COURS-TP : Sql 1

I] Introduction et vocabulaire

Une base de données est un ensemble de *tables* (aussi appelées *relations* dans ce contexte), une table pouvant être représentée comme un tableau à deux dimensions. Pour ce premier TP, nous allons utiliser des données fournies par l'INSEE, résultat des recensements (ce fichier est relativement simple mais très long : plus de 35000 lignes).

Voici un extrait (il y a plus de colonnes et beaucoup plus de lignes) de la table qu'on va utiliser :

code	dept	nom	pop2010	pop1999	surf
54395	54	Nancy	105421	103552	15
59183	59	Dunkerque	92005	70834	44
59350	59	Lille	227560	184647	35

Les en-têtes des *colonnes* sont aussi appelés les *attributs*, les *lignes* sont appelés *enregistrement*, par exemple `code` est un attribut (il correspond au code INSEE de la commune, il commence par le numéro du département puis le numéro de la commune, il est utilisé, par exemple, dans le numéro de sécurité social). Les attributs ont un *type*, ce qu'on appelle aussi *domaine*, on peut avoir des entiers (comme `code`, `dept`, etc.), des flottants, des chaînes (comme `nom`), etc.

Chaque enregistrement d'une table doit pouvoir être identifiée de manière unique par un (ou plusieurs) attribut, ce choix s'appelle *clef primaire*, dans notre table c'est l'attribut `code` la clef primaire. Il n'aurait pas été possible de choisir `nom`, en effet il existe plusieurs communes qui ont le même nom (par exemple il existe 11 communes nommées Beaulieu), comme elles sont dans des départements différents le couple `(dept,nom)` aurait pu être choisi comme clef primaire.

En général, s'il n'y a pas de clef primaire "naturelle" (comme `code` ici) on rajoute un attribut, souvent appelé `id`, qui numérote les enregistrements.

Récupérer les fichiers `communes.sqlite` (qu'il faut copier sur son espace personnel), il sera peut-être aussi nécessaire de récupérer et d'installer le logiciel DB Browser for SQLite.

II] DB Browser for SQLite

En général l'intérêt d'une base de données est de pouvoir être utilisée dans un réseau : la mise en place est assez lourde mais le partage d'une même base par plusieurs utilisateurs est un gros avantage. Une base peut aussi être aussi utilisée dans un cadre logiciel (historique de navigation, high score, etc.).

Nous allons utiliser un logiciel plus léger pour l'apprentissage du langage SQL : chaque poste exploitera sa base de données. Le logiciel choisi est DB Browser for SQLite.

Lancer le logiciel (il est en anglais).

Ouvrir la base de données `communes.sqlite` depuis le menu `Files`, item `Open Database`.

En cliquant sur le signe + à côté du nom de la table utilisée (communes ici) on voit apparaître le nom des différentes colonnes.

Selon les versions il peut y avoir d'autres onglets que ceux indiqués ici mais les seuls que nous utiliserons sont les 3 indiqués ici :

- L'onglet par défaut, `Database Structure`, permet de voir l'agencement des différentes tables. Notre premier TP se contente d'une seule table. C'est ici que l'on peut s'assurer du nom des variables.
- L'onglet `Browse Data` permet de voir toutes les données comme dans un tableur : en général il n'est pas utilisable car les tables auront une grande taille.
- L'onglet `Execute SQL` est l'endroit où nous allons travailler.
  On entre la commande dans la fenêtre `SQL` ou `SQL String` et on la fait exécuter en cliquant sur la fleche (ou sur `Execute Query`). On peut créer des onglets pour garder sous la main les anciennes requêtes.

III] Extraction

Le premier usage du langage de requêtes est d'extraire les informations qui nous intéressent.

La structure de base est :

```
SELECT nom, pop2010 /* intitulés des colonnes demandées */
FROM communes       /* tables utilisées */
WHERE pop2010 > 200000 /* Conditions qui filtrent les lignes */
```

Cette requête affiche le nom et la population en 2010 des villes de plus de 200.000 habitants. Si on veut afficher tous les attributs on peut écrire `SELECT *`.
l'opérateur `SELECT` permet de choisir les attributs qu'on veut afficher (on réalise une projection), l'opérateur `WHERE` permet de n'afficher que les entités qui vérifient une certaine condition (on réalise une sélection). Les conditions seront souvent basés sur des comparaisons, les opérateurs sont : `=`, `<>` (différent), `<`, `>`, `<=`, `>=`.

Sauf indication, on s'intéressera à la population en 2010.

Q1. a) Trouver tous les renseignements d'une ville que vous choisissez (le nom d'une ville étant une chaîne de caractère on n'oubliera pas d'encadrer le nom par des guillemets).
Remarque : Il se peut que plusieurs communes portent le même nom (par exemple Beaulieu), on peut demander de n'afficher que des lignes différentes avec `SELECT DISTINCT`, fait des essais (avec juste `SELECT nom`, avec `SELECT DISTINCT nom` et avec `SELECT DISTINCT nom, pop2010`).

b) Trouver les noms des communes de moins de 10 habitants avec leur département et leur population.

c) Si les valeurs sont des chaînes de caractères on peut filtrer sur une partie du nom :

i. `_` représente un caractère quelconque,

ii. `%` représente une suite (éventuellement vide) de caractères.

iii. L'opérateur s'écrit alors `LIKE` et non `=`.

Trouver les communes dont le nom contient `kerque`.

d) On peut mettre plusieurs conditions, pour cela on compose les recherches avec`OR` (ou), `AND`(et), ou `NOT` (négation).
Trouver les noms des communes de plus de 30000 habitants et de surface inférieure à 5 km^2 avec leur département, leur population et leur surface.

e) Les tests peuvent contenir des calculs. Attention une division entre deux entiers correspond en général à une division entière, on pourra rajouter 0.0 au numérateur pour le forcer à être un flottant).
Trouver les noms des communes dont la densité de population est supérieure à 20000 habitants par km^2 avec leur département, leur population et leur surface.

f) Les résultats montrés (ie dans le `SELECT`) peuvent aussi contenir des calculs et on peut nommer le résultat d'un calcul avec le mot-clé `AS` (par exemple `SELECT nom, pop2010-pop1968 AS deltapop`) pour le réutiliser (ou lui donner plus de clarté).
Les valeurs affichées peuvent présenter un nombre déterminé de décimales avec `round(x,k)` qui affiche x avec k décimales [1].
Trouver les noms des communes dont la densité de population est inférieure à 1 habitant par km^2 avec leur département, leur population, leur surface et leur densité (arrondie à 2 chiffres après la virgule).

g) On peut améliorer la présentation en triant les résultats. Pour cela on ajoute une ligne `ORDER BY variable` où `variable` est le nom de l'attribut ou de la variable calculée qui sert de clé de tri. Les résultats sont affichés selon l'ordre croissant, si on veut l'ordre inverse on suit la commande par `DESC` (`ASC` pour l'ordre croissant).
Trouver les noms des communes avec leur département et leur population dans l'ordre décroissant de population.

h) On peut limiter le nombre de résultats aux `n` premiers avec une ligne supplémentaire : `LIMIT n`.
Trouver les 10 communes dont le taux d'augmentation de population a été le plus important entre 1999 et 2010. On pourra se restreindre aux villes de plus de 100.000 habitants.
Dans le même ordre d'idée il existe aussi la commande `OFFSET m` qui permet de n'afficher les résultats sans les `m` premiers (ie à partir du `m+1`-ième), elle s'utilise conjointement avec `LIMIT` : `LIMIT 10 OFFSET 3`.

i) Le langage `SQL` possède pas mal de fonctions, par exemple, si `vartxt` est une variable (ou un attribut) contenant une chaine de caractère, on peut avoir le nombre de caractères avec `length(vartxt)`, ce type de commande qui ne sont pas au programme seront (a priori) documenté dans les sujets de concours.
Quelle commune a le nom le plus long? Le plus court ?

IV] Regroupements (agrégats)

`SQL` possède un certain nombre de fonctions statistiques (fonctions d'agrégation) qui par défaut s'applique à l'ensemble des éléments de la requête. Les fonctions au programme sont

— `MIN(at)` : valeur minimale de l'attribut `at`
— `MAX(at)` : valeur maximale de l'attribut `at`
— `SUM(at)` : somme de l'attribut `at`
— `AVG(at)` : moyenne de l'attribut `at`
— `COUNT()` : nombre de lignes

Par exemple, si on veut le nombre moyen d'habitant dans une commune en France :

1. cette commande `round` n'est pas exigible

```
SELECT AVG(pop2010)
FROM communes
```

Q2. Combien y-a-t-il de communes dans la base ?

Q3. Combien y-a-t-il d'habitant dans le département du nord ?

Q4. Et dans les Hauts-de-France les cinq départements des Hauts-de-France sont : Aisne (02), Nord (59), Oise (60), Pas-de-Calais (62) et Somme (80)) ?
Indication : après avoir écrit la requête on la récrira en utilisant `WHERE dept IN (2,59,60,62,80)`.

Il est aussi possible d'appliquer des fonctions d'agrégations sur des groupes d'éléments, pour cela on utilise le mot-clef `GROUP BY`

Par exemple si on veut les trois départements qui ont le plus grand nombre de communes :

```
SELECT dept, count() AS nbrCom
FROM communes
GROUP BY dept
ORDER BY nbrCom DESC
LIMIT 3
```

Attention, si on regroupe la table par rapport à la colonne `dept`, la colonne `nom` n'a plus grand sens, il s'avère que (`SQL` affichera quand même quelque chose (le nom d'une des commune du département).

Q5. a) Quels sont les 3 départements les plus peuplés ?

b) Quels est le nom de commune le plus courant ?

c) On peut avoir besoin de filtrer selon un résultat du regroupement. Pour cela l'instruction de test s'écrit dans une ligne commençant par `HAVING` située en dessous de la ligne `GROUP BY`.

Quels sont les départements dont la population est supérieure à 1.000.000 ?

d) Quels sont les départements dont la population de la ville la plus peuplée est au moins un tiers de la population totale ?

V] Requêtes imbriquées

Si le résultat d'une requête est un tableau on peut l'utiliser pour une autre requête.

Si le résultat d'une requête est un un nombre (tableau à une ligne et une colonne) on peut l'utiliser dans un test ou un calcul.

Q6. a) Après avoir calculé la population par département, déterminer la moyenne de la population par département.

b) Déterminer les départements qui ont une population inférieure au quart de la moyenne.

c) Déterminer le nombre de personnes habitant dans une ville comportant 1 à 9 habitants, 10 à 99 habitants, ...(pour cela la fonction `length(n)` qui renvoie le nombre de chiffres de n est utile)

VI] Comment faire ?

On peut aussi vouloir obtenir la liste des communes les plus peuplées de chaque département, pour cette requête il serait bon d'avoir dans notre table la colonne de la population de la ville la plus peuplée, pour cela on a besoin de la notion de jointure qui sera étudiée pendant la prochaine séance (sur une autre base de données), voici une requête possible :

```
SELECT communes.dept, communes.nom
FROM (SELECT dept, MAX(pop2010) AS popMaxA
          FROM communes
          GROUP BY dept) AS maxdept JOIN communes ON maxdept.dept=communes.dept
WHERE maxdept.popMaxA=communes.pop2010
```