Bases de données relationnelles

Une base de données est un ensemble de *tables* ou *relations*, qu'on représente comme des tableaux à deux dimensions.

Chaque table contient des *attributs*, qu'on peut voir comme les colonnes du tableau correspondant ; à chaque attribut est alors associé un *domaine*, c'est-à-dire le type des objets de cet attribut (des entiers, des chaînes de caractères, *etc.*).

Les lignes des tableaux sont appelés *enregistrements* ; chaque enregistrement d'une table peut alors être vu comme un n-uplet, chaque élément étant correspondant à un attribut (et devant donc être du domaine correspondant).

Les tables contiennent une clef particulière, appelée *clef primaire*, qui est un attribut permettant d'identifier de façon unique chaque enregistrement. Les autres attributs sont appelés *clefs secondaires*. On note que parfois, une table peut contenir un attribut qui est la clef primaire d'une autre table ; on l'appelle alors *clef secondaire*.

Pour récupérer les enregistrements d'une table, on utilisera la commande `SELECT` *attribut1, attribut2, ...* `FROM` *table*. Le token `*` permet de récupérer tous les attributs d'une table.

Dans ce TP, on utilisera la base de donnée `mondial`, avec laquelle on peut interagir à l'adresse suivante : https://www.semwebtech.org/sqlfrontend/.

Exercice 1

Donner les attributs de la table `country`.

Écrire une commande permettant de ne récupérer que les pays avec leur code.

```
SELECT * FROM country
SELECT name, code FROM country
```

On peut aussi filtrer les résultats selon certains critères avec le mot-clef `WHERE`. On peut alors utiliser les opérations usuelles (+, -, *, /), des opérateurs de comparaison (>,<,>=,<=,=,<>) et des connecteurs logiques (`AND`, `OR`, `NOT`).

Exercice 2

Afficher la liste des noms de pays dont la population est inférieure à 1000.

```
SELECT name FROM country
WHERE population < 1000
```

Si besoin, on peut trier les résultats d'une requête avec le mot-clef `ORDER BY` *attribut* `ASC/DESC`, selon si on veut un ordre croissant ou décroissant.

Exercice 3

Afficher la liste des noms de pays par ordre décroissant de superficie.

```
SELECT name FROM country
ORDER BY area DESC
```

Avec plusieurs tables de la même base qui ont un attribut commun, on peut fusionner des tables sur cet attribut, en créant une nouvelle table (virtuelle), avec la commande

table1 `JOIN` *table2* `ON` *table1.attribut1* `=` *table2.attribut2*.

On note que dès qu'on utilise plusieurs tables, il faut préfixer chaque attribut par le nom de la table correspondante, sous la forme *table.attribut*.

Exercice 4

La base de donnée contient une table `spoken`. Quel attribut de cette table est commun avec un attribut de la table `country` ?

Afficher alors tous les attributs de la jointure de ces deux tables.

Afficher ensuite la liste des langues parlées en France.

```
SELECT * FROM
city JOIN country ON city.country = country.code

SELECT * FROM
country JOIN spoken ON country.code=spoken.country
WHERE country.name='France'
```

Exercice 5

En utilisant la table `encompasses` contenant des informations géographiques sur les continents, afficher la liste des pays d'Afrique, classés par ordre croissant de population.

Afficher la liste des pays qui sont sur plusieurs continents.

```
SELECT country.name FROM
encompasses JOIN country ON encompasses.country=country.code
WHERE encompasses.continent='Africa'
ORDER BY country.population

SELECT country.name FROM
encompasses JOIN country ON encompasses.country=country.code
WHERE encompasses.percentage < 100
```

Dans ce dernier exemple, on voit que les pays apparaissent plusieurs fois chacun (en fait, une fois par continent). Pour éviter ça, on peut rajouter le mot-clef `DISTINCT` après `SELECT` pour enlever les doublons.

La dernière opération qui nous intéresse est la fonction d'agrégation : on peut regrouper les résultats d'une requête pour lui appliquer une opération ; les opérations disponibles sont `COUNT, MAX, MIN, SUM, AVG`.

Exercice 6

Que renvoie la commande suivante ?

```
SELECT continent, COUNT(*) AS nombre
FROM encompasses
GROUP BY continent
```

Prévoir ce que renvoie la commande suivante, avant de la tester :

```
SELECT encompasses.continent, SUM(country.population) AS Pop FROM
country c JOIN encompasses e ON encompasses.country=country.code
GROUP BY encompasses.continent
```

Après une agrégation, on peut utiliser le mot-clef `HAVING` pour filtrer les résultats.

Exercice 7

Modifier la commande précédente pour n'afficher que les continents dont la population est supérieure à un milliard.

```
SELECT encompasses.continent, SUM(country.population) AS Pop FROM
country c JOIN encompasses e ON encompasses.country=country.code
GROUP BY encompasses.continent
HAVING SUM(country.population) > POWER(10,9)
```

Exercice 8

Écrire une commande permettant de donner la liste des langues qui ne sont parlées que dans un seul pays.

```
SELECT language FROM spoken
GROUP BY language
HAVING COUNT(country)=1
```

Exercice 9

Écrire une commande permettant d'afficher la liste des langues parlées par moins de 1000 personnes dans le monde.

```
SELECT spoken.language
FROM spoken JOIN country ON spoken.country = country.code
GROUP BY spoken.language
HAVING SUM ( country.population*spoken.percentage*0.01) < 5000
```

Exercice 10

Écrire une commande permettant d'afficher la liste langues parlées en Amérique du Sud, ainsi que le nombre de personnes les parlant, dans l'ordre décroissant.

```
SELECT spoken.language, SUM(country.population*spoken.percentage*0.01) Nb FROM
spoken
JOIN country ON spoken.country=country.code
JOIN encompasses ON encompasses.country = country.code
WHERE encompasses.continent='South America'
GROUP BY spoken.language
ORDER BY Nb DESC
```

Exercice 11

Donner le nombre de rivières françaises présentes dans la table `geo_river`.

```
SELECT COUNT (DISTINCT geo_river.river)
FROM geo_river JOIN country ON geo_river.country = country.code
WHERE country.name = 'France'
```

Exercice 12

Afficher le nom des trois plus hautes montagnes africaines.

On pourra utiliser la table `geo_mountain`.

```
SELECT m.name , m.elevation FROM mountain m
JOIN geo_mountain gm ON m. name = gm . mountain
JOIN country c ON gm.country = c.code
JOIN encompasses e ON c.code = e.country
WHERE e.continent = 'Africa'
ORDER BY m.elevation DESC
FETCH FIRST 3 ROWS ONLY
```

Exercice 13

Donner la liste des pays d'Amérique avec la hauteur de leur plus haute montagne.

On pourra utiliser la table `mountain`.

```
SELECT c.name, MAX(m.elevation)
FROM geo_mountain gm JOIN mountain m ON gm.mountain = m.name
JOIN country c ON gm.country = c.code
JOIN encompasses e ON c.code = e.country
WHERE e.continent = 'South America' OR e.continent = 'North America'
GROUP BY c.name
```