Université de Bordeaux

Sciences et Technologies

Master de Bioinformatique - Parcours Biologie Computationnelle

Année 2019-2020

Projet de Bases de Données

TD Insee

Abdelghani Neuhaus

Sommaire

Introduction

L'INSEE (Institut National Statistiques Etudes Economiques) publie régulièrement de nombreuses bases de données et est en charge du recensement de la population française. Parmi les données issues du référencement, nous allons travailler sur des données sociales et environnementales sur les départements français de 2008 à 2016. Il s'agira de construire une base de données contenant la géographie française actuelle (régions et départements) et les données sociales et environnementales associées à chaque département.

Partie 1

Implémentation de la base de données

1.1 Création de la base de données

La création de la base de données est réalisée manuellement via un terminal. Elle est donc locale. Il faut renseigner son identifiant (username) et le nom de la base de données que l'on souhaite donner (databaseName).

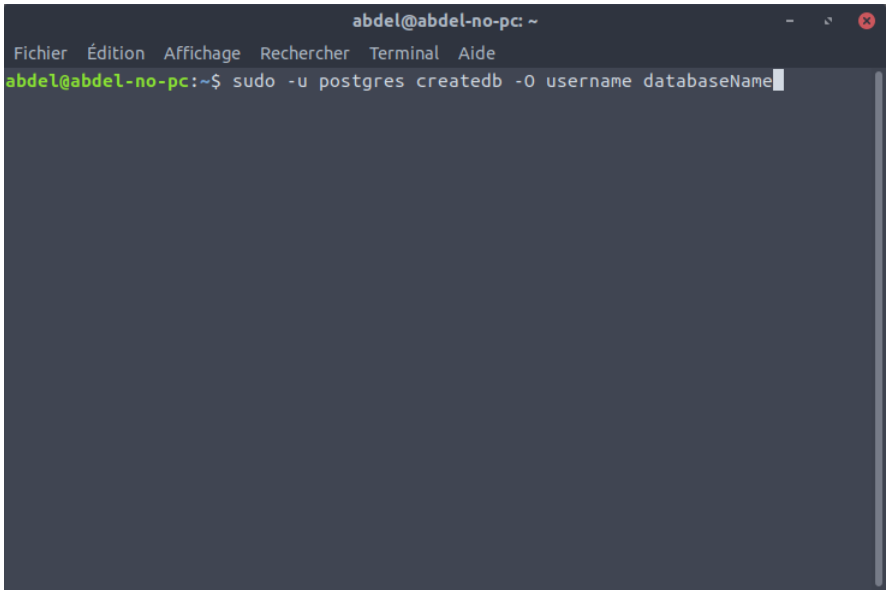


Figure 1 : Création d'une base de données via un terminal Unix.

On peut ensuite y accéder avec la commande suivante : **psql -h 127.0.0.1 -U username databaseName**. La base de données est désormais créée. Nous pouvons la remplir.

1.2 Liaison avec la base de données

Pour la connexion à la base de données grâce à l'exécution d'un script python, nous utiliserons le module **psycopg2**. Ce dernier est l'adaptateur de la base de données PostgreSQL le plus populaire pour le langage de programmation Python. Nous utilisons principalement cinq méthodes de ce module :

— **connect** : cette méthode permet d'établir la connexion avec une base de données. Il faut renseigner divers paramètres tels que notamment notre identifiant, notre mot de passe, le nom de

la base de données d'intérêt.

— **cursor** : c'est un objet qui parcourt les éléments que l'on va récupérer lors des requêtes que l'on effectuera.

— **execute** : cette méthode permet d'exécuter une requête SQL.

— **commit** : cette méthode permet d'appliquer les requêtes que l'on effectue et ainsi mettre à jour la base de données.

— **close** : permet de fermer la connexion et le curseur utilisés.

1.3 Extraction des données et création des relations de la base de données

a) Depuis un fichier CSV

Pour extraire des données depuis un fichier CSV, nous utilisons le module **csv**. Ce module permet de lire des tableaux stockés dans des fichiers au format CSV, puis de les manipuler avec Python (affichage des premières lignes, stockage des données, etc...). Dans notre cas, nous avons crée une table que nous remplissons ligne par ligne avec les éléments récupérés dans le fichier CSV. L'ouverture du fichier est possible grâce à la méthode **open()**. La méthode **next()** permet de passer à la ligne suivante lors de la lecture du fichier.

```
1  cursor.execute("""CREATE TABLE Regions(reg varchar(2) PRIMARY KEY,
2  cheflieu varchar(10) not null,
3  tncc varchar(10) not null,
4  ncc varchar(40) not null,
5  nccenr varchar(40) not null,
6  libelle varchar(40) not null)""")
7  with open('region2020.csv', 'r') as file:
8      reader = csv.reader(file)
9      next(reader)
10     for row in reader:
11         cursor.execute("INSERT INTO Regions VALUES (%s, %s, %s, %s, %s, %s)",row)
```

Figure 2 : Implémentation d'une relation à partir de données d'un fichier CSV.

b) Depuis un fichier Excel

Pour extraire les données depuis un fichier Excel, nous utilisons la librairie **Pandas**. Pandas est un outil rapide, puissant, flexible et facile d'utilisation pour l'analyse et la manipulation de données scientifiques principalement et est utilisé avec Python.

Tout d'abord, nous ouvrons le fichier grâce à la méthode ExcelFile. Ensuite, nous créons un data frame qui sera rempli avec les éléments lus par la méthode **read_excel()**. Il faut renseigner le nom du fichier, le nom de la feuille Excel utilisée, éventuellement les colonnes et les lignes à extraire. Ensuite, nous remplaçons les cases contenants des nd, nc et des vides par la valeur NaN qui est considérée

comme un float (pour cela, nous utilisons la librairie **NumPy**) avec la méthode **remplace()**.

```
cursor.execute("""CREATE TABLE DepartementsInfos(
cheflieu varchar(10) PRIMARY KEY,
dep varchar(30),
espH2015 float,
espH2010 float,
espF2015 float,
espF2010 float,
disSanteSup7 float,
inondable2013 float,
inondable2008 float)""")
fileOpen = pd.ExcelFile('DD-indic-reg-dep_janv2018.xls')
dataFrame2 = pd.read_excel(fileOpen, sheet_name="Social", usecols="A:I", skiprows=
    28, nrows=104 , header=None)
dataFrame2.replace(["nd", "nd ", "nc", "nc "], np.NaN, inplace = True)
for i in range (len(dataFrame2)):
    values = dataFrame2.iloc[i]
    cursor.execute("INSERT INTO DepartementsInfos values (%s, %s, %s, %s, %s, %s, %s
    , %s, %s)", values)
```

Figure 3 : Création d'une relation à partir de données issues d'un fichier Excel.

1.4 Schéma de base de données

Ci-dessous, voici un schéma de la base de données créée et utilisée :

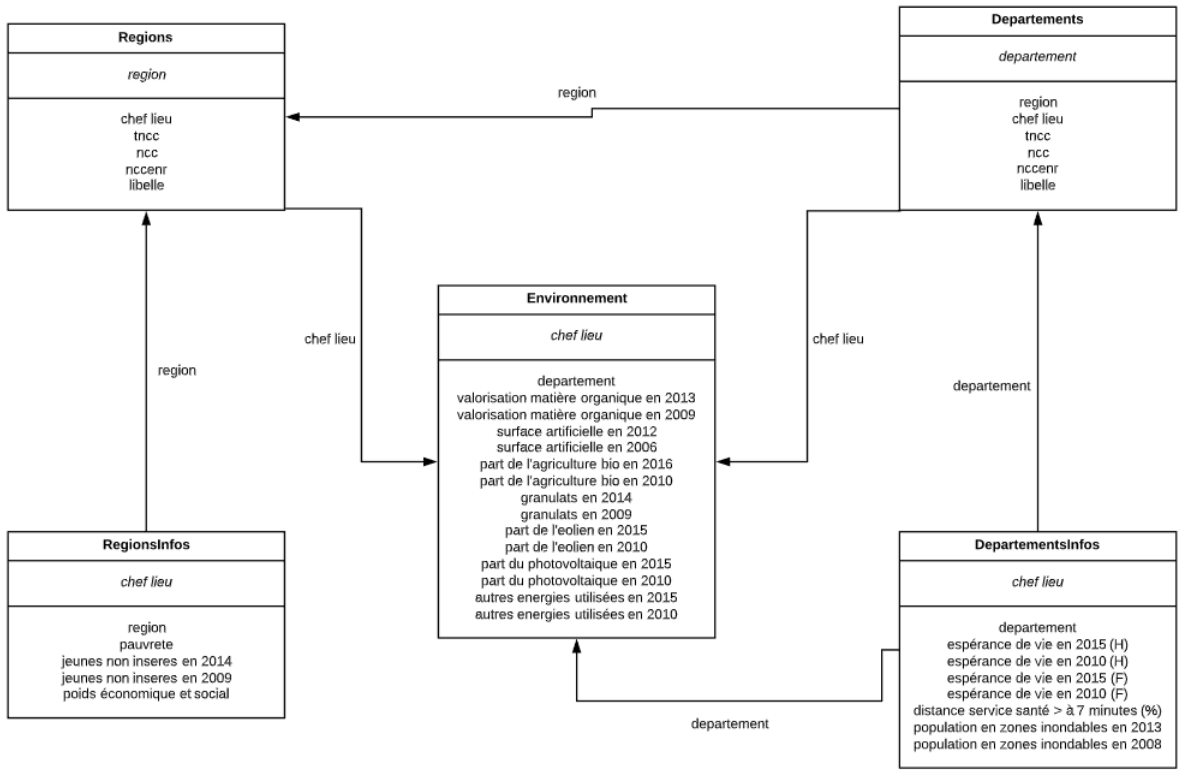


Figure 4 : Schéma de la base de données (noms des relations en gras et clés primaires en italique). Cette image est aussi en annexe avec une taille plus grande.

Partie 2

Réponses aux questions

Les réponses écrites ci-dessous sont bien évidemment dans le script de réponses **questionsBBD.py** présent dans l'archive fournie.

2.1 Première partie

1) **Liste des régions**
SELECT nccenr FROM Regions ;
Explication : Nous sélectionnons tous les noms en minuscules des régions de la table Regions.

2) **Liste des départements**
SELECT nccenr FROM Departements ;
Explication : Nous sélectionnons tous les noms en minuscules des départements de la table Departements.

3) **Données de la région saisie par l'utilisateur**
SELECT nccenr, cheflieu, tncc FROM Regions ;
Explication : Nous sélectionnons tous les noms en minuscules des départements de la table Departements.

4) **Données demandées (sociales ou environnementales) pour le département choisi par l'utilisateur**
SELECT nccenr FROM Departements ;

Puis, selon le choix :

Social
SELECT cheflieu, espH2015, espH2010, espF2015, espF2010, distravailsup7, inondable2013, inondable2008 FROM DepartementsInfos WHERE dep = '%s' ;(*%s est remplacé par la saisie de l'utilisateur*)

Environnemental
SELECT cheflieu, valorga2013, valorga2009, surfarti2012, surfarti2006, agribio2016, agribio2010, granulats2014, granulats2009, eolienne2015, eolienne2010, photovoltaique2015, photovoltaique2010, autreenergie2015, autreenergie2010 FROM Environnement WHERE dep = '%s' ; (*%s est remplacé par la saisie de l'utilisateur*)
Explication : Nous sélectionnons toutes les données de la table correspondant au choix de l'utilisateur et nous les affichons.

5) **Liste des départements où la part de l'énergie choisie a augmenté entre les deux années de référence, classés de la plus forte augmentation à la plus faible**

SELECT dep, energieAprès, energieAvant FROM environnement WHERE energie2015 > energie2010 ORDER BY energie2015 - energie2010 DESC ;

Explication : Nous sélectionnons les départements, les valeurs de consommation d'énergie de deux moments données. Ensuite, nous conservons les départements dont la valeur de consommation d'énergie a augmenté entre 2015 et 2010, puis nous les trions par ordre décroissant selon la valeur du différence de pourcentage (on affiche une différence entre deux pourcentages).

2.2 Deuxième partie

1) **Départements dont la région a eu une production de granulats supérieure à 25 000 000 tonnes en 2014**

SELECT D1.nccenr FROM Departements D1 WHERE D1.reg IN (SELECT R1.reg FROM departements D1 JOIN environnement E1 ON D1.nccenr = E1.dep JOIN regions R1 ON R1.reg = D1.reg GROUP BY R1.reg HAVING SUM(granulats2014) > 25000000) ;

Explication : Nous sélectionnons les départements (via leurs noms) depuis la table Departements où les régions auxquelles ils appartiennent existent dans la jointure entre les tables Departements avec Environnement (jointure sur le nom du département) et avec Regions (selon le numéro de la région). Nous avons ainsi pour chaque département un numéro qui correspond à la région. Nous réalisons un groupement des numéros de régions identiques (par exemple Gironde et Landes ont le même numéro et on additionne leurs valeurs de granulats). Nous affichons seulement les valeurs à 25 000 000 de tonnes de granulats produits en 2014.

2) **Afficher les 5 départements avec le plus grand taux d'énergie éolienne comme source de la puissance électrique en 2015**

SELECT dep FROM environnement WHERE eolienne2015 != 'nan' ORDER BY eolienne2015 DESC LIMIT 5 ;

Explication : Nous sélectionnons les départements de la table Environnements dont la valeur du pourcentage d'utilisation de l'éolienne en 2015 est renseignée (différente de "NaN") et nous les affichons par ordre décroissant.

3) **Département ayant le plus faible taux de valorisation matière et organique en 2013**

SELECT dep FROM environnement WHERE valorga2013 != 'nan' ORDER BY valorga2013 ASC LIMIT 1 ;

Explication : Nous sélectionnons les départements de la table Environnements où la valeur du taux de valorisation matière et organique en 2013 est renseignée (différente de "NaN") et nous les affichons par ordre croissant. Étant donné que nous voulons uniquement un seul département, nous limitons l'affichage à une valeur (LIMIT 1).

4) **Part (en %) de l'agriculture biologique dans la surface agricole totale du département contenant le plus grand pourcentage de population éloignée de plus de 7 minutes des services de santé de proximité**

```
SELECT E1.dep, agribio2016, disSanteSup7 FROM environnement E1 JOIN departementsinfos D1 ON E1.dep = D1.dep WHERE agribio2016 != 'nan' and disSanteSup7 != 'nan' ORDER BY disSanteSup7 DESC LIMIT 1 ;
```

Explication : Nous sélectionnons tous les départements, la part de l'agriculture bio en 2016 et le pourcentage de personnes dont la durée du trajet vers les services de santé est supérieure à 7 minutes. Pour cela, nous joignons les tables Environnement et DepartementsInfos (jointure sur le nom des départements). Ensuite, nous sélectionnons les départements où les deux valeurs sont renseignées (différentes de "NaN") et nous les trions de manière décroissante par pourcentage de personnes dont la durée de trajet vers les services de santé est supérieure à 7 minutes. Puis, comme précédemment, nous ne gardons qu'une valeur.

5) **Taux de pauvreté en 2014 des régions dont la part des jeunes non insérés est supérieure à 30% en 2014**

```
SELECT nccenr, pauvrete FROM regionsinfos WHERE pauvrete != 'nan' and jeunesnoninseres2014 > 30 ;
```

Explication : Nous sélectionnons les départements et les valeurs du taux de pauvreté de la table RegionsInfos. Nous sélectionnons les régions (nccenr) dont le taux de pauvreté est renseigné (différent de "NaN") et dont le pourcentage de jeunes non insérés en 2014 est supérieure à 30 %.

6) **Poids de l'économie sociale dans les emplois salariés de la région dont la source de la puissance électrique en énergies renouvelables provenait à au moins 10% de l'énergie photovoltaïque et dont la part de l'agriculture biologique dans la surface agricole totale était d'au moins 5% en 2015**

```
SELECT R1.nccenr, R2.poidsecosoc FROM environnement E1 JOIN departements D1 ON D1.nccenr = E1.dep JOIN regions R1 ON R1.reg = D1.reg JOIN regionsinfos R2 ON R1.nccenr = R2.nccenr GROUP BY R1.nccenr, R2.poidsecosoc HAVING (SUM(E1.photovoltaique2015)/COUNT(R2.nccenr)) > 10 AND (SUM(E1.agribio2016)/COUNT(R2.nccenr)) > 5 ORDER BY R2.poidsecosoc DESC ;
```

Explication : Ici, nous réalisons une jointure entre trois tables : Environnement et Départements (jointure sur le nom du département) puis nous y joignons la table RegionsInfos (sur le numéro de la région associé au département, comme pour la question 1 de la partie 2). Nous sélectionnons les départements dont les régions possèdent des valeurs moyenne d'utilisation du photovoltaïque en 2015 supérieure à 10 % et dont la part d'agriculture bio en 2016 est supérieure à 5 %. Nous affichons finalement le poids économique et sociale de la région (obtenue grâce au group by, classés de manière décroissante).

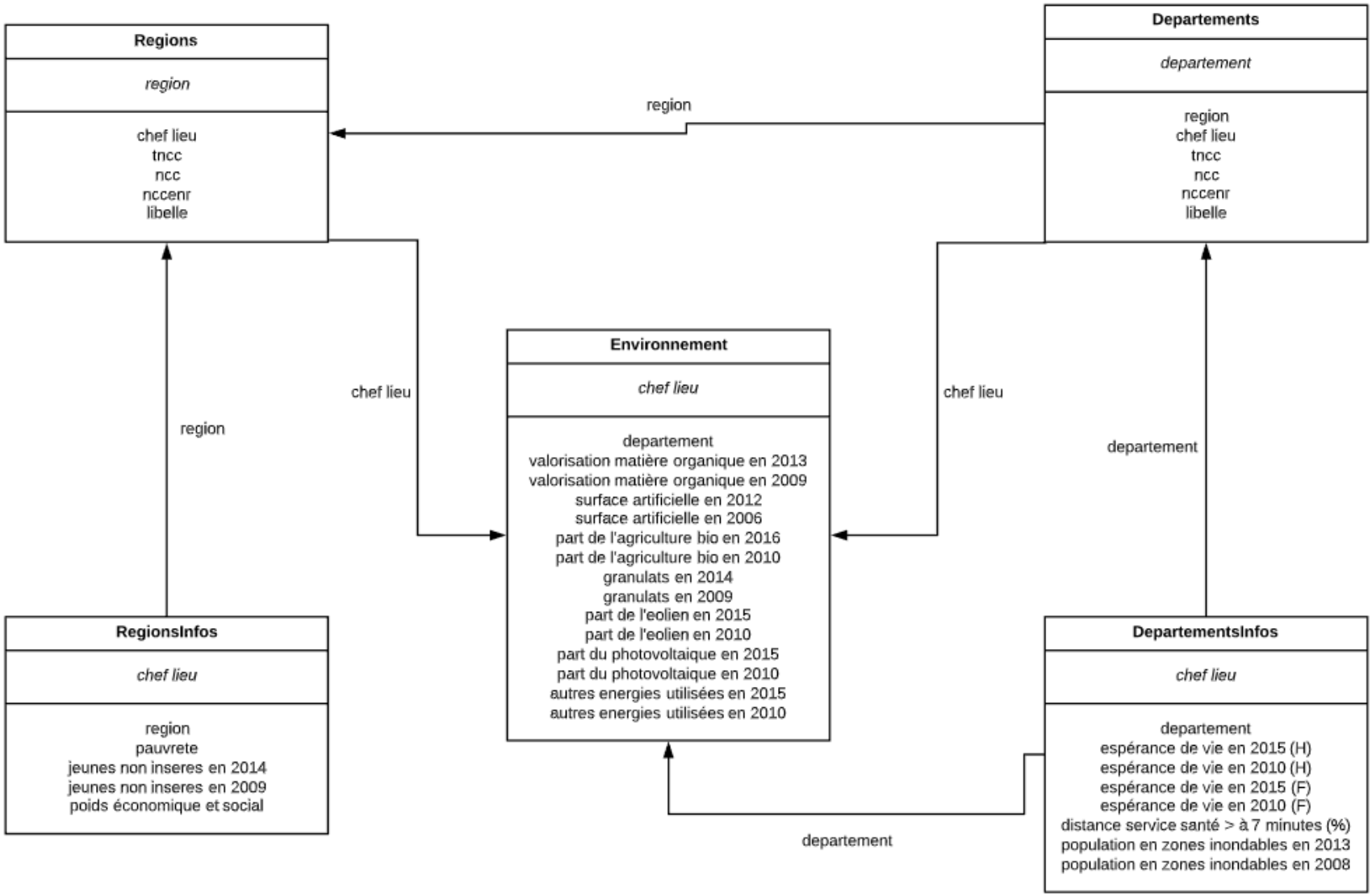
Partie 3

Discussions et remarques

En ce qui concerne la saisie des régions et départements par l'utilisateur, j'ai par défaut sélectionné le format "minuscule" (*exemple : Nouvelle-Aquitaine*). Ainsi, si l'utilisateur entre une région correcte en majuscule, même si elle existe, il y'a retour au menu. Notez qu'il faut mettre la première lettre en majuscule. On peut copier-coller un nom depuis la liste des régions pour se faciliter la tâche (j'affiche aussi aux questions 1 et 2 de la partie 1 tous les éléments en "minuscules")

Un autre problème que j'ai rencontré concerne le numéro des départements, qui sont pour la plupart uniquement des chiffres, a été de gérer les numéros pour la Corse. J'ai donc mis comme type de variable varchar.

Annexes


Regions
region
chef lieu
tncc
ncc
nccenr
libelle
Departements
departement
region
chef lieu
tncc
ncc
nccenr
libelle
region
Environnement
chef lieu
departement
valorisation matière organique en 2013
valorisation matière organique en 2009
surface artificielle en 2012
surface artificielle en 2006
part de l'agriculture bio en 2016
part de l'agriculture bio en 2010
granulats en 2014
granulats en 2009
part de l'eolien en 2015
part de l'eolien en 2010
part du photovoltaique en 2015
part du photovoltaique en 2010
autres energies utilisées en 2015
autres energies utilisées en 2010
chef lieu
chef lieu
region
departement
RegionsInfos
chef lieu
region
pauvrete
jeunes non inseres en 2014
jeunes non inseres en 2009
poids économique et social
DepartementsInfos
chef lieu
departement
espérance de vie en 2015 (H)
espérance de vie en 2010 (H)
espérance de vie en 2015 (F)
espérance de vie en 2010 (F)
distance service santé > à 7 minutes (%)
population en zones inondables en 2013
population en zones inondables en 2008
departement