# TP noté: entrepôts de données

M1 ISD 2021-2022
Internet et toute forme de communication interdits (sauf cours).
Barême purement indicatif et susceptible de changer.
Le nombre de ⋆ indique le niveau approximatif de difficulté.

### Instructions

Écrire vos réponses dans le fichier `m1isd-nov21-nom-prenom.sql`. Renommer le fichier en remplaçant "nom" et "prénom" par votre nom et votre prénom, et en ajoutant le suffixe "-reponses": si vous vous appelez paul michu, le fichier devra donc être nommé: `m1isd-nov21-michu-paul-reponses.sql`. À la fin du TP, déposer le fichier sous ecampus.

Attention: il est important de répondre à la fois aux questions SQL et aux questions regex: les 2 parties seront corrigées indépendemment donc ne faites pas l'impasse sur l'une des 2.

## 1 Création de la base de données

Pour créer et charger la base de donnée de ce TP, un script vous est donné.

```
vous devriez avoir chargé (inutile de tout vérifier)
28370 lines dans geoitem
680 lines dans feature
7 lines dans continent
9 lines dans featureclass
252 lines dans country
```

## 2 Requêtes

- Écrire en SQL les requêtes ci-dessous **en respectant les noms de colonne** des illustrations.
- N'utilisez qu'un `SELECT` par requête: ceci vous interdit en particulier d'utiliser des sous-requêtes. Sauf lorsque l'énoncé indique d'utiliser un CTE, auquel cas vous pouvez bien sûr utiliser plusieurs `SELECT`.
- Dans chaque question, vous n'aurez qu'au plus la moitié des points si votre réponse fait appel à `GROUPING SETS`.

La base étant en anglais, les requêtes ont aussi été rédigées dans ce langage.

1. nombre d'éléments (`geoitems`) pour chaque paire constituée d'un classe (`featureclass`) et d'un continent. Trier le résultat par ordre décroissant. **(1pt)**⋆

```
        description          |    ctname     |  nb
-----------------------------+---------------+------
 city, village, ...          | Asia          | 7578
...
 country, state, region,... | Africa        |    1
(13 rows)
```

2. Nombre d'éléments (`geoitems`) et la plus haute altitude d'un élément (`elevation`) pour chacune des 4 combinaisons suivantes: **(4pt)**⋆

   - globalement (donc nombre total d'éléments et plus haute altitude répertoriée)
   - `(continent)`
   - `(country, continent)`
   - `(country, continent, item)`

   La requête sera limitée aux éléments dont l'altitude est au moins 1000, et le résultat sera trié par nombre croissant.

```
   continent   |    country     |              geoitem               |  nb  | elevation_max
---------------+----------------+------------------------------------+------+---------------
 Europe        | Italy          | Corno Plazza                       |    1 |        2650.0
...
 Europe        | Italy          |                                    | 2633 |        4765.0
 Europe        |                |                                    | 3808 |        4765.0
               |                |                                    | 3991 |        4765.0
(3939 rows)
```

3. Liste des éléments ayant une altitude renseignée, par altitude décroissante, avec leur rang (en cas d'ex-aequos, le rang n'augmente que de un pour l'élément suivant: par exemple; 1,2,2,2,2,2,3,4,4,5...) **(5pt)**⋆

```
                  asciiname                   | elevation | rang
----------------------------------------------+-----------+------
 Monte Bianco di Courmayeur                   |    4765.0 |    1
 ...
 Spalla (La Spedla)                           |    4020.0 |   16
 La Spedla                                    |    4020.0 |   16
 Dent du Geant                                |    4013.0 |   17
 ...
(8348 rows)
```

4. Afficher la liste des 10 éléments (geoitem) les plus peuplés par pays. Vous afficherez le résultat par ordre alphabétique de pays, et pour un même pays par population décroissante[1].

   Vous pouvez utiliser pour cette requête un CTE (il est conseillé de n'utiliser le CTE que pour filtrer les résultats d'une première requête). Vous n'êtes pas autorisés à utiliser une instruction du type "fetch" (comme `FETCH FIRST 10 ROWS ONLY`). **(5pt)**⋆⋆

```
            country             |                    asciiname                     |   pop
--------------------------------+--------------------------------------------------+----------
 Afghanistan                    | Kabul                                            |  3043532
 Afghanistan                    | Kandahar                                         |   391190
...
 Zimbabwe                       | Masvingo                                         |    76290
 Zimbabwe                       | Chinhoyi                                         |    61739
(1709 rows)
```

5. Cette question utilise, en plus de `geoitem`, une table `voyage(personne text,depart int,destination int,nb_j Decimal);` où `depart` et `destination` sont des clés étrangères référençant `geoitem.geonameid`. Chaque personne a effectué un voyage (visitant chaque item/localité au plus une fois).

   Il n'y a pas de contraintes pour la requête SQL sur ces questions: vous pouvez utiliser CTE, sous-requêtes... **(5pt)**⋆⋆

   a) Ecrire une requête sélectionnant l'id de la localité de départ de Durand (celle qui n'est pas une destination).

```
 depart
---------
 2973385
(1 row)
```

   b) Identifier pour chaque localité (geonameid) visité par "Durand" au bout de combien de jours de voyage et de combien d'étape il y est arrivé (on ne représentera pas la ville de départ).

```
  dest   | nb_j | nb_etapes
---------+------+-----------
 2972315 |  0.9 |         1
 2972328 |  2.4 |         2
 2974733 |  3.2 |         3
 3006767 |  5.6 |         4
(4 rows)
```

   c) Même requête que b) mais on veut afficher le nom de la localité au lieu de son id.

[1] Pour ne sélectionner que 10 éléments même dans le cas où, par exemple, les 11 plus peuplés ont la même population, on départagera les ex-aequos en prenant ceux de plus petit id

```
    asciiname     | nb_j | nb_etapes
------------------+------+-----------
 Toulouse         |  0.9 |         1
 Toulon           |  2.4 |         2
 Sete             |  3.2 |         3
 La Roche-sur-Yon |  5.6 |         4
(4 rows)
```

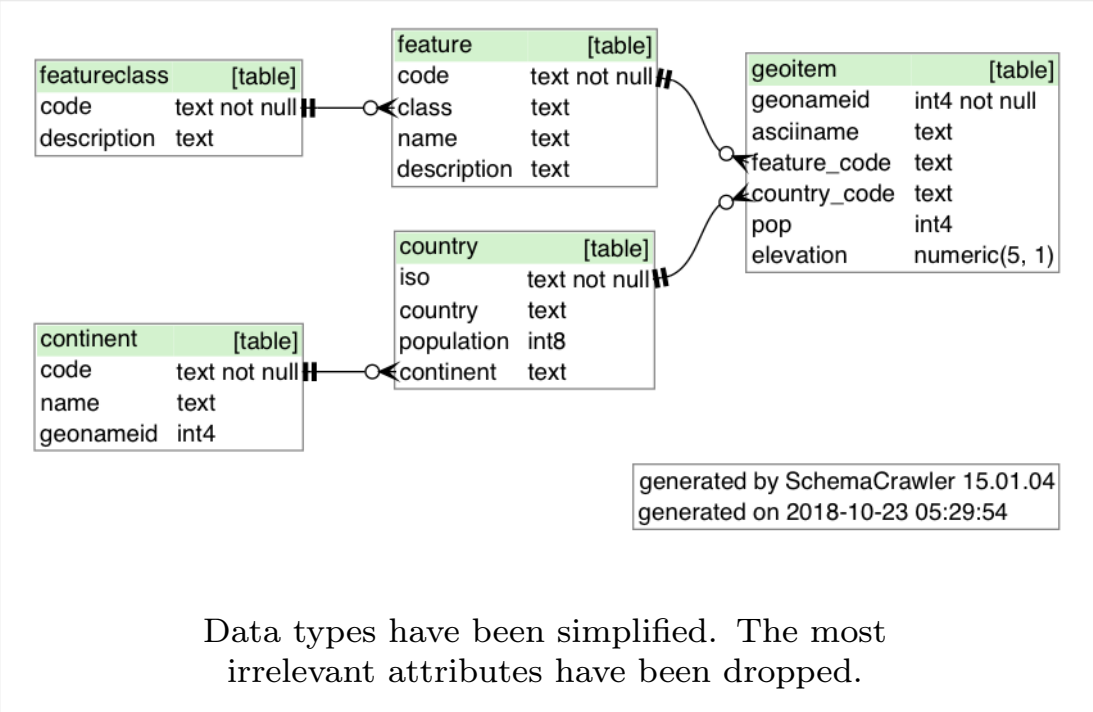


Figure 1: Database schema.

## 3 Expressions Régulières

1. Expliquer le résultat des 3 expressions Python ci-dessous : **(1pt)**⋆

```
import re
re.match(r'a*(b|ac)+c','aaacc').group()
re.match(r'a*?(b|ac)*c','aaacc').group()
re.match(r'a*(b|ac)*c','aaacc').group()
```

2. Compléter le code ci-dessous avec une expression régulière pour convertir des formules latex en asciidoc en remplaçant dans le texte ci-dessous chaque expression entre "$" par stem:[ l'expression ] **(4pt)**⋆⋆

```
import re
regex_find = r'a completer'
motif_replace = r'a completer'
res = re.sub(regex_find,motif_replace,'Ceci est un texte avec une formule $2+3$ et aussi $4=8/2$')
# res == 'Ceci est un texte avec une formule stem:[2+3] et aussi stem:[4=8/2]'
# Indication : une variable de rege python se note \1 dans le motif de remplacement ex: r'\3-\1'
```