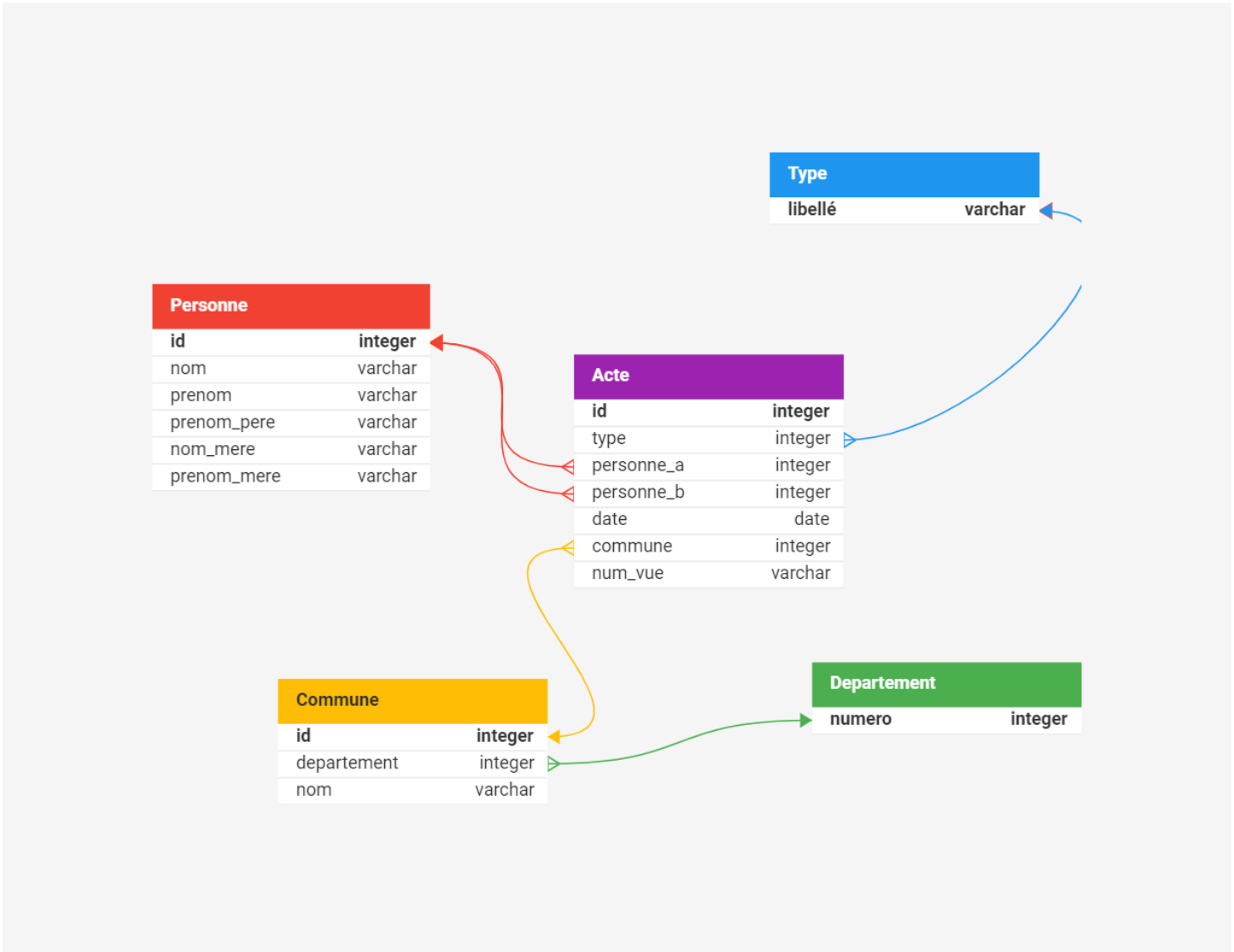
Projet modélisation de bases de données

Arthur Gillier - Florian Chacun

Introduction

Ce projet a été réalisé dans le cadre du cours de Modélisation de Bases de données de l'Université de La Rochelle. Il a pour but de mettre en place une base de données relationnelle à partir de données issues de fichiers d'actes de mariage. Il faudra les nettoyer, les parser et les insérer de manière optimale dans une base de données PostgreSQL.

Schéma de la base de données



Notez que les tables `departement` et `type` ne sont pas réellement des tables mais des enums.

Création des tables

Département

```
CREATE TYPE departement AS ENUM ('44', '49', '79', '85');
```

Type

```
CREATE TYPE type_acte AS ENUM ('Certificat de mariage', 'Contrat de mariage', 'Divorce', 'Mari
```

Commune

```
CREATE TABLE IF NOT EXISTS commune(
    id INT PRIMARY KEY,
    nom VARCHAR(255) NOT NULL,
    departement departement NOT NULL
)
```

Personne

```
CREATE TABLE IF NOT EXISTS personne (
    id INT PRIMARY KEY,
    nom VARCHAR(255),
    prenom VARCHAR(255),
    prenom_pere VARCHAR(255),
    nom_mere VARCHAR(255),
    prenom_mere VARCHAR(255)
);
```

Acte

```
CREATE TABLE acte(
    id INT PRIMARY KEY,
    type_id type_acte NOT NULL,
    personne_a INT,
    personne_b INT,
    commune INT,
    date_ TIMESTAMP WITH TIME ZONE,
    num_vue VARCHAR(255) DEFAULT 'null'
)
```

Découpage des données

En suivant la méthode donnée nanani nanana

Découpage du csv :

```
# Découpage du fichier mariage_L3_5k.csv pour obtenir les communes et les personnes.
cut -f13,14 -d ',' mariages_L3_5k.csv | sort | uniq > commune.csv
cut -f3,4,5,6,7 -d ',' mariages_L3_5k.csv | sort | uniq > personnes.csv

# Notons que le >> permet de rajouter les données à la fin du fichier sans écraser le contenu
cut -f8,9,10,11,12 -d ',' mariages_L3_5k.csv | sort | uniq >> personnes.csv

# Ajout des id pour chaque ligne des csv de personne et commune.
awk -F, '{print NR","$0}' personne.csv > personne_id.csv
awk -F, '{print NR","$0}' commune.csv > commune_id.csv
```

Récupération des actes

Afin de récupérer les actes nous avons utiliser un script python qui utilise les fichiers `mariages_L3_5k.csv`, `personne_id.csv` et `commune_id.csv` pour générer un fichier `actes.csv` qui contient les actes de mariage.

```
import pandas as pd

# Charger le fichier personnes.csv dans un DataFrame
personnes_df = pd.read_csv('/media/Qi/agillier/L3/Projet-Modélisation/data/personne_id.csv', h
commune_df = pd.read_csv('/media/Qi/agillier/L3/Projet-Modélisation/data/commune_id.csv', head

# Fonction pour rechercher le numéro dans personnes.csv
def trouver_id_personne(nom, prenom, prenom_pere, nom_mere, prenom_mere):
    filtre = (personnes_df['nom'] == nom) & (personnes_df['prenom'] == prenom) & (personnes_df
    resultats = personnes_df[filtre]
    if not resultats.empty:
        return resultats['id'].values[0]
    else:
        return None

def trouver_id_commune(nom, departement):
    filtre = (commune_df['nom'] == nom) & (commune_df['departement'].astype(str) == str(depart
    resultats = commune_df[filtre]
    if not resultats.empty:
        return resultats['id'].values[0]
    else:
        return None


# Ouvrir un fichier de sortie CSV pour écrire les résultats
with open('/media/Qi/agillier/L3/Projet-Modélisation/data/mariages_L3_5k.csv', 'r') as f:
```

```
    with open('/media/Qi/agillier/L3/Projet-Modélisation/data/actes.csv', 'w') as output_file:
        output_file.write("Identifiant d’acte,Type d’acte,Id Personne A,Id Personne B,Commune,
        for line in f:
            mariage_info = line.strip().split(',')
            if len(mariage_info) == 16:  # Assurez-vous que la ligne contient suffisamment de
                nom_personne_a = mariage_info[2]
                prenom_personne_a = mariage_info[3]
                prenom_pere_personne_a = mariage_info[4]
                nom_mere_personne_a = mariage_info[5]
                prenom_mere_personne_a = mariage_info[6]
                id_personne_a = trouver_id_personne(nom_personne_a, prenom_personne_a, prenom_
                nom_personne_b = mariage_info[7]
                prenom_personne_b = mariage_info[8]
                prenom_pere_personne_b = mariage_info[9]
                nom_mere_personne_b = mariage_info[10]
                prenom_mere_personne_b = mariage_info[11]
                id_personne_b = trouver_id_personne(nom_personne_b, prenom_personne_b, prenom_
                commune = trouver_id_commune(mariage_info[12], mariage_info[13])
                temps = mariage_info[14].split('/')
                if len(temps) == 3:
                    mariage_info[14] = temps[2] + '-' + temps[1] + '-' + temps[0]
                else:
                    mariage_info[14] = ''
                if (id_personne_a is not None and id_personne_b is not None):
                    output_file.write(f"{mariage_info[0]},{mariage_info[1]},{id_personne_a},{i
```

Insertion des données dans la base de données :

```
COPY personne FROM 'C:\Program Files\PostgreSQL\16\mariages\personne_id.csv' DELIMITER ',' CSV

/* Notons que l'on utilise le CSV HEADER afin d'éliminer le header du csv. */
COPY acte (id,type_id,personne_a,personne_b,commune,date_,num_vue) FROM 'C:\Program Files\Post

COPY commune (id,nom,departement) FROM 'C:\Program Files\PostgreSQL\16\mariages\commune_id.csv
```

Ajout des relations

Acte -> Commune

```
ALTER TABLE acte ADD CONSTRAINT acte_fk3 FOREIGN KEY (commune) REFERENCES commune(id);
```

Acte -> Personne (a) & (b)

```
/* Acte -> Personne (a) */
ALTER TABLE acte ADD CONSTRAINT acte_fk1 FOREIGN KEY (personne_a) REFERENCES personne(id);
```

```
/* Acte -> Personne (b) */
ALTER TABLE acte ADD CONSTRAINT acte_fk2 FOREIGN KEY (personne_b) REFERENCES personne(id);
```

Requêtes

La quantité de communes par département

```
SELECT departement, COUNT(*) AS nombre_de_communes
FROM commune
GROUP BY departement;
```

Résultat :

	departement departement	nombre_de_communes bigint
1	79	51
2	49	2
3	85	313
4	44	9

La quantité d'actes à LUÇON

```
SELECT COUNT(*) AS nombre_d_actes
FROM acte
INNER JOIN commune ON acte.commune = commune.id
WHERE commune.nom = 'LUÇON';
```

Résultat : `105`

La quantité de “contrats de mariage” avant 1855

```
SELECT COUNT(*) AS nombre_de_contrats_de_mariage
FROM acte
WHERE type_id = 'Contrat de mariage' AND date_ < '1855-01-01';
```

Résultat : `196`

La commune avec la plus quantité de “publications de mariage”

```
SELECT commune.nom AS commune,
       COUNT(*) AS nombre_de_publications_de_mariage
FROM acte
```

```
INNER JOIN commune ON acte.commune = commune.id
WHERE acte.type_id = 'Publication de mariage'
GROUP BY commune.nom
ORDER BY COUNT(*) DESC
LIMIT 1;
```

Résultat : `SAINT PIERRE DU CHEMIN : 20`

La date du premier acte et le dernier acte

```
SELECT MIN(date_) AS premiere_date_acte,
       MAX(date_) AS derniere_date_acte
FROM acte;
```

Résultat : `Première date : "1581-12-23 00:00:00+00:09:21"` & `Dernière date : "1915-09-14 00:00:00+00"`

Conclusion

Tout est fonctionnel pour le fichier de 5k lignes, pour passer sur le fichier à 500k lignes il faudrait ajouter une table département et une table type d'actes. Il faudrait également modifier le script python pour qu'il prenne en compte ces nouvelles tables.

Auteurs

- Arthur Gillier
- Florian Chacun
- Sujet : Université de La Rochelle.