FACULTÉ DES SCIENCES ET TECHNOLOGIES
Département Informatique

Université de Lille

Formations en Informatique de Lille – 2022/2023

Bases de Données (BDD1)

septembre 2022

TDM n°3

Exercice 1 :

Commencez par installer les tables fournies en exécutant le fichier SQL `base_mel.sql`

On s'intéresse tout d'abord à deux des tables : la table **communes** rassemble des informations générales et intemporelles sur les communes de la Métropole Européenne de Lille (dans son périmètre d'avant 2017). La table population contient des données de recensement de population de plusieurs années. Consultez dans DBeaver la composition de ces 2 tables.

NB : les tables nécessaires à cet exercice sont situées dans un schéma PostgreSQL nommé `mel` . Vous devez veiller à ce que votre éditeur de script dans DBeaver soit bien positionné dans l'environnement de ce schéma.

Q 1 . Concevez les requêtes SQL nécessaires pour obtenir

1. la liste des communes par ordre décroissant des superficies. [code insee, nom, superficie]
2. la liste des communes dont le nom contient 'Lille' [code insee, nom]
3. la liste des communes avec leur code de département [code insee, département, nom]. NB : les 2 premiers caractères du code INSEE constituent le code de département. Consultez la doc de la fonction `substring`.
4. la liste des recensements dans toutes les communes par ordre décroissant alphabétique des communes et pour une même commune par ordre croissant des années de recensement. [code insee, nom, année de recensement, population totale]
5. la liste des communes par ordre décroissant de population (totale) selon le recensement de 2016 [code insee, nom, population totale]
6. la liste des communes par ordre décroissant de densité selon le recensement de 2016 [code insee, nom, population municipale, superficie, densité]. NB : la densité est le nombre d'habitants (population municipale) par km^2. On l'exprimera par un nombre entier.

`JOIN USING` *Cette forme syntaxique permet de simplifier l'écriture d'une jointure à réaliser selon l'égalité entre des colonnes de même nom.*
Si `r1` *comporte les attributs* `(a,b,c)` *et* `r2` *les attributs* `(a,c,d)`

```
SELECT  a, b, c, d
FROM r1
  JOIN r2
  USING (a,c)
```

équivaut à

```
SELECT  r1.a, b, r1.c, d
FROM r1
  JOIN r2
  ON r1.a = r2.a and r1.c = r2.c
```

Note :

- *il n'est pas indispensable (mais autorisé) de préfixer par le nom d'une table les noms des attributs intervenant dans le* `USING` *(ex :* `a` *au lieu de* `r1.a`*)*
- *le ou les attributs du USING figurent nécessairement entre parenthèses, même s'il y en a un seul.*

`NATURAL JOIN` *La forme «* `natural join` *» (jointure naturelle) est un raccourci syntaxique pour désigner une jointure selon l'égalité entre tous les attributs portant le même nom dans les 2 tables.* ***Attention : TOUS*** *les noms d'attributs communs aux deux tables interviennent « automatiquement » dans la condition de jointure.*

```
SELECT a, b, c, d
FROM r1
  NATURAL JOIN  r2
```

équivaut à

```
SELECT  a, b, c, d
FROM r1
  JOIN r2
  USING (a,c)
```

Note :

- *un même JOIN ne peut être associé qu'à* ***une seule*** *des formes ON, USING ou NATURAL*
- *NATURAL présente le défaut important de rendre la requête dépendante de l'ensemble des noms d'attributs. Supposons, par exemple, que lors d'une évolution ultérieure de la base de données, on ajoute une colonne nommée* `b` *à la table* `r2`*. Un requête déjà écrite avec NATURAL JOIN va changer de signification puisque les attributs nommés* `b` *vont maintenant intervenir dans la jointure, ce qui n'est pas forcément l'effet escompté. Il vaut donc souvent mieux utiliser USING plutôt que NATURAL*

Q 2 .
Proposez une écriture de la dernière requête de la question précédente en utilisant `USING`.

Q 3 . La table `stations` contient les stations de mesures de pollution sur le territoire de la MEL. Consultez la description de la table via phppgadmin.
Concevez les réquêtes SQL nécessaires pour obtenir

1. La liste des stations classées par ordre des noms de commune [nom de commune, nom de station, latitude, longitude]
2. La liste des stations par ordre des noms de commune. On souhaite que toutes les communes de la MEL apparaissent au moins une fois, même si elles ne disposent pas de station. [nom de commune, nom de station, latitude, longitude]

Q 4 . La table `mesures_mensuelles` contient des données de mesures mensuelles de pollution. Consultez la description de la table via phppgadmin.
Concevez les réquêtes SQL nécessaires pour obtenir

1. le nombre total de mesures
2. la moyenne, le maximum et le minimum mesurés pour le polluant de code 7
3. le nombre de mesures, la moyenne, le maximum et le minimum mesurés pour le polluant de code 7 par chaque station qui opère cette mesure.
4. idem, mais le résultat comportera en plus le nom de la station
5. idem, pour le polluant 6001
6. les stations dont la moyenne mesurée pour le polluant de code 6001 est supérieure à 10. Mêmes informations que précédemment.

7. le nombre de mesures, la moyenne, le maximum et le minimum mesurés pour chaque station et chaque polluant, classées par code de polluant. On ne souhaite que le code de la station et le code du polluant, pas leurs noms.
8. le nombre de mesures, la moyenne, le maximum et le minimum mesurés pour chaque station et chaque polluant. On souhaite, en plus les noms des stations et des polluants.