XV – Requêtes dans une base de données

1/ Utilisation d'un gestionnaire de BDD

On peut se servir de `Python` pour créer et gérer une BDD : on définit tout d'abord les attributs et leurs domaines, puis on saisit les enregistrements un par un. Cette façon de procéder est toutefois un peu austère.

On privilégiera plutôt l'usage de **systèmes de gestion des bases de données** (ou SGBDD) munis d'interfaces graphiques conviviales, qui permettent de rentrer les données sous forme de ***table*** que l'on manipule ensuite à loisir.

Exemple : sur ma page Web, vous pouvez récupérer `SQLiteDataBrower`. C'est un SGBDD libre qui s'utilise sans installation : on peut ainsi le laisser sur une clef USB.

Que ce soit pour créer, modifier ou interroger une BDD, un langage standardisé s'est imposé (et c'est rare en informatique) : le langage `SQL` ou `Structured Query Language`. Mêmes les SGBDD munis d'interfaces évoluées s'en servent, seulement l'utilisateur ne s'en rend pas toujours compte. Il comporte trois instructions fondamentales :

- `CREATE` : créer une table
- `INSERT` : ajouter des données dans la table
- `SELECT` : interroger la table.

On va s'intéresser à la formulation des requêtes, qui commenceront toutes par `SELECT`.

2/ Le langage SQL

2.1) Introduction

Ce langage a été précisément conçu pour traduire les opérations de l'algèbre relationnelle. Il en reprend les opérateurs élémentaires et les structures, en ajoutant des moyens de calculs et d'autres améliorations (comme le tri des résultats).

Définition *La forme générale d'une requête en* `SQL` *est*

```
SELECT ... FROM table ;
```

Remarque : on pourra noter que toutes les requêtes se terminent par un **point-virgule**.

Voyons maintenant la traduction en `SQL` des différentes opérations relationnelles. On reprendra les exemples du chapitre précédent pour illustrer notre propos.

2.2) Opérations spécifiques aux BDD

Soit $\mathcal{R}$ une relation de schéma relationnel $\mathcal{S}$. Dans les exemples, $\mathcal{R}$ = `Eleves2018`.

Projection étonnamment, la commande `SELECT` sert avant tout à effectuer une projection.

Proposition

Soit $X = (A_1, \dots, A_n) \subset \mathcal{S}$. *La projection* $\pi_X(\mathcal{R})$ *s'écrit en* SQL

```
SELECT A1, .., An FROM R ;
```

Il suffit donc d'énumérer les attributs que l'on compte garder en les séparant par des virgules. Si l'on souhaite conserver tous les attributs, on utilise le joker * comme dans

```
SELECT * FROM R ;
```

Remarque : le résultat d'une telle projection n'est pas une vraie relation car les doublons ne sont pas éliminés. Pour remédier à cela, on utilise le mot-clef `DISTINCT`.

Proposition

Soit $X = (A_1, \dots, A_n) \subset \mathcal{S}$. *La « vraie » projection* $\pi_X(\mathcal{R})$ *s'écrit en fait*

```
SELECT DISTINCT A1, ... , An FROM R ;
```

Exemple :
```
SELECT DISTINCT prenom FROM Eleves2018 ;
```

Sélection on impose une condition sur la table à l'aide de l'instruction `WHERE`.

Proposition

Soit $\mathcal{P}$ *un prédicat sur* $\mathcal{R}$. *La sélection* $\sigma_{\mathcal{P}}(\mathcal{R})$ *s'écrit en* SQL

```
SELECT * FROM R WHERE P ;
```

La propriété $\mathcal{P}$ s'écrit exactement comme les propositions logiques en `Python`, à l'aide d'opérateur élémentaires et de connecteurs logiques.

Exemple :
```
SELECT * FROM Eleves2018 WHERE prenom=='Elodie'
                               OR lycee_origine=='Raynade' ;
```

On est amenés à composer les sélections et les projections quand on s'intéresse à certains attributs de certains enregistrements. La commande `SELECT` permettant de réaliser ces deux opérations, les requêtes vont s'écrire de manière très concise.

Exemple :
```
SELECT nom, prenom FROM Eleves2018 WHERE filiere=='BCPST'
                                     AND lycee_origine=='Négal' ;
```

Renommage pour renommer un ou plusieurs attributs, on fait précéder les nouveaux noms du mot-clef `AS` dans une projection.

Proposition

Soient $A_1, \dots, A_n \in \mathcal{S}$. *Le renommage* $\rho_{A_1,\dots,A_n \to B_1,\dots,B_n}$ *s'écrit en* SQL

```
SELECT A1 AS B1, ... , An AS Bn FROM R ;
```

Exemple :
```
SELECT filiere AS section FROM Eleves2018 ;
```

2.3) Opérations sur les attributs

Il est tout à fait possible d'effectuer des opérations sur les attributs lors d'une projection, d'une sélection ou d'un renommage. Elles s'écrivent très naturellement.

Exemple :
```
SELECT note_bac/2 AS note_sur_10 FROM Eleves2018 ;
```

Pour les attributs numériques, on dispose de toutes les opérations usuelles. Pour les attributs de type `str`, voici quelques opérations spécifiques :

- `CHAR_LENGTH(ch)` : longueur de la chaîne `ch`
- `ch1 || ch2` : concaténation des deux chaînes `ch1` et `ch2`
- `LOWER(ch)` : passage en minuscules
- `UPPER(ch)` : passage en majuscules

Exemple :
```
SELECT prenom || ' ' || nom FROM Eleves2018
                            WHERE lycee_origine=='Négal' ;
```

2.4) Opérations ensemblistes usuelles

On les aborde seulement maintenant car il fallait commencer par découvrir les projections et les sélections. Les opérations de base s'écrivent à l'aide de projections complètes des tables et des commandes `UNION`, `INTERSECT` et `EXCEPT`.

Proposition

Soient $\mathcal{R}_1$ *et* $\mathcal{R}_2$ *deux relations suivant le même schéma relationnel* $\mathcal{S}$.

- $\mathcal{R}_1 \cup \mathcal{R}_2$ *s'écrit* `SELECT * FROM` $\mathcal{R}_1$ `UNION SELECT * FROM` $\mathcal{R}_2$ `;`
- $\mathcal{R}_1 \cap \mathcal{R}_2$ *s'écrit* `SELECT * FROM` $\mathcal{R}_1$ `INTERSECT SELECT * FROM` $\mathcal{R}_2$ `;`
- $\mathcal{R}_1 \setminus \mathcal{R}_2$ *s'écrit* `SELECT * FROM` $\mathcal{R}_1$ `EXCEPT SELECT * FROM` $\mathcal{R}_2$ `;`

Exemple :
```
SELECT * FROM BCPST2_2018 INTERSECT SELECT * FROM BCPST2_2017 ;
```

Bien entendu, on pourra aisément combiner ces opérations aux projections, sélections et renommages en enrichissant simplement l'instruction `SELECT`.

2.5) Fonctions supplémentaires

ORDER BY combinée à une expression arithmétique des attributs, cette instruction permet d'ordonner les résultats d'une requête selon les valeurs de l'expression. La syntaxe est

```
SELECT * FROM R ORDER BY expression ;
```

Exemple : `SELECT * FROM Eleves2018 ORDER BY note_bac ;`

On peut ajouter une ou plusieurs expression(s) afin de départager les enregistrements en cas d'égalité. On peut également spécifier si l'on veut trier les résultats dans l'ordre croissant ou décroissant à l'aide des mots-clefs `ASC` (par défaut) et `DESC`.

Exemple : `SELECT * FROM Eleves2018 ORDER BY note_bac DESC ;`

LIMIT lorsque les résultats sont ordonnés, on peut limiter l'affichage aux n premiers via

```
SELECT * FROM R ORDER BY expression LIMIT n ;
```

On ignorera les m premiers résultats avec l'option `OFFSET m`.

LIKE si l'on recherche les enregistrements pour lesquels les valeurs d'un attribut de type `str` suivent un certain modèle, on utilise l'instruction

```
SELECT * FROM R WHERE attribut LIKE modele ;
```

où `modele` est une chaîne de caractères dans laquelle on utilise les jokers `%` pour remplacer un nombre quelconque de caractères et `_` pour remplacer un seul caractère.

Exemple : `SELECT * FROM Eleves2018 WHERE prenom LIKE 'P%' ;`

3/ Traduction des opérations complexes

3.1) Produit et jointure

Soient $\mathcal{R}$ et $\mathcal{R}'$ deux relations de schémas respectifs $\mathcal{S}$ et $\mathcal{S}'$.

Produit cartésien il s'écrit très facilement et revient à sélectionner plusieurs tables.

Proposition

En SQL*, le produit cartésien* $\mathcal{R} \times \mathcal{R}'$ *s'écrit* `SELECT * FROM R,R' ;`

Si l'on combine ce produit à une sélection ou une projection, il faut spécifier la table lorsque des attributs ont le même nom. Ainsi, un attribut A de $\mathcal{R}$ sera noté $\mathcal{R} \cdot$ A.

Exemple : `SELECT prenom, Eleves.nom, Lycees.nom FROM Eleves, Lycees ;`

Pour renommer une table et écourter les requêtes, on fera suivre son nom d'un alias.

Exemple : `SELECT prenom, E.nom, L.nom FROM Eleves E, Lycees L ;`

Jointure on pourrait l'écrire comme une sélection sur un produit

`SELECT * FROM` $\mathcal{R}, \mathcal{R}'$ `WHERE` $\mathscr{C}$ `;`

mais il est plus efficace d'utiliser la syntaxe appropriée.

Proposition

Soit $\mathscr{C}$ *une condition booléenne sur* $\mathcal{S} \cup \mathcal{S}'$. *La jointure* $\mathcal{R} \underset{\mathscr{C}}{\bowtie} \mathcal{R}'$ *s'écrit en* SQL

`SELECT * FROM` $\mathcal{R}$ `JOIN` $\mathcal{R}'$ `ON` $\mathscr{C}$ `;`

Typiquement, pour une jointure symétrique, on met dans `ON` la condition d'égalité des attributs et dans `WHERE` les autres conditions de sélection.

Exemple : `SELECT * FROM Eleves E JOIN Lycees L ON E.id_L==L.id_L;`

3.2) Agrégats

Soit $\mathcal{R}$ une relation de schéma relationnel $\mathcal{S}$.

Agrégation simple c'est juste une opération sur un attribut.

Proposition

Soient $A \in \mathcal{S}$ *et* f *une fonction d'agrégation sur* dom(A). La requête $\gamma_{f(A)}(\mathcal{R})$ s'écrit

`SELECT f(A) FROM` $\mathcal{R}$ `;`

De base, SQL dispose des fonctions d'agrégation suivantes :

COUNT	MAX	MIN	SUM	AVG
cardinal	maximum	minimum	somme	moyenne

Exemples : `SELECT AVG(note_bac) FROM Eleves ;`

Agrégation complexe rajoutons le regroupement selon les valeurs d'un attribut ou plus.

Proposition

Soient $X \subset \mathcal{S}$, $A \in \mathcal{S} \setminus X$ *et* f *une fonction d'agrégation. La requête* ${}_X\gamma_{f(A)}(\mathcal{R})$ *s'écrit*

`SELECT X,f(A) FROM` $\mathcal{R}$ `GROUP BY X;`

Exemple :
```
SELECT lycee_origine, AVG(note_bac) FROM Eleves2018
                                    GROUP BY lycee_origine ;
```

Agrégation et sélection on a vu que l'on peut combiner agrégation et sélection, en effectuant celle-ci en amont ou en aval.

- Les sélections en amont – qui sont à privilégier – se font à l'aide d'une clause `WHERE` placée **avant** le `GROUP BY`.
- Les sélections en aval – incontournables dès qu'elles utilisent les résultats des fonctions d'agrégation – se font avec une clause `HAVING` placée **après** le `GROUP BY`.

Proposition

Soient $X \subset \mathcal{S}$, $A \in \mathcal{S} \setminus X$, $\mathscr{P}$ *un prédicat sur* $\mathcal{R}$ *et* f *une fonction d'agrégation sur* dom(A).

- ${}_X\mathcal{Y}_{f(A)}(\sigma_{\mathcal{P}}(\mathcal{R}))$ s'écrit `SELECT X,f(A) FROM` $\mathcal{R}$ `WHERE` $\mathcal{P}$ `GROUP BY X ;`
- $\sigma_{\mathcal{P}}\left({}_X\mathcal{Y}_{f(A)}(\mathcal{R})\right)$ s'écrit `SELECT X,f(A) FROM` $\mathcal{R}$ `GROUP BY X HAVING` $\mathcal{P}$ `;`

Exemples :

- ```
  SELECT lycee_origine, AVG(note_bac) FROM Eleves2018
         GROUP BY lycee_origine HAVING AVG(note_bac)>=12 ;
  ```
- ```
  SELECT lycee_origine, AVG(note_bac) FROM Eleves2018
                   WHERE numero==1 GROUP BY lycee_origine ;
  ```

Exercices

Exercice 1.
Créer des relations illustrant les schémas proposés à l'exercice 2 du chapitre précédent.
Essayer des requêtes amusantes.

Exercice 2.
Traduire en langage SQL les requêtes de l'exercice 4 du chapitre précédent.
La relation à utiliser est contenue dans la base de données `cpge.sqlite`.

Exercice 3.
Traduire en langage SQL les requêtes de l'exercice 5 du chapitre précédent.
Les relations sont contenues dans la base de données `hotel.sqlite`.

Les exercices suivants utilisent la table des communes de France `communes`*, qui est contenue dans la base de données* `geographie.sqlite`*. Elle contient les attributs suivants :*

- `num-departement` *: numéro du département*
- `nom` *: nom de la commune*
- `canton` *: numéro du canton de la commune*
- `population-2010` *: nombre d'habitants lors du recensement de 2010*
- `population-1999` *: nombre d'habitants lors du recensement de 1999*
- `surface` *: superficie de la commune en* km^2

On dispose aussi de tables des départements et des régions de France, à savoir `departements` *de schéma* `(num_departement,num_region,nom)` *et* `regions` *de schéma* `(num_region,nom)`.

Exercice 4.
Combien y a-t-il :

1) d'habitants en France ?
2) de communes en France (avec ou sans noms distincts) ?
3) de communes en Bretagne ?

Exercice 5.
Obtenir la liste des 20 communes :

1) les plus petites de France (en terme de surface) ;
2) les plus peuplées de France ;
3) les plus densément peuplées d'Ille-et-Vilaine.

Exercice 6.
Combien de communes ont un nom :

1) contenant la lettre x ?
2) contenant les six voyelles a, e, i, o, u et y ?

Dans ce dernier cas, y en a-t-il une dans le Morbihan ?

Exercice 7.
Quelles sont les cinq communes de Loire-Atlantique où la population a le plus augmenté entre 1999 et 2010 ?
On considérera les augmentations absolue puis relative (en pourcentage).

Exercice 8.
Obtenir la liste des départements et, pour chacun :

1) le nombre de communes ;
2) la population totale ;
3) la superficie totale ;
4) la densité de population.

On rangera les listes dans l'ordre décroissant.

Exercice 9.

1) Donner la liste des départements, région par par région.
2) Donner la liste des départements contenant une ville de plus de 200 000 habitants.

Exercice 10.
Obtenir la liste des régions et, pour chacune :

1) le nombre de départements ;
2) la population totale ;
3) la superficie totale ;
4) la densité de population.

On rangera les listes dans l'ordre décroissant.