# BUT1 Semestre 2 - SAé 2.04 - Exploitation d'une base de données

Florine LEFEBVRE
Charlie DARQUES
Groupe C

Année universitaire 2023-2024

# Exercice 1 : Comprendre les données

## Q1. Analyse du fichier récupéré

NB : le fichier *fr-esr-parcoursup.csv* a été déplacé dans un répertoire dédié à la SAé et renommé en tant que *data.csv*

**1. Combien y-a t-il de lignes ? Justifiez !**
La commande `cat data.csv | wc -l` nous donne 13870, si on enlève l'en-tête ça nous donne 13869 lignes.

**2. Que représente une ligne ?**
Chaque ligne représente une formation renseignée sur Parcoursup.

**3. Combien y-a t-il de colonnes ? Justifiez !**
On fait la commande `head data.csv -n 1 | grep -o ";" | wc -l`, qui nous donne le nombre de séparateurs de colonnes. La commande renvoie 117, on ajoute 1 car il y a une colonne de plus que de séparateurs, donc il y a 118 colonnes.

**4. Quelle colonne identifie un établissement ? (numéro et nom de col)**
La troisième colonne indique l'en-tête "Code UAI de l'établissement", ce qui est un numéro d'identifiant pour chaque établissement.

**5. Quelle colonne identifie une formation ? (numéro et nom de col)**
La 110e colonne dont le nom est "cod_aff_form" indique le numéro d'identifiant de chaque formation.

**6. Combien de lignes font référence à notre BUT Informatique ?**
Le code de l'iut est 0597215X, quand on la recherche dans le document il y a une ligne qui y fait référence, la ligne 6982.

**7. Quelle colonne identifie un département ? (numéro et nom)**
La cinquième colonne ("code départemental de l'établissement") identifie chaque département.

**8. Comment envisagez vous importer ces données ?**
Nous pensons écrire un script SQL pour créer une table de 118 colonnes (pour la création de colonnes, on s'aide de Java pour écrire plus rapidement les déclarations avec une boucle). On créera une table *import* pour stocker les données du fichier CSV et on importera petit à petit les données importées dans la nouvelle table pour la structurer.

**9. Quels problèmes identifiez vous dans ces données initiales ? (il y en a surement plusieurs, expliquez les clairement)**
Le fichier est très long, notamment à cause de la redondance qui est dûe au fait que les colonnes sont calculables : c'est-à-dire qu'on a pas besoin de les écrire pour trouver leurs valeurs, donc elles ne sont pas essentielles à la table. De plus, plusieurs colonnes calculent

la même chose mais avec une unité différente (exemple : en pourcentage ou par effectif), et cela crée de la redondance.

## Q2. Importer les données

Les noms des colonnes étant particulièrement complexes, on fera en sorte que l'ensemble des colonnes soit renommé à partir de n1.

**1. Fournir un fichier dico.xls permettant la correspondance entre les numéros de colonnes et les noms du fichier initial. Expliquez comment vous vous y êtes pris pour le constituer.**
Dans un terminal depuis le répertoire de travail, on exécute la commande '`head data.csv -n1 | tr ";" "\n" | nl -i 1 | tr "\t" ";" > dico.xls`'. La première commande ne garde que l'en-tête, la deuxième remplace les ";" par des retours à la ligne pour avoir le résultat sous forme de liste, la troisième ajoute les numéros de ligne pour se référencer à la colonne correspondante, la quatrième rajoute des ";" pour respecter le format CSV.

**2. Créer une table import permettant l'importation de ces données (fournir le code)**
On s'aide du résultat d'une boucle Java pour lister toutes les colonnes à créer (cf. TableImport.java dans l'archive) :

```
class TableImport{
	public static void main(String[] args) {
	String s = "";
	for (int j = 1; j < 119; j++) {
		s+="n" + j + " text, ";
	}
	System.out.println(s);
	}
}
```

Ce qui donne, après le copié-collé du résultat dans la requête SQL :

```
CREATE TABLE import (
	n1 text, n2 text, n3 text, n4 text, n5 text, n6 text,
	n7 text, n8 text, n9 text, n10 text, n11 text, n12 text,
	n13 text, n14 text, n15 text, n16 text, n17 text, n18 text,
	n19 text, n20 text, n21 text, n22 text, n23 text, n24 text,
	n25 text, n26 text, n27 text, n28 text, n29 text, n30 text,
	n31 text, n32 text, n33 text, n34 text, n35 text, n36 text,
	n37 text, n38 text, n39 text, n40 text, n41 text, n42 text,
	n43 text, n44 text, n45 text, n46 text, n47 text, n48 text,
	n49 text, n50 text, n51 text, n52 text, n53 text, n54 text,
	n55 text, n56 text, n57 text, n58 text, n59 text, n60 text,
	n61 text, n62 text, n63 text, n64 text, n65 text, n66 text,
	n67 text, n68 text, n69 text, n70 text, n71 text, n72 text,
	n73 text, n74 text, n75 text, n76 text, n77 text, n78 text,
```

```
      n79 text, n80 text, n81 text, n82 text, n83 text, n84 text,
      n85 text, n86 text, n87 text, n88 text, n89 text, n90 text,
      n91 text, n92 text, n93 text, n94 text, n95 text, n96 text,
      n97 text, n98 text, n99 text, n100 text, n101 text, n102
      text, n103 text, n104 text, n105 text, n106 text, n107 text,
      n108 text, n109 text, n110 text, n111 text, n112 text, n113
      text, n114 text, n115 text, n116 text, n117 text, n118 text
);
```

**3. S'assurer que les types de colonnes soient les plus restrictifs possibles (des int pour les colonnes contenant des entiers, des char(x) pour les données textuelles de taille x etc ...)**

Pour que les types des colonnes soient plus restrictifs, on modifie chaque colonne d'un type approprié et on obtient :

```
CREATE TABLE import (
      n1 SMALLINT, n2 VARCHAR(40), n3 CHAR(8), n4 VARCHAR(200), n5 CHAR(3),
      n6 VARCHAR(30), n7 VARCHAR(30), n8 VARCHAR(30), n9 VARCHAR(30),
      n10 TEXT, n11 CHAR(30), n12 VARCHAR(20), n13 TEXT, n14 VARCHAR(100),
      n15 VARCHAR(200), n16 VARCHAR(300), n17 CHAR(40), n18 SMALLINT,
      n19 SMALLINT, n20 SMALLINT, n21 SMALLINT, n22 INT, n23 INT,
      n24 SMALLINT, n25 SMALLINT, n26 SMALLINT, n27 SMALLINT, n28 SMALLINT,
      n29 SMALLINT, n30 SMALLINT, n31 SMALLINT, n32 SMALLINT, n33 SMALLINT,
      n34 SMALLINT, n35 SMALLINT, n36 SMALLINT, n37 SMALLINT, n38 SMALLINT,
      n39 SMALLINT, n40 SMALLINT, n41 SMALLINT, n42 SMALLINT, n43 SMALLINT,
      n44 SMALLINT, n45 SMALLINT, n46 SMALLINT, n47 SMALLINT, n48 SMALLINT,
      n49 SMALLINT, n50 SMALLINT, n51 NUMERIC(5,1), n52 NUMERIC(5,1),
      n53 NUMERIC(5,1), n54 SMALLINT, n55 SMALLINT, n56 SMALLINT,
      n57 SMALLINT, n58 SMALLINT, n59 SMALLINT, n60 SMALLINT, n61 SMALLINT,
      n62 SMALLINT, n63 SMALLINT, n64 SMALLINT, n65 SMALLINT,
      n66 NUMERIC(5,1), n67 SMALLINT, n68 SMALLINT, n69 SMALLINT,
      n70 SMALLINT, n71 SMALLINT, n72 SMALLINT, n73 SMALLINT,
      n74 NUMERIC(4,1), n75 NUMERIC(4,1), n76 NUMERIC(4,1),
      n77 NUMERIC(4,1), n78 NUMERIC(4,1), n79 NUMERIC(4,1),
      n80 NUMERIC(4,1), n81 NUMERIC(4,1), n82 NUMERIC(4,1),
      n83 NUMERIC(4,1), n84 NUMERIC(4,1), n85 NUMERIC(4,1),
      n86 NUMERIC(4,1), n87 NUMERIC(4,1), n88 NUMERIC(4,1),
      n89 NUMERIC(4,1), n90 NUMERIC(4,1), n91 NUMERIC(4,1),
      n92 NUMERIC(4,1), n93 NUMERIC(4,1), n94 NUMERIC(4,1),
      n95 NUMERIC(5,1), n96 NUMERIC(5,1), n97 NUMERIC(5,1),
      n98 NUMERIC(5,1), n99 NUMERIC(5,1), n100 NUMERIC(5,1),
      n101 NUMERIC(5,1), n102 VARCHAR(40), n103 NUMERIC(6,1), n104
      CHAR(40), n105 SMALLINT, n106 CHAR(40), n107 SMALLINT, n108
      VARCHAR(45), n109 VARCHAR(25), n110 INT, n111 VARCHAR(100), n112
      CHAR(100), n113 NUMERIC(4,1), n114 NUMERIC(4,1), n115 NUMERIC(4,1),
      n116 NUMERIC(4,1), n117 CHAR(5), n118 CHAR(5)
);
```

**4. Remplir cette table avec les données récupérées (fournir le code)**

D'abord on supprime la première ligne étant donné qu'elle ne contient que les entêtes des colonnes. On utilise donc la commande : `sed 1d data.csv > data1.csv`

Le nouveau fichier est donc data1.csv.  Pour importer les données dans la table, on utilise la commande : `\copy import from data1.csv CSV DELIMITER ';'`

**5. En s'appuyant sur la table import fournir les requêtes et les réponses qui permettent de savoir :**

**(a) Combien il y a de formations gérés par Parcoursup ?**

```
SELECT COUNT(*) FROM import;
```

=> 13869

**(b) Combien il y a d'établissements gérés par Parcoursup ?**

```
SELECT COUNT(DISTINCT n4) from import;
```

=> 3602

**(c) Combien il y a de formations pour l'université de Lille ?**

```
SELECT COUNT(*) FROM import WHERE n4 = 'Université de Lille';
```

=> 124

**(d) Combien il y a de formations pour notre IUT ?**

```
SELECT COUNT(*) FROM import WHERE n3 = '0597215X';
```

=> 10

**(e) Quel est le code du BUT Informatique de l'université de Lille ?**

```
SELECT n110 FROM import WHERE n3 = '0597215X' AND n10 = 'BUT - Informatique';
```

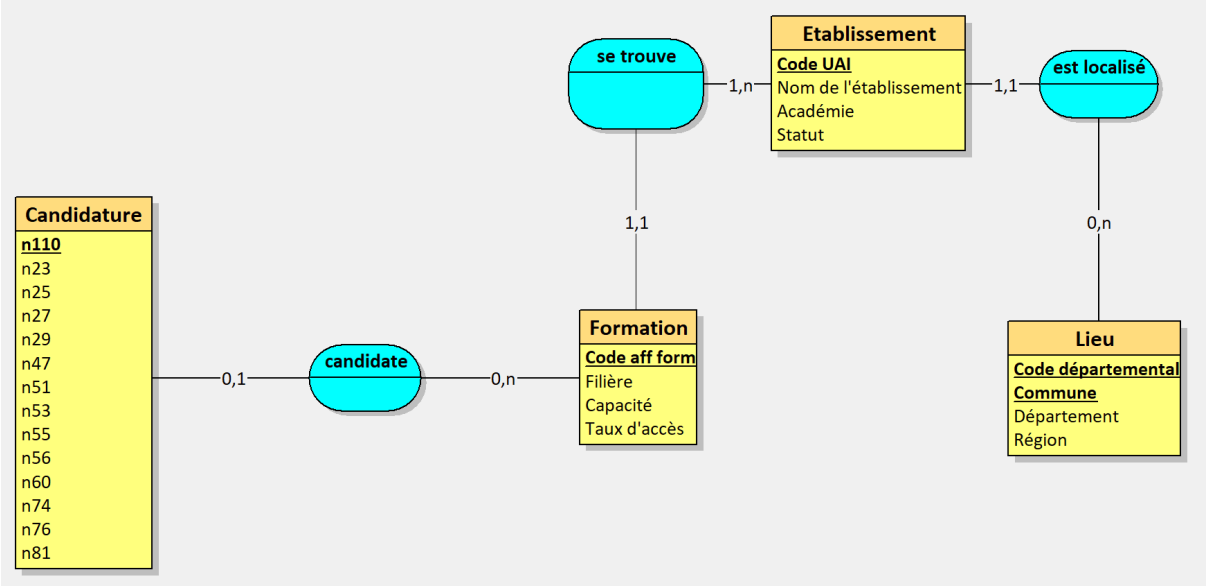
=> 6888

**(f) Citez 5 colonnes contenant des valeurs nulles**

# Exercice 2 : Ventiler les données

## Q1. Normalisation des données

**1. Fournir le MCD correspondant à la structuration**




**2. Ecrire le script `parcourssup.sql` qui permet de réaliser toutes les actions d'importation et de création/remplissage des différentes formations parcourssup.**
cf. archive rendue sur Moodle

## Q2. Une question de taille

**1. Quelle taille en octet fait le fichier récupéré ?**
On utilise la commande `wc -c parcoursup.sql` pour connaître le nombre d'octets : la réponse est 5181.

**2. Quelle taille en octet fait la table import ?**
Sur PSQL, on exécute la requête `SELECT pg_relation_size('import');` qui affiche la taille de la table import en octets : la réponse est 13 647 872.

**3. Quelle taille en octet fait la somme des tables créées ?**
Sur PSQL, on exécute la requête `SELECT pg_relation_size('lieu') + pg_relation_size('candidature') + pg_relation_size('formation') + pg_relation_size('etablissement');` : la réponse est 2 580 480.

**4. Quelle taille en octet fait la somme des tailles des fichiers exportés correspondant à ces tables ?**
On copie toutes les tables créées dans des fichiers CSV avec la commande `\copy <table> TO <nom_fichier> DELIMITER ';'`

On additionne ensuite les résultats de chaque commande `wc -c <nom_fichier>` : la réponse est 2 087 201 octets pour la somme des fichiers exportés.

```
charlie.darques.etu@epicea18:~/Bureau/SAéBDD$ wc -c lieu.csv
56926 lieu.csv
charlie.darques.etu@epicea18:~/Bureau/SAéBDD$ wc -c formation.csv
1045846 formation.csv
charlie.darques.etu@epicea18:~/Bureau/SAéBDD$ wc -c etablissement.csv
251392 etablissement.csv
charlie.darques.etu@epicea18:~/Bureau/SAéBDD$ wc -c candidature.csv
733037 candidature.csv
```

## Exercice 3 : Requêtage

cf. requetes.sql dans l'archive.