TP - Importation de données

L'exercice consiste à télécharger une liste de fichiers au format `XML`, les convertir au format `CSV` puis peupler une base de données à l'aide de ces fichiers.

Organisation du travail:

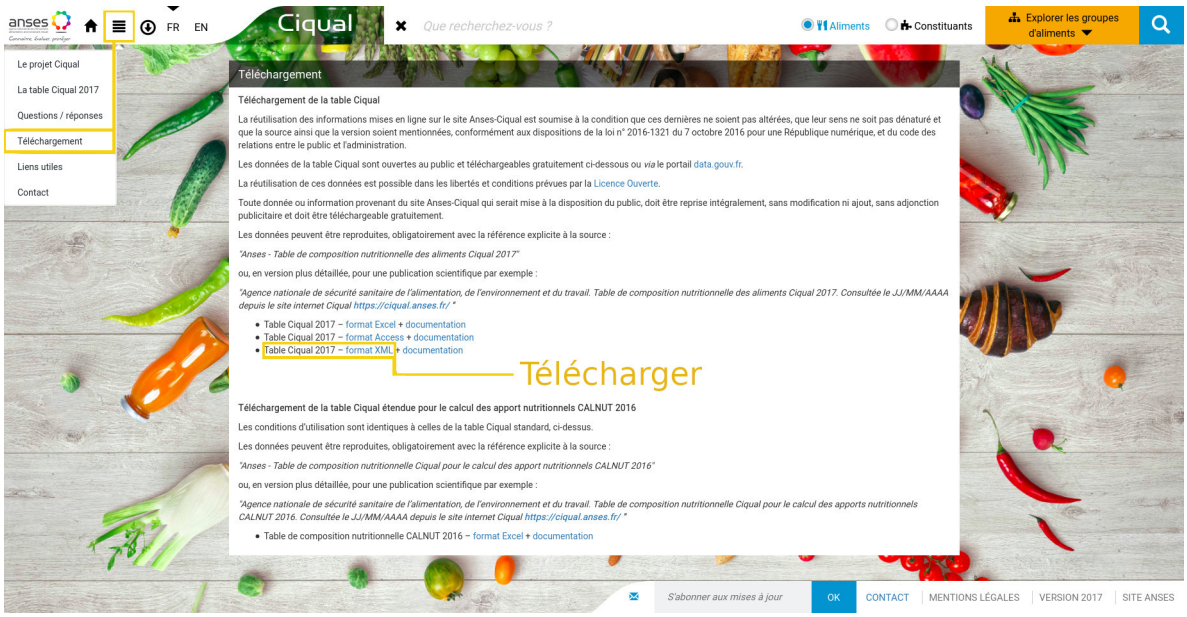
Pré-requis

Pour ce TP, il faut au préalable avoir installer:

- **PostgreSQL** : **S**ystème de **G**estion de **B**ase de **D**onnée (**SGBD**)
- **PgAdmin4** : Plateforme open source de développement et d'administration pour **PostgreSQL**
- **PgModeler** : Outil de modélisation de base de données
- **Apache Maven** : Outil logiciel libre pour la gestion et l'automatisation de production des projets logiciels Java comparable au système `Make` sous `Unix`
- **OpenJDK** : Implémentation libre de la société **Oracle®** du standard **Java** sous *Licence Publique Générale*

Télécharger les fichiers au format XML sur la platforme Ciqual

Se rendre sur Ciqual pour télécharger les fichiers :



2 - Utiliser un outil de conversion XML -> CSV

Se rendre sur le **xml2csv** puis cloner ce projet dans le dossier de votre choix:

```
# Exemple
cd /home/nom_utilisateur
git clone https://github.com/fordfrog/xml2csv.git
```

Pour pouvoir utiliser l'outil **xml2csv**, il faut au préalable installer la plate-forme d'exécution JRE d'**OpenJDK 7 ou +** et **Apache Maven 3 ou +** :

```
# Effectuer tout d'abord la mise à niveau et à jour des paquets
sudo apt update && sudo apt upgrade

# Puis installer ces paquets
sudo apt install default-jdk
sudo apt install default-jre
sudo apt install maven
```

Verifier ensuite l'installation de ces paquets :

```
javac -version  # javac 11.0.6
java -version   # openjdk version "11.0.6" 2020-01-14
mvn -version    # Apache Maven 3.6.3  Maven home: /opt/maven
```

Passons maintenant à la compilation de l'outil **xml2csv** :

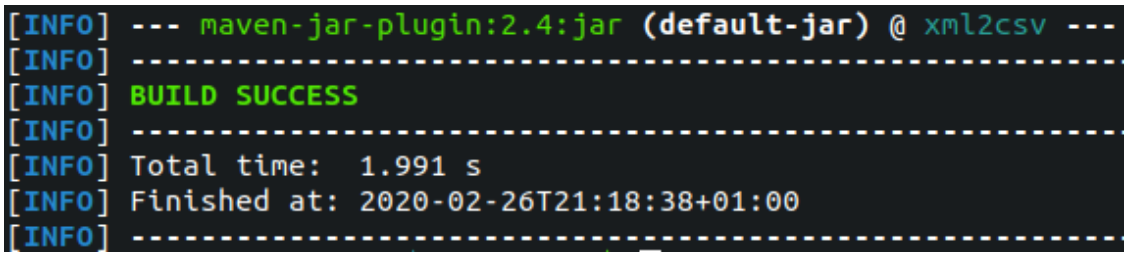
- Placer vous dans le dossier **xml2csv** cloné plus tôt en veillant bien à être à la racine où se trouve le fichier `pom.xml` :

  ```
  cd xml2csv
  ls  # pom.xml  README.md  src  target
  ```

- Vous pouvez maintenant lancer la compilation à l'aide de **maven** et sa commande :

```
mvn package
```

Vous devriez obtenir ce résultat en terminal :



Vous devrez également renommer le dossier et les fichiers téléchargés au format **XML** en veillant a ce que les espaces soient remplacés par des `underscore` _ :

Fichiers originaux téléchargés	Fichier renommés
alim_2017 11 21.xml	alim_2017_11_21.xml
alim_grp_2017 11 21.xml	alim_grp_2017_11_21.xml
compo_2017 11 21.xml	compo_2017_11_21.xml
const_2017 11 21.xml	const_2017_11_21.xml
sources_2017 11 21.xml	sources_2017_11_21.xml

Nous pouvons maintenant passer à la conversion de nos fichiers XML en utilisant ces commandes :

Créer un dossier prêt à recevoir vos fichiers convertis en **CSV** :

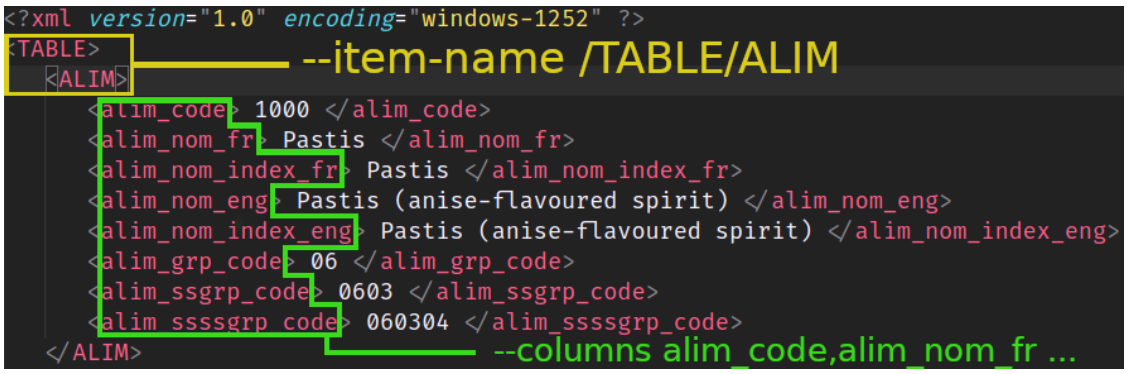
```
# Exemple
mkdir Documents/Ciqual
```

Positionnez vous dans le dossier `target` du dossier `xml2csv` cloné, le fichier `xml2csv-1.2.2.jar` présent dans ce dossier nous est utile pour lancé la conversion

```
cd xml2csv/target
```

Puis convertissez vos fichiers xml en csv à l'aide de ces commandes, adaptez les en fonction de vos besoin / tables:

- Pour cela et avant tout, ouvrez vos fichiers `.xml` afin de connaitre le nom de table et le nom des colonnes :
    - Exemple pour la table aliments



La conversion peut maintenant commencer:

```
# --columns => nom_des_colonnes contenues dans le fichier xml (voir image ci
dessus)
# --input => emplacement du dossier contenant les fichiers .xml téléchargés
depuis https://ciqual.anses.fr/
# --output => emplacement souhaité pour recevoir les fichiers .csv une fois la
conversion achevée => Documents/Ciqual/ dans notre cas
# --item-name => nom des tags XML, TABLE => tag générique 'englobant' toutes les
colonnes, ALIM => nom de la table

# aliments.csv
java -jar xml2csv-*.jar  # xml2csv-*.jar signifie => utilise n'importe quelle
version
--columns
alim_code,alim_nom_fr,alim_nom_index_fr,alim_nom_eng,
alim_nom_index_eng,alim_grp_code,alim_ssgrp_code,alim_sssgrp_code
--input ~/Documents/Ciqual/TableCiqual2017_XML_2017_11_21/alim_2017_11_21.xml
--output ~/Documents/Ciqual/aliments.csv
--item-name /TABLE/ALIM

# groupes_aliments.csv
java -jar xml2csv-*.jar
--columns
alim_grp_code,alim_grp_nom_fr,alim_grp_nom_eng,alim_ssgrp_code,alim_ssgrp_nom_fr
,
alim_ssgrp_nom_eng,alim_sssgrp_code,alim_sssgrp_nom_fr,alim_sssgrp_nom_eng
--input
~/Documents/Ciqual/TableCiqual2017_XML_2017_11_21/alim_grp_2017_11_21.xml
--output ~/Documents/Ciqual/groupes_aliments.csv
--item-name /TABLE/ALIM_GRP

# compositions.csv
java -jar xml2csv-*.jar
--columns alim_code,const_code,teneur,min,max,code_confiance,source_code
--input ~/Documents/Ciqual/TableCiqual2017_XML_2017_11_21/compo_2017_11_21.xml
--output ~/Documents/Ciqual/compositions.csv
--item-name /TABLE/COMPO

# constituants.csv
java -jar xml2csv-*.jar
--columns const_code,const_nom_fr,const_nom_eng
--input ~/Documents/Ciqual/TableCiqual2017_XML_2017_11_21/const_2017_11_21.xml
--output ~/Documents/Ciqual/constituants.csv
--item-name /TABLE/CONST

# sources.csv
```

```
java -jar xml2csv-*.jar
--columns source_code,ref_citation
--input ~/Documents/Ciqual/TableCiqual2017_XML_2017_11_21/sources_2017_11_21.xml
--output ~/Documents/Ciqual/sources.csv
--item-name /TABLE/SOURCES
```

Peupler des tables temporaires dans pgAdmin à l'aide des données converties en CSV

Nous allons devoir créer les tables (5 au total) dans pgAdmin.

Tout d'abord, créer une base de donnée du nom **ciqual**, une fois créee, effectuer un click droit sur cette base et ouvrer **Query Tool**, un éditeur de script **PLSQL** s'ouvre alors.

Nous allons créer les 5 tables temporaires à l'aide de 5 requêtes. Il va falloir bien respecter l'ordre des colonnes et surtout le **type** auxquelles les données sont liées implicitement:

```
-- Table aliments_temp
CREATE TABLE aliments_temp(
    alim_code INTEGER,
    alim_nom_fr VARCHAR (255),
    alim_nom_index_fr VARCHAR (255),
    alim_nom_eng VARCHAR (255),
    alim_nom_index_eng VARCHAR (255),
    alim_grp_code INTEGER,
    alim_ssgrp_code INTEGER,
    alim_sssgrp_code INTEGER
);

-- Table groupes_aliments_temp
CREATE TABLE groupes_aliments_temp(
    alim_grp_code INTEGER,
    alim_grp_nom_fr VARCHAR (255),
    alim_grp_nom_eng VARCHAR (255),
    alim_ssgrp_code INTEGER,
    alim_ssgrp_nom_fr VARCHAR (255),
    alim_ssgrp_nom_eng VARCHAR (255),
    alim_sssgrp_code INTEGER,
    alim_sssgrp_nom_fr VARCHAR (255),
    alim_sssgrp_nom_eng VARCHAR (255)
);

-- Table compositions_temp
CREATE TABLE compositions_temp(
    alim_code INTEGER,
    const_code INTEGER,
    teneur VARCHAR (255),
    min_missing VARCHAR (255),
    max_missing VARCHAR (255),
    code_confiance VARCHAR (255),
    source_code INTEGER
);

-- Table constituants_temp
```

```
CREATE TABLE constituants_temp(
    const_code INTEGER,
    const_nom_fr VARCHAR (255),
    const_nom_eng VARCHAR (255)
);

-- Table sources_temp
CREATE TABLE sources_temp(
    source_code INTEGER,
    ref_citation VARCHAR (255)
);
```

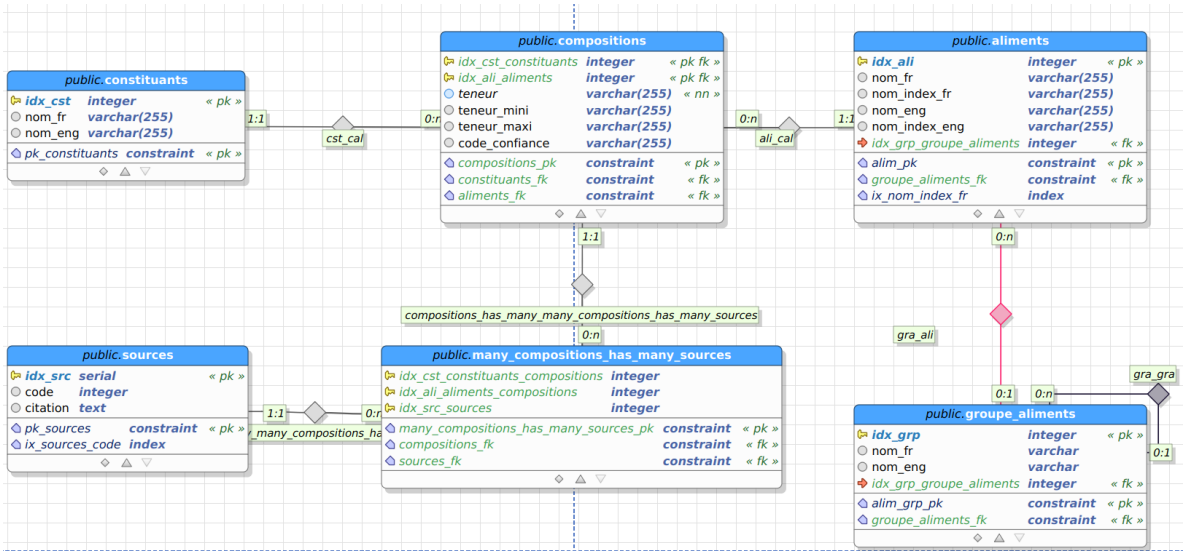
Nous pouvons à présent importer les données contenues dans nos fichiers **CSV** dans nos tables temporaires:

- Toujours dans **pgAdmin4**, clicker sur le chevron ❯ devant `Schemas`, puis sur le chevron ❯ devant `Public` et enfin sur le chevron ❯ devant `Tables`, vous devriez voir vos 5 tables temporaires fraichement crées.
- Clicker droit sur chacune de ces tables temporaires puis clicker sur **Import/Export**, dans la fenêtre qui s'ouvre, veillez bien a ce que le bouton Import/Export soit sur la position **Import** (en vert), dans le champ **Filename**, selectionnez le fichier correspondant a la table temporaire dans laquelle vous souhaitez importer les données, Ex: aliments_temp => aliments.csv à selectionner. Pour le champ **Format** selectionnez bien **CSV**, **Encoding** sur **UTF-8** et faites en sorte que le bouton **Header** soit sur position **Yes** (en vert).

  Vous pouvez maintenant clicker sur **Ok**.

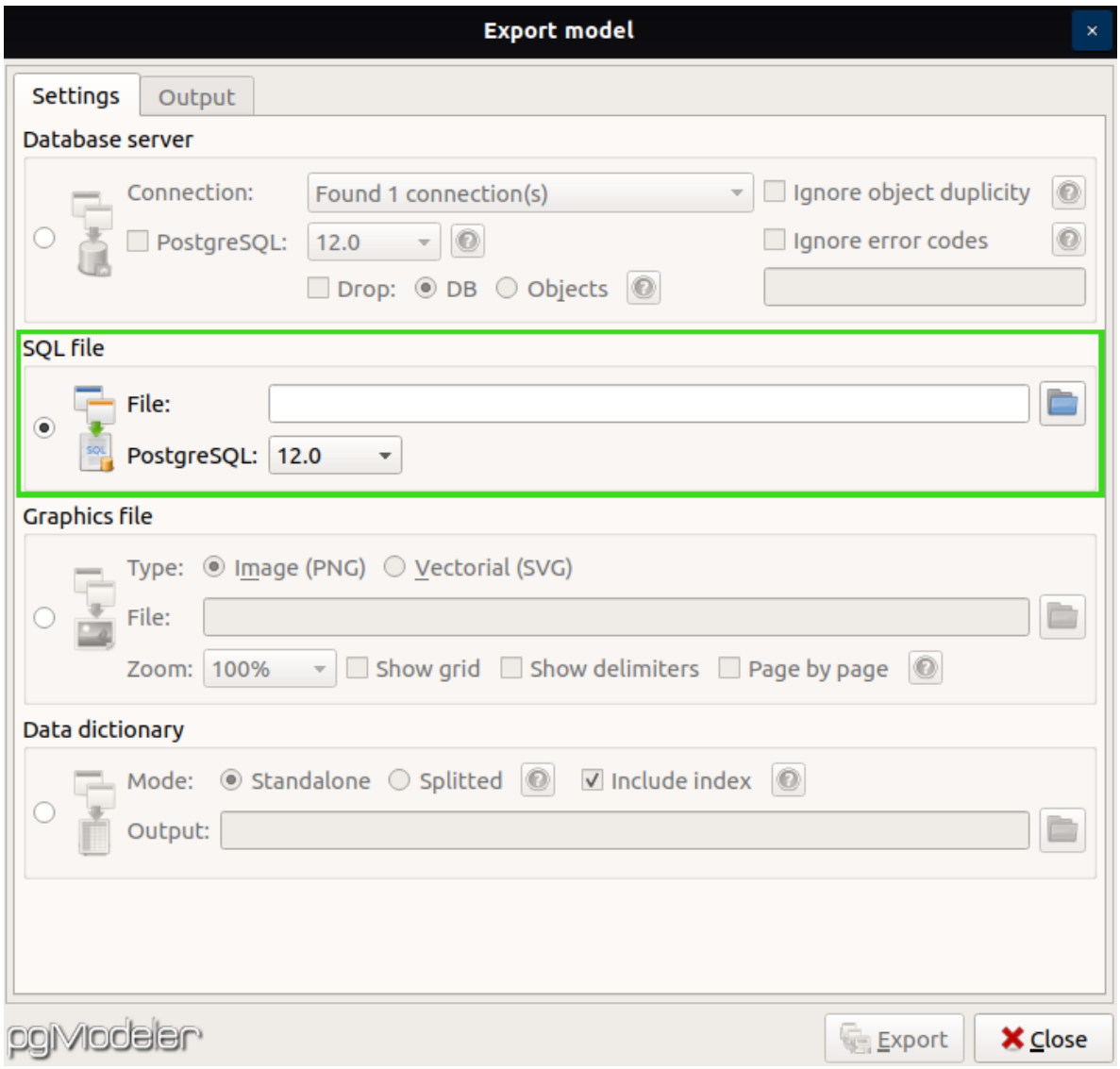
Nos tables temporaires sont maintenant bien peuplées à partir des données contenues dans les fichiers **CSV**.

Réaliser un Modèle Conceptuel de Données (MCD) à l'aide de pgModeler



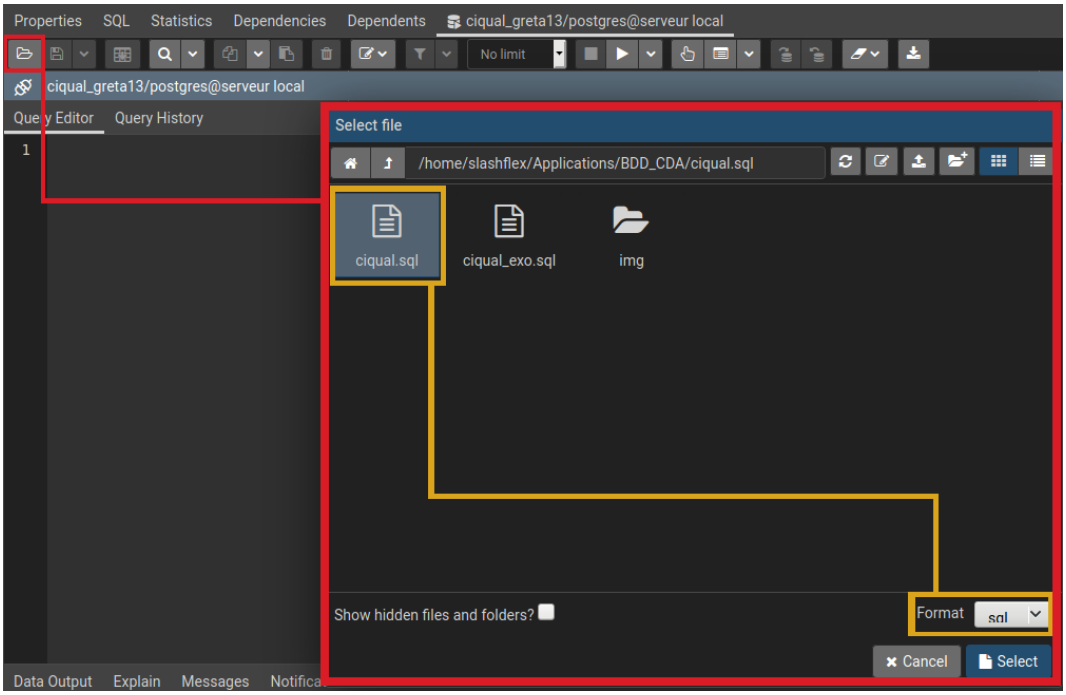
Une fois le **M**odel **C**onceptuel de **D**onnées crée, nous pouvons maintenant l'exporter au format **SQL** en clickant sur le bouton **Export** dans le menu de gauche, une fenetre s'ouvre alors:

- Cochez le bouton correspondant à Fichier **SQL** et sauvegardez votre fichier .sql à l'emplacement de votre choix



Nous pouvons ensuite importer notre script **SQL** fraichement créer dans **pgAdmin4** :

- Positionnez vous sur votre base de donnée (ciqual, dans notre cas)
- Clicker droit sur la base de données **ciqual** puis selectionner **Query Tool**
- Dans la fenetre s'ouvrant sur la droite, cliquer sur le bouton **Open file** (en forme de dossier ouvert) :
    - Cette fenêtre s'ouvre alors :



Veillez bien a selectionner le format **SQL**, puis clicker sur **Select**.

Vous devriez maintenant voir vos 5 Tables (`aliments`, `compositions`, `constituants`, `groupes_aliments` et `sources`).

Nous pouvons maintenant finaliser l'insertion des données inclues dans nos 5 tables temporaires, et peupler nos tables finales avec ces même données.

Toujours à l'aide de **Query Tools**, nous allons executer ce script

```
-- INSERTION DES DONNEES A PARTIR DES TABLES TEMPORAIRES

-- Table sources
INSERT INTO sources(code, citation)
    SELECT source_code, ref_citation
    FROM sources_temp;

-- Table constituants
INSERT INTO constituants (idx_cst, nom_fr, nom_eng)
    SELECT const_code, const_nom_fr, const_nom_eng
    FROM constituants_temp;

-- Table groupe_aliments
INSERT INTO groupe_aliments (idx_grp, nom_fr, nom_eng)
    SELECT DISTINCT alim_grp_code, alim_grp_nom_fr, alim_grp_nom_eng
    FROM groupes_aliments_temp;

INSERT INTO groupe_aliments(idx_grp, nom_fr, nom_eng, idx_grp_groupe_aliments)
    SELECT DISTINCT alim_ssgrp_code, alim_ssgrp_nom_fr, alim_ssgrp_nom_eng,
alim_grp_code
    FROM groupes_aliments_temp;

INSERT INTO groupe_aliments(idx_grp, nom_fr, nom_eng, idx_grp_groupe_aliments)
    SELECT DISTINCT alim_ssssgrp_code, alim_ssssgrp_nom_fr,
alim_ssssgrp_nom_eng, alim_ssgrp_code
    FROM groupes_aliments_temp
    WHERE alim_ssssgrp_code != 0;
```

```
-- Table aliments
INSERT INTO aliments (idx_ali, nom_fr, nom_index_fr, nom_eng, nom_index_eng,
idx_grp_groupe_aliments)
    SELECT alim_code, alim_nom_fr, alim_nom_index_fr, alim_nom_eng,
alim_nom_index_eng,
        CASE
            WHEN alim_sssgrp_code != 0 THEN alim_sssgrp_code
            WHEN alim_sssgrp_code = 0 AND alim_ssgrp_code != 0 THEN
alim_ssgrp_code
            WHEN alim_ssgrp_code = 0 THEN alim_grp_code
        END AS code_grp
    FROM aliments_temp;

-- Nous devons maintenant créer une fonction qui premet de convertir une entrée
de type chaine de charactère en type numerique et d'appliquer un traitement
CREATE FUNCTION charToNum(chaine VARCHAR) RETURNS NUMERIC AS $$
DECLARE
    multi VARCHAR = '1';
    param NUMERIC = LENGTH(chaine) - (POSITION(',' IN chaine) + 1);
    counter INTEGER = 0;
BEGIN
    LOOP
        EXIT WHEN counter > param ;
            counter = counter + 1 ;
            multi = CONCAT(multi, '0')  ;
    END LOOP ;

    -- Si la chaine est un entier, on soustrait par 0.1
    IF POSITION(',' IN chaine) = 0
        THEN RETURN (CAST(REPLACE(chaine,',','.') AS NUMERIC) - 0.1);
    ELSE
        -- Le TRIM TRAILING permet de retirer les 0 inutiles 10.02000.. devient
10.02
        RETURN TRIM(TRAILING'0' FROM CAST(CAST(REPLACE(chaine,',','.') AS
NUMERIC) - (1 / CAST(multi AS NUMERIC)) AS VARCHAR(255)));
    END IF;
END; $$
LANGUAGE PLPGSQL;

-- Nous pouvons maintenant utiliser cette fonction pour importer nos données et
les traiter correctement
-- Table compositions
INSERT INTO compositions (idx_cst_constituants, idx_ali_aliments, teneur_mini,
teneur_maxi, code_confiance, teneur)
  SELECT const_code, alim_code, CAST(REPLACE(min_missing, ',', '.') as NUMERIC),
CAST(REPLACE(max_missing, ',', '.') as NUMERIC), code_confiance,
      CASE
        WHEN TRIM(teneur) = '-'
            THEN 0
        WHEN TRIM(teneur) = 'traces'
            THEN 0.0000001
        WHEN LEFT(TRIM(teneur), 1) = '<'
            THEN charToNum(REPLACE(TRIM(teneur),'<',''))
        ELSE
            CAST(REPLACE(teneur,',','.') as NUMERIC)
      END AS teneur
    FROM compositions_temp;
```

```
-- Table many_compositions_has_many_sources
INSERT INTO many_compositions_has_many_sources (idx_src_sources,
idx_ali_aliments_compositions, idx_cst_constituants_compositions)
    SELECT S.idx_src, CO.alim_code, CO.const_code
    FROM compositions_temp CO
    INNER JOIN sources S
    ON CO.source_code = S.code;
```

Voilà, toutes nos données sont correctement insérées.

Réaliser un Modèle Conceptuel de Données (MCD) à l'aide de pgModeler

Exercice n°1 :

Créer une requête qui permet d'afficher une liste de valeurs avec une chaîne de caractères en paramètre. La list e permet de choisir un aliment. Il faut afficher le nom et les groupes des aliments pour aider au choix :

```
SELECT
  aliments.nom_index_fr AS nom_aliment,
  g1.idx_grp AS id_grp,
  g1.nom_fr AS nom_grp,
  --g2.idx_grp AS id_grp_father,
  g2.nom_fr AS nom_grp_father,
  --g3.idx_grp AS id_grp_grand_father,
  g3.nom_fr AS nom_grp_grand_father
FROM aliments
  LEFT JOIN groupe_aliments AS g1 ON g1.idx_grp =
aliments.idx_grp_groupe_aliments
  LEFT JOIN groupe_aliments AS g2 ON g2.idx_grp = g1.idx_grp_groupe_aliments
  LEFT JOIN groupe_aliments AS g3 ON g3.idx_grp = g2.idx_grp_groupe_aliments
WHERE TRIM(nom_index_fr)
ILIKE '%Ban%';
```

Exercice n°2 :

Créer une requete pour afficher toute la composition d'un aliment avec les teneurs en fonction du poids et de l'aliment indiqués dans la requête :

```
-- Création d'une vue
CREATE VIEW compo_const
AS
  SELECT idx_ali_aliments, nom_fr, teneur
  FROM compositions
  INNER JOIN constituants
  ON idx_cst = idx_cst_constituants;

-- L'on créer une fonction qui prends en paramètre l'ID d'un aliment (idx_ali)
et le poids en grammes (poids)
CREATE OR REPLACE FUNCTION calcteneur (idx_ali INTEGER, poids NUMERIC)
  RETURNS TABLE (nom_fr VARCHAR, teneur DOUBLE PRECISION)
```

```
  AS $$
BEGIN
  RETURN QUERY
  SELECT CAST(REPLACE(compo_const.nom_fr, '/100g', '') AS VARCHAR),
CAST(compo_const.teneur AS DOUBLE PRECISION) * poids / 100
  FROM compo_const
  WHERE idx_ali = compo_const.idx_ali_aliments;
END;
$$
LANGUAGE PLPGSQL;

-- On appele la fonction avec en paramètre :
-- 2018 = ID d'un aliment, 100 = poids en grammes
SELECT * FROM calcteneur(2018, 100);
```

Cette dernière commande nous retourne la teneur de chaque constituants pour un aliment donnée et un poids en grammes donné, Ex :

nom_fr	teneur
Energie, Règlement UE N° 1169/2011 (kJ)	0
Eau (g)	81.2
Cendres (g)	0.68
Fer (mg)	1.18

Une dernière commande afin de concaténer dans une nouvelle colonne `text` les données contenues dans la table `aliments` et les noms du groupe, sous-groupe et sous-sous-groupe auquel cet `aliment` appartient :

```
SELECT CONCAT(aliments.nom_index_fr,  ' / ', g1.nom_fr, ' / ', g2.nom_fr, ' / ',
g3.nom_fr) AS text
FROM aliments
    LEFT JOIN groupe_aliments AS g1
        ON g1.idx_grp = aliments.idx_grp_groupe_aliments
    LEFT JOIN groupe_aliments AS g2
        ON g2.idx_grp = g1.idx_grp_groupe_aliments
    LEFT JOIN groupe_aliments AS g3
        ON g3.idx_grp = g2.idx_grp_groupe_aliments
WHERE TRIM(nom_index_fr) ILIKE '%Pastis%';
```

Ce qui nous donne pour une recherche avec en mot clé **Pastis** :

text
Pastis sans alcool / cocktails / boisson alcoolisées / boissons
Pastis / cocktails / boisson alcoolisées / boissons
