Informatique

Bases de données

PSI : Lycée Rabelais

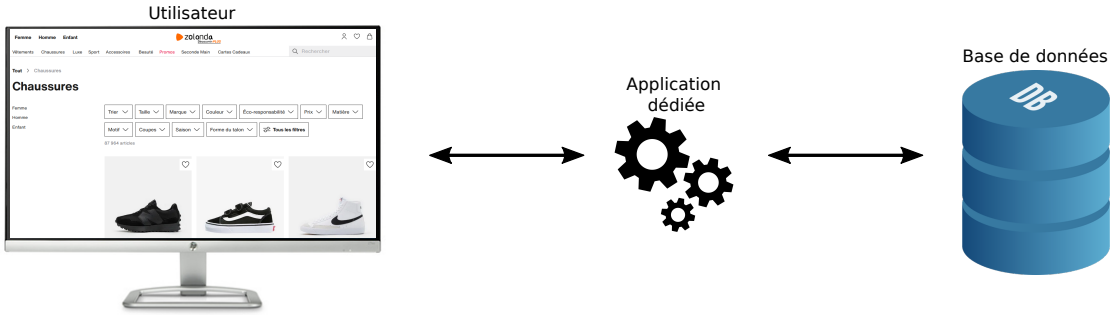
Objectifs

Comprendre la notion de base de données

Être capable d'écrire une requête SQL

1 A quoi ça sert ?

De nos jours, les données consultées à travers un ordinateur, une tablette ou un smartphone sont de plus en plus grandes.



Afin d'accéder à ces données, des protocoles spécifiques ont été créés. Ils fonctionnent selon l'architecture suivante :

- Sur votre ordinateur, vous sélectionnez les données que vous souhaitez consulter sur une interface graphique (par exemple chez un marchand de chaussures, vous cherchez des mocassins adadis en taille 54 et de couleur rose).
- Votre sélection est transmise à un logiciel intermédiaire. Ce logiciel élabore alors une requête afin de communiquer avec la base de données. Le résultat de la requête est renvoyé à l'application.
- L'application transfère ce résultat sur votre ordinateur afin de les afficher.

L'objet du cours est d'apprendre à écrire la requête permettant d'interroger la base de données.

Cette base de données est un fichier qui est constitué d'un ensemble organisé et structuré d'informations sous la forme de tables (tableaux) indépendants. Pour gérer les données enregistrées dans une base, on utilise un système de gestion de bases de données. Ce sont des logiciels qui permettent à des utilisateurs ou des programmes d'exprimer des requêtes pour interroger des bases de données ou pour les modifier.

Plusieurs systèmes de gestion de base de données existent. Nous nous focaliserons ici sur l'un des plus répandus d'entre ces systèmes, le système relationnel, parmi lesquels nous trouvons des logiciels propriétaires (ORACLE, Microsoft SQL server, ...) et des logiciels libres très utilisés (MySQL, SQLite, ...).

Python propose également un module (sqlite3) qui permet d'interroger des bases de données. De gros efforts de standardisation ont été effectués et un langage de requêtes commun aux différents systèmes de gestion de base de données est né : **le langage SQL** (Structured Query Language).

2 La structure des données, le modèle relationnel

2.1 Schéma général

Le modèle relationnel est le principal modèle employé par les systèmes de gestion de base de données. Le but du modèle relationnel est de percevoir une réalité sous la forme de tableaux (tables) de valeurs à deux dimensions appelées relations :

- une relation (table) a un nom ;
- une colonne d'une relation est un **attribut** ;
- une ligne d'une relation est un **tuple** ou un **n-uplet**.

Pour illustrer ces définitions, nous nous appuierons sur la base de données `plan_de_vol` issue du sujet *Centrale 2016*. Cette base de données contient deux tables dont le descriptif est donné ci-dessous.

La table `vol` répertorie les plans de vol déposés par les compagnies aériennes. Elle contient les colonnes suivantes :

- `id_vol` : numéro du vol (chaine de caractères) ;
- `depart` : code de l'aéroport de départ (chaine de caractères) ;
- `arrivee` : code de l'aéroport d'arrivée (chaine de caractères) ;
- `jour` : jour du vol (de type date, affiché au format aaaa-mm-jj) ;
- `heure` : heure de décollage souhaitée (de type time, affiché au format hh:mi) ;
- `niveau` : niveau de vol souhaité (entier).

id_vol	depart	arrivee	jour	heure	niveau
AF1204	CDG	FCO	2016-05-02	10:25	300
AF1205	FCO	CDG	2016-05-02	07:35	300
AF1504	CDG	FCO	2016-05-02	06:05	310
AF1505	FCO	CDG	2016-05-02	13:00	310
AF2305	FCO	MRS	2016-05-02	12:00	320
EZ0005	ORY	MRS	2016-05-02	17:00	320
.....	....	....	.....	.....	.....

Extrait de la table `vol` : vols de la compagnie Air France entre les aéroports Charles de Gaules (Paris) et Léonard-de-Vinci (Fiumicino-Rome)

La table `aeroport` répertorie les différents aéroports européens. Elle contient les colonnes suivantes :

- `id_aero` : code de l'aéroport (chaine de caractères) ;
- `ville` : principale ville desservie (chaine de caractères) ;
- `pays` : pays dans lequel se situe l'aéroport (chaine de caractères).

id_aero	ville	pays
CDG	Paris	France
ORY	Paris	France
MRS	Marseille	France
FCO	Rome	Italie
....	.....	.....

Extrait de la table `aeroport`

2.2 Les attributs et leur domaines

Dans la table (ou relation) `vol`, les noms des attributs sont : `id_vol`, `depart`, `arrivee`, `jour`, `heure` et `niveau`. Dans la table (ou relation) `aeroport`, les noms des attributs sont : `id_aero`, `ville`, et `pays`.

Les différentes valeurs prises par un même attribut appartiennent toutes à un même ensemble, appelé **le domaine de l'attribut**. Suivant son domaine, chaque attribut est représenté par un type. Les principaux types rencontrés sont les entiers (`INT`), les flottants (`REAL`), les chaînes de caractères (`CHAR` ou `VARCHAR(20)`, l'entier précisant la longueur maximale), les dates (`DATE`) et les temps (`TIME`).

Le schéma relationnel de la table `vol` est par exemple :

```
(id_vol :  CHAR ; depart :  CHAR ; arrivee :  CHAR ;
    jour :  DATE ; heure :  TIME ; niveau :  INT)
```

Les attributs ne sont pas ordonnés : demander le " premier attribut" de la table n'a pas de sens ! Par contre chaque attribut possède un nom qui lui est propre (`id_vol`, `depart`...) et qui permet d'identifier l'attribut de façon univoque (deux attributs distincts d'une même table ne peuvent pas avoir le même nom).

2.3 Clé primaire et clé étrangère

Dans une relation, l'ordre des lignes n'est pas important. En revanche, chaque tuple doit être unique. Pour les identifier, on définit une clé primaire qui peut être constituée d'un ou plusieurs attributs. En pratique la clé primaire est souvent limitée à un seul attribut qui prend des valeurs distinctes pour chaque entrée de la table.

Exemple : Dans la table `vol`, l'attribut `id_vol` joue le rôle de clé primaire. Dans la table `aeoroport`, c'est l'attribut `id_aeroport` qui joue le rôle de clé primaire.

S'il existe plusieurs clés possibles, on parlera de clé primaire (ou clé principale) pour la clé choisie et de clés secondaires pour les autres clés possibles. Pour des raisons d'efficacité lors de l'interrogation d'une table, il est nécessaire de disposer d'une clé la plus simple possible.

Une colonne (ou un ensemble de colonne) dont le rôle est de référencer une colonne d'une autre table est dénommé clé étrangère.

Exemple : Dans la table `vol`, les attributs `depart` et `arrivee` sont des clés étrangères issues de la table `aeoroport`.

Ces clés sont essentielles lorsque l'on souhaite obtenir des informations qui nécessitent la lecture de plusieurs tables. On peut les représenter par des flèches entre les tables. La donnée des tables d'une base de donnée et des clés étrangères qui les relient les unes aux autres constituent le schéma relationnel global de la base de données.

id_aero	ville	pays
CDG	Paris	France
ORY	Paris	France
MRS	Marseille	France
FCO	Rome	Italie
....	.....	.....

aeroport

id_vol	depart	arrivee	jour	heure	niveau
AF1204	CDG	FCO	2016-05-02	10:25	300
AF1205	FCO	CDG	2016-05-02	07:35	300
AF1504	CDG	FCO	2016-05-02	06:05	310
AF1505	FCO	CDG	2016-05-02	13:00	310
AF2305	FCO	MRS	2016-05-02	12:00	320
EZ0005	ORY	MRS	2016-05-02	17:00	320
.....	....	....	.....	.....	.....

vol

3 L'algèbre relationnelle et interrogation de données en langage SQL

3.1 Présentation succincte et formalisme

La théorie des bases relationnelles est très proche de la théorie des ensembles en mathématiques : une table est assimilable à un ensemble mathématique. On entend par algèbre relationnelle, une collection d'opérations qui agissent sur des relations (tables) et produisent une relation (table) en résultat : $R3 \leftarrow R1$ Opérateur $R2$.

L'aspect théorique des bases de données n'est pas au programme mais **il faut savoir écrire une requête (au formalisme SQL) permettant d'accéder aux données voulues.**

Les requêtes SQL ne respectent pas la même syntaxe que le langage Python. On notera par exemple que :

- Le code n'est pas soumis à des règles d'indentation ;
- Majuscules et minuscules peuvent être confondues pour tous les mots clés (mais pas pour la nomination des attributs) ;
- **L'ordre des mots-clés est primordial !**

3.2 Les opérateurs ensemblistes et relationnels usuels

3.2.1 Sélection

La sélection consiste à ne retenir que certaines colonnes d'une table. Sélectionner les attributs A_i et A_j de la relation $R1$ s'écrit :

```
SELECT Ai , Aj FROM R1
```

Exemple : Écrire la requête donnant les codes de départ de tous les vols de la table `vol` et le résultat obtenu.

Utilisation de mots-clés :

- La commande pour sélectionner la totalité des attributs et donc visualiser la totalité de la table $R1$ s'écrit : `SELECT * FROM R1`
- On utilise pour éviter les doublons la commande : `SELECT DISTINCT`. Pour obtenir la table des différents codes de départ de la table `vol`, on écrira :

- On peut rajouter la commande `LIMIT` pour limiter le nombre de tuples sélectionnés à la valeur spécifiée. Par exemple, la requête suivante

```
SELECT id_vol,heure
FROM vol
LIMIT 3
```

  renverra :

- La commande `OFFSET` peut-être utilisées en complément de `LIMIT` pour "décaler" les tuples sélectionnés. Par exemple, la requête suivante

```
SELECT id_vol,heure
FROM vol
LIMIT 2
OFFSET 3
```

renverra :

- Pour faire les choses proprement, il faudrait toujours, en complément de `LIMIT` et `OFFSET`, utiliser la commande `ORDER BY` qui permet de spécifier quelle est la règle pour l'ordonnancement. Par exemple, la requête suivante

```
SELECT id_vol,heure
FROM vol
ORDER BY heure
LIMIT 2
OFFSET 3
```

renverra :

On peut rajouter `ORDER BY heure ASC` pour spécifier que c'est bien un classement dans l'ordre croissant (par défaut) ou `DESC` pour un classement dans l'ordre décroissant.

- Une notion essentielle des requêtes SQL est celle de condition qui s'exprime avec le mot-clé `WHERE`. Le schéma de la table résultat est identique à celui de la table sur laquelle porte la sélection.
  - Pour exprimer la condition, on pourra utiliser les opérateurs suivants : `=` (différent du == sur Python), `<>` (et != sur Python), <, >, <=, >= (>=), `AND`, `OR`, `NOT`.
  - Il est possible de comparer des chaînes de caractères directement. La méthode de classement est celle du dictionnaire. Par exemple, `'TOMATE'` > `'MELON'`.

  Sélectionner tous les tuples tels que l'attribut A de la relation $R1$ vérifie la condition c s'écrit :

```
SELECT * FROM R1 WHERE A = c
```

  Exemple : Écrire la requête retournant le numéro de vol de tous les avions décollant le 2 mai 2016 au départ de Charles-de-Gaule.

3.2.2 Union

L'union est un opérateur dédié à des relations de même schéma relationnel. L'union de deux tables $R1$ et $R2$ donne une nouvelle table contenant les tuples de $R1$ et les tuples de $R2$ en éliminant les doublons s'il y en a. L'union de deux relations de même schéma s'écrit :

```
R1 UNION R2
```

Exemple : Écrire la requête retournant le numéro de vol de tous les avions décollant le 2 mai 2016 et le 3 mai 2016 de deux manières différentes.

3.2.3 Intersection

L'intersection est un opérateur dédié à des relations de même schéma relationnel. L'intersection de deux tables $R1$ et $R2$ donne une nouvelle table contenant les tuples appartenant à la fois à $R1$ et à $R2$. L'intersection de deux relations de même schéma s'écrit :

```
R1 INTERSECT R2
```

3.2.4 Différence

La différence est un opérateur dédié à des relations de même schéma relationnel. La différence entre les tables $R1$ et $R2$ donne une table contenant les tuples de $R1$ n'appartenant pas à $R2$. Cela s'écrit :

```
R1 EXCEPT R2
```

3.2.5 Produit cartésien

Le résultat du produit cartésien entre deux tables $R1$ et $R2$ est une relation contenant l'ensemble des possibilités d'association entre une valeur de $R1$ et une valeur de $R2$.

```
R1 CROSS JOIN R2
```

ou plus simplement

```
R1,R2
```

3.2.6 Renommage

Un attribut "A" est renommé en "B". Le schéma du résultat est identique à celui sur lequel porte le renommage. Ce renommage n'est valable que pour la requête dans lequel il est défini, son but est principalement de faciliter la lecture et l'écriture des requêtes complexes. Un renommage s'écrit :

```
SELECT A as B FROM R1
```

3.2.7 Jointure

Une jointure consiste à combiner une paire de tuples de deux relations en un seul tuple. Autrement dit, cela permet de lier (ou de joindre) deux tables. L'écriture d'une jointure où l'attribut A_i de la relation $R1$ devra être égal à l'attribut A_j de la relation $R2$ est :

```
SELECT * FROM R1 JOIN R2 ON R1.Ai = R2.Aj
```

Exemple 1 : Écrire la requête permettant d'obtenir un tableau contenant la liste des numéros de vol ainsi que le nom de la ville de départ.

Exemple 2 : Écrire la requête permettant d'obtenir un tableau contenant la liste des numéros de vol ainsi que le nom de la ville de départ et le nom de la ville d'arrivée.

3.3 Fonctions d'agrégation

Les fonctions d'agrégation dans le langage SQL permettent d'effectuer des opérations statistiques sur un ensemble de tuples. **Ces fonctions d'agrégation ne peuvent s'appliquer que sur une sélection de tuples.**

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

Exemple : Écrire la requête permettant d'obtenir le nombre de vol au départ de Charles-De-Gaule le 2 mai 2016.

3.4 Structure complète d'une requête `SELECT`

De manière générale, une requête s'écrit de la manière suivante :

`SELECT` *< liste d'attributs >*

`FROM` *< liste de table >*

`WHERE` *< condition(s) >*

`GROUP BY` *< liste d'attributs >*

`HAVING` *< condition(s) >*

`ORDER BY` *< liste d'attributs >*

`LIMIT` *< nombre de tuples >*

`OFFSET` *< nombre de tuples >*

Le rôle de certaines clauses (`SELECT`, `FROM`, `WHERE`, `LIMIT`, `OFFSET` et `ORDER BY`) a déjà été défini. Il reste à expliquer l'utilité des mots-clés `GROUP BY` et `HAVING`.

• La clause `GROUP BY` permet de regrouper des n-uplets résultats en groupes.

Exemple : Écrire la requête permettant d'obtenir un tableau affichant les différentes destinations et le nombre de vol au départ de Charles-De-Gaule le 2 mai 2016 pour chacune de ces destinations.

• La clause `HAVING` impose une condition **SUR LES GROUPES préalablement créés avec la clause `Group BY`**.

Exemple : Écrire la requête permettant de renvoyer la(es) destination(s) du tableau précédent pour lesquelles il n'y a qu'un seul vol quotidien.

4 Un peu d'entraînement

4.1 À partir de l'exemple utilisé en cours

Les types SQL `date` et `time` permettent de mémoriser respectivement un jour du calendrier grégorien et une heure du jour. Deux valeurs de type `date` ou de type `time` peuvent être comparées avec les opérateurs habituels (=, <, <=, etc.). La comparaison s'effectue suivant l'ordre chronologique. Ces valeurs peuvent également être comparées à une chaine de caractères correspondant à leur représentation externe (`'aaaa-mm-jj'` ou `'hh:mi'`).

I.A - Écrire une requête SQL qui fournit le nombre de vols qui doivent décoller dans la journée du 2 mai 2016 avant midi.

I.B - Écrire une requête SQL qui fournit la liste des numéros de vols au départ d'un aéroport desservant Paris le 2 mai 2016.

I.C - Que fait la requête suivante ?

```
SELECT id_vol
  FROM vol
    JOIN aeroport AS d ON d.id_aero = depart
    JOIN aeroport AS a ON a.id_aero = arrivee
      WHERE
        d.pays = 'France' AND
        a.pays = 'France' AND
        jour = '2016-05-02'
```

I.D - Certains vols peuvent engendrer des conflits potentiels : c'est par exemple le cas lorsque deux avions suivent un même trajet, en sens inverse, le même jour et à un même niveau. Écrire une requête SQL qui fournit la liste des couples (Id1, Id2) des identifiants des vols dans cette situation. Compléter la requête pour éviter les doublons.

4.2 Réseau routier

On modélise ici un réseau routier par un ensemble de croisements et de voies reliant ces croisements. Les voies partent d'un croisement et arrivent à un autre croisement. Ainsi, pour modéliser une route à double sens, on utilise deux voies circulant en sens opposés.

La base de données du réseau routier est constituée des relations suivantes :

- Croisement(`id`, `longitude`, `latitude`)
- Voie(`id`, `longueur`, `id_croisement_debut`, `id_croisement_fin`)

Dans la suite on considère c l'identifiant (`id`) d'un croisement donné.

Q26 – Ecrire la requête SQL qui renvoie les identifiants des croisements atteignables en utilisant une seule voie à partir du croisement ayant l'identifiant c.

Q27 – Ecrire la requête SQL qui renvoie les longitudes et latitudes des croisements atteignables en utilisant une seule voie, à partir du croisement c.

Q28 – Que renvoie la requête SQL suivante ?

```
SELECT V2.id_croisement_fin
FROM Voie as V1
   JOIN Voie as V2
   ON V1.id_croisement_fin = V2.id_croisement_debut
WHERE V1.id_croisement_debut = c
```

Q29 – Donner les identifiants des 5 croisements donnant accès au plus d'autres croisements.

4.3 Intelligence artificielle - Application en médecine

Une base de données médicale contient des informations administratives sur les patients et des informations médicales. Pour simplifier le problème, on considère deux tables : PATIENT et MEDICAL.

La table PATIENT contient les attributs suivants :

- id : identifiant d'un individu (entier), clé primaire;
- nom : nom du patient (chaîne de caractères);
- prenom : prénom du patient (chaîne de caractères);
- adresse : adresse du patient (chaîne de caractères);
- email : (chaîne de caractères);
- naissance : année de naissance (entier).

La table MEDICAL contient les attributs suivants :

- id : identifiant d'un ensemble de propriétés médicales (entier), clé primaire;
- data1 : donnée (flottant);
- data2 : donnée (flottant);
- ... ;
- idpatient : identifiant du patient représenté par l'attribut id de la table PATIENT (entier);
- etat : description de l'état du patient (chaîne de caractères).

Les attributs data1, data2... sont des données relatives à l'analyse médicale souhaitée (dans notre cas des données biomécaniques). L'attribut « etat » permet d'affecter un label à un ensemble de données médicales : «normal», «hernie discale», «spondylolisthésis».

Q1 : Écrire une requête SQL permettant d'extraire les identifiants des patients ayant une «hernie discale».

Q2 : Écrire une requête SQL permettant d'extraire les noms et prénoms des patients atteints de «spondylolisthésis».

Q3 : Écrire une requête SQL permettant d'extraire chaque état et le nombre de patients pour chaque état.