12.1 Requêtes SQL

12.1.1 Bases de données relationnelles

Comment organiser et interagir avec des giganstesques quantité de données ? Cette question est à l'origine d'un champ crucial de l'informatique : les *systèmes de gestion de bases de données* (SGBD). Un SGBD est un logiciel dont la charge est précisément de stocker, organiser et accéder à un ensemble de données. Leur fonctionnement est extrêmement complexe, nous nous contenterons d'apprendre à *interagir* avec ces systèmes, par le biais du langage SQL (pour *Structured Query Language*).

Plus précisément, nous nous intéressons au bases de données dites *relationnelles* (BDR). Schématiquement, les bases de données relationnelles sont composées de plusieurs *tables* (sorte d'équivalent des tableaux bidimensionnels d'Excel), et de *relations* entre ces tables. Par exemple, dans la base de données MONDIAL (voir le site https ://www.dbis.informatik.uni-goettingen.de/Mondial/), qui contient pléthore d'informations géographiques et socio-économiques à l'échelle mondiale, on trouve une table `country` qui contient 6 *attributs*, voir tableau 12.2

`Name`	`Code`⋆	`Capital`	`Province`	`Area`	`Population`
				Surface du pays en km^2	

TABLE 12.1 – la table `country`

Un attribut est simplement le nom donné à une des colonnes de la table. Un *enregistrement*, ou *entrée*, est une ligne de la table. Par exemple, voici les 3 premiers enregistrements de la table `country`

`Name`	`Code`⋆	`Capital`	`Province`	`Area`	`Population`
Albania	AL	Tirana	Albania	28750	281977
Greece	GR	Athina	Attixis	131940	10432481
Cyprus	CY	Nicosia	Cyprus	9251	918100

TABLE 12.2 – 3 premiers enregistrements de la table `country`

En général, une table contient un grand nombre d'enregistrements et l'utilisateurs connaît uniquement le nom des attributs de la table (il y a trop d'enregistrements pour tous les regarder individuellement).

Clefs primaires

Que veut dire le symbole ⋆ à côté de l'attrubut `Code` ? C'est une indication pour signifier que l'attribut `Code` forme la clef primaire de la table `country`, c'est-à-dire que chaque enregistrement de la table possède une valeur pour `Code` distincte (il n'y a pas de doublons). Ainsi, la valeur de `Code` permet d'identifier *uniquement* un enregistrement de la table ; par exemple GR identifie uniquement la seconde entrée de `country`.

En général, la clef primaire est constituée d'un seul attribut – mais ce n'est pas obligatoire. Par exemple, il existe une table `encompasses`, toujours dans la base de données MONDIAL, dont la clef primaire est le couple d'attributs `(Country, Continent)` – voir tableau 12.3. La table `encompasses` indique pour chaque pays les continents dans lequel se trouve le pays et la pourcentage de la surface du pays dans chacun de ces coninents. L'attribut `country` seul n'est pas une clef primaire car il est possible qu'un pays soit à cheval sur plusieurs continents (par exemple la Russie, représentée par un "R" dans la table).

`Country`⋆ Clef étrangère pour `country.Code`	`Continent`⋆	`Percentage` Pourcentage de la surface du pays dans le continent
R	Europe	23.15
R	Asia	76.85
RA	South America	100
RB	Africa	100

TABLE 12.3 – 4 premières entrées de la table `encompasses`

Clef étrangère

Dans la table `encompasses`, vous aurez peut être remarqué que `Country` est une "clef étrangère pour `country.Code`". Qu'est-ce que cela signifie ?

Tout l'intérêt des bases de données relationnelles est de considérer les tables non pas individuellement mais comme un ensemble lié par des relations. Une clef étrangère permet de mettre en correspondance les lignes d'une table avec les lignes d'une autre table, en mettant en relation la clef étrangère de la première avec la clef primaire de la seconde. En l'occurrence, cela signifie simplement qu'un enregistrement de la table `encompasses` dont l'attribut `Country` est, par exemple, FR évoque le même pays que l'enregistrement de `country` dont l'attribut `Code` est FR.

Cela nous permettra par exemple de créer une grande table rassemblant à la fois les informations de `Country` et `Code`, appelée une *jointure*. Une image vallant mille mots, regarder figure 12.1. Une telle table pourra être utile pour par exemple répondre à la question "Quelle est la surface totale d'Europe ?".

Table e

Continent	Percentage	Country
Asia	76.85	R
Europe	23.15	R
South America	100	RA
Africa	100	RB
Asia	100	RC

Table c

Name	Code	Capital	Province	Area	Population
Russia	R	Moskva	Moscow	17075200	144699673
Argentina	RA	Buenos Aires	Distrito Federal	2766890	42669500
Botswana	RB	Gaborone	Bostwana	600370	2346179
Taiwan	RC	Taipei	Taiwan	36179	23571227

Table Join(e,c)

e.Country = c.Code	e.Continent	e.Percentage	c.Name	c.Capital	c.Province	c.Area	c.Population
R	Asia	76.85	Russia	Moskva	Moscow	17075200	144699673
R	Europe	23.15	Russia	Moskva	Moscow	17075200	144699673
RA	South America	100	Argentina	Buenos Aires	Distrito Federal	2766890	42669500
RB	Africa	100	Botswana	Gaborone	Bostwana	600370	2346179
RC	Asia	100	Taiwan	Taipei	Taiwan	36179	23571227

FIGURE 12.1 – Jointure de `encompasses` et `country` sur `encompasses.Country` = `country.Code`

12.1.2 Le language SQL

Comme mentionné avant, le language SQL permet d'interagir avec le système de base de données : manipuler des tables, extraire des enregistrements, sélectionner certains attributs, faire des jointures, etc. On opte ici par une présentation du language par l'exemple, en augmentant petit à petit le niveau de difficulté.

Pour tester ses requêtes en ligne

Il pourra être utile de tester les requêtes réellement sur la base de données MONDIAL. Pour cela, suivre les instructions suivantes :

(i) aller sur la page https://sqliteonline.com/.

(ii) cliquer sur `Import` dans la barre d'outils du haut et importer le fichier de base de données que vous pouvez télécharger à l'adresse https://gabriel.belouze.com/tps/data/bdd-mondial.sql, qui crée le sous-ensemble de MONDIAL avec lequelle on va travailler.

(iii) cliquer sur OK. Vous devriez voir apparaître dans la barre de gauche les tables `Country`, `City`, `Economy`, `Encompasses`, `Spoken`.

Ensuite, il suffit d'écrire les requêtes dans l'éditeur de texte et de les tester avec le bouton `Run`.

Premiers pas

Regardons juste l'attribut `Name` de la table `country`.

```
SELECT Name FROM country;
```

Remarquez que les requêtes SQL se terminent par un point virgule. Remarquez également que SQL ne fait pas la différence entre minuscules et majuscules, on aurait tout aussi bien pu écrire

```
SeleCt NAMe froM COUNTry;
```

Traditionnellement, les mots clefs (ici `SELECT` et `FROM`) sont écrits tout en majuscules.
Regardons les attributs `Area` et `Population` de la table `country`

```
SELECT Area, Population FROM Country;
```

Remarque : contrairement à `Python`, les espaces, retours à la ligne et indentations ne sont pas importants en SQL. On aurait tout aussi bien pu écrire

```
SELECT
                Area,
            Population              FROM

    Country
;
```

Mais évidemment, une bonne indentation clarifiera la lecture et relecture de vos requêtes.

1. a) Écrire une requête pour obtenir les pays du monde et leurs capitales.

b) Écrire une requête pour obtenir toute la table `country`

Lorsque l'on veut récupérer l'intégralité d'une table, il devient pénible de devoir écrire le nom de tous les attributs. SQL permet de mettre le symbole `*` à la place. Ainsi, votre dernière requête aurait pu s'écrire

```
SELECT * FROM Country;
```

On peut supprimer les doublons avec le mot clef `DISTINCT`. Par exemple, on obtient tous les continents avec la requête

```
SELECT DISTINCT continent FROM encompasses;
```

Conditions La requête suivante répond à la question : quels sont les pays de plus de 100 000 000 habitants ?

```
SELECT Name FROM Country WHERE Population > 100000000;
```

On peut créer des conditions plus complexes grâce aux opérateurs `OR` et `AND`, par exemple voici une requête pour trouver les petits pays (de surface inférieure à 50 000 km^2) mais à grande densité de population (plus que 600 habitants par km^2).

```
SELECT Name
    FROM Country
    WHERE Area < 50000 AND Population / Area > 600;
```

Tri du résultat On peut aussi obtenir la même liste mais triée dans l'ordre croissant

```
SELECT Name
    FROM Country
    WHERE Population > 100000000
    ORDER BY Name ASC;
```

Ou dans l'ordre décroissant

```
SELECT Name
    FROM Country
    WHERE Population > 100000000
    ORDER BY Name DESC;
```

Tronquer le résultat On peut limiter le nombre de résultat avec la syntaxe `LIMIT n`, où `n` est un entier. On peut aussi ignorer les `m` premiers enregistrements du résultat avec la syntaxe `OFFSET m`. Par exemple, la requête suivante trouve les 5 premiers pays (pour l'ordre alphabétique) dont la population dépasse 100M

```
SELECT Name
    FROM Country
    WHERE Population > 100000000
    ORDER BY Name ASC
    LIMIT 5;
```

et les 3 suivants

```
SELECT Name
    FROM Country
    WHERE Population > 100000000
    ORDER BY Name ASC
    LIMIT 3
    OFFSET 5;
```

2. Rédiger une requête pour obtenir

a) les 3 pays les plus peuplés du monde, avec leur population correspondante.

b) le 4e et 5e pays le moins peuplé du monde.

Jointures

Jusque là on n'a pu considérer les tables qu'une par une. Comment faire en SQL l'équivalent de ce qui est représenté figure 12.1 ? Voici par exemple une requête pour obtenir les pays d'Europe

```
SELECT country.Name
    FROM country JOIN encompasses ON country.Code = encompasses.Country
    WHERE encompasses.Continent = "Europe";
```

La partie `country JOIN encompasses ON country.Code = encompasses.Country` crée une table similaire à celle représentée figure 12.1, c'est-à-dire dont les attributs sont

country.Name	country.Code	country.Capital	country.Province	country.Area
country.Population	encompasses.Country	encompasses.Continent	encompasses.Percentage	

et dans laquelle pour chaque enregistrement *country.Code* et *encompasses.Country* sont égaux.

Pour éviter de devoir écrire le nom complet de la table d'origine, on peut fournir des noms plus courts avec le mot clef `AS`. Ainsi la requête précédente aurait pu s'écrire

```
SELECT c.Name
    FROM country AS c JOIN encompasses AS e ON c.Code = e.Country
    WHERE e.Continent = "Europe";
```

3. Rédiger une requête pour obtenir

a) les pays du continent américain qui comptent moins de 10 habitants par km^2.

b) les capitales européennes situées à une latitude supérieure à 60°.

c) les pays ayant plus de 1 000 000 km^2 dans l'Asie, triés par surface de territoire asiatique décroissante. Donner également la valeur de cette surface de territoire asiatique.

Fonctions d'agrégation

SQL donne également accès à des fonctions qui réalisent des calculs sur l'ensemble d'une table, les fonctions d'agrégation. figure 12.2 rassemble les plus importantes de ces fonctions (et celles qqui sont au programme).

`COUNT()`	nombre d'enregistrements
`MAX()`	valeur maximale d'un attribut
`MIN()`	valeur minimale d'un attribut
`SUM()`	somme d'un attribut
`AVG()`	moyenne d'un attribut

FIGURE 12.2 – Fonctions d'agrégation

Ainsi la requête suivante calcule le nombre de pays en Asie

```
SELECT COUNT(Country) FROM Encompasses WHERE Continent="Asia";
```

Puisqu'ici on compte juste le nombre de lignes, la colonne que l'on donne à l'intérieur de `COUNT(...)` n'importe pas. Il est donc plus élégant d'écrire

```
SELECT COUNT(*) FROM Encompasses WHERE Continent="Asia";
```

4. Rédiger une requête pour obtenir

a) la surface totale du continent `Africa`.

b) la population mondiale.

c) la population moyenne par pays.

d) le nombre de pays de plus de 1 000 000 de km^2