattribut`)
Produit cartésien : `SELECT * from table1, table2;` on obtiendra alors tous les couples constitués des entités de la table1 et de la table2
Jointure Interne (`INNER JOIN` ou `JOIN`) :
`SELECT * from table1 INNER JOIN table2 ON table1.attribut_x= table2.attribut_y;`
permet de restreindre les résultats du produit cartésien aux couples dont l'`attribut_x` est égale à l'`attribut_y`
Remarque : ce type de jointure peut se faire avec un produit cartésien et une clause `WHERE` (c'est ce que l'on faisait avant l'introduction du la commande `JOIN` avant la version 2 de SQL)
On peut aussi utiliser la jointure naturelle quand les attributs sont communs à deux tables (mais à deux tables uniquement).

▶ Exercice 7

Dans cet exercice on utilise deux jointures avec trois tables `SEANCE <-> FILM` et `SEANCE <-> CINEMA`. Pour répondre aux questions de l'activité d'introduction, completer et utiliser cette requête.

```
SELECT FILM.nom, SEANCE.horaire, CINEMA.nom, CINEMA.ville
FROM SEANCE
INNER JOIN FILM ON SEANCE.idFilm = FILM.idFilm
INNER JOIN CINEMA ON SEANCE.idCinema = CINEMA.idCinema;
```

1. Quelles sont les séances du matin ?
2. Quel(s) film(s) puis-je voir à Lunéville à 14h ?
3. Peut-on voir le film « Contagion » à Ludres ? Si oui à quel(s) horaire(s) ?
4. Dans quel(s) cinéma(s) est diffusé le film « L'armée des douze singes » ?