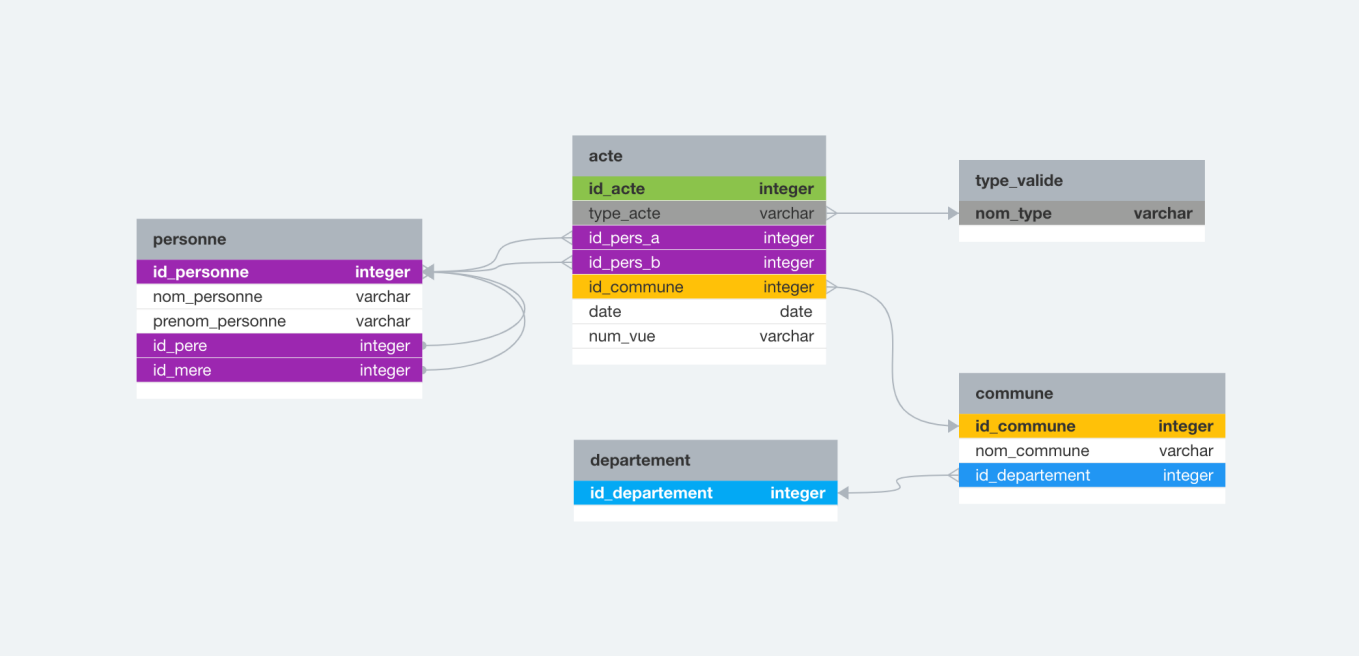
Projet Modélisation bases de données

Binôme : FONTENOY Anya & PUGH Joshua

Création et modélisation de la base de données

Pour notre projet, nous avons réalisé schéma de modélisation pour mettre au point notre base de données :



Ci-dessus, nous avons 5 tables :

- acte
- type_Valide
- personne
- commune
- departement

Table Personne

Chaque personne a la possibilité d'avoir un **nom**, un **prénom** mais aussi un **père** et une **mère** (qui sont eux même des personnes ayant les mêmes attributs).

-> Toutes ces personnes sont identifiables par un identifiant unique.

Table Type Valide et Table Departement

Ces deux tables sont très simples et ne contiennent qu'**un seul attribut, clé unique.**

Table Commune

Chaque commune possède un seul et unique département associé, mais un département peut posséder plusieurs communes.

Table Acte

Cette table est composée d'un **identifiant unique**, d'un **type d'acte** (reconnu par son identifiant nom), d'une **personne A** (reconnue par son id), d'une **personne B** (reconnue par son id), d'une **commune** (reconnue par son id), une **date**, et un **num vue**.

> ***Grâce à ces relations, nous sommes capables d'obtenir toutes les informations que nous voulons à partir de n'importe quel table.***

Par exemple :

Nous voulons obtenir le prénom du père de la personne A à partir de l'acte de mariage.

> Pour cela, nous récupérons l'identifiant de la personne A, puis nous récupérons l'identifiant de son père, puis nous récupérons le prénom.

id	type	id_pers_a	id_pers_b	id_commune	date	numvue
4	"Mariage"	8	11	3	"1844-09-11"	"329/455"

-> Ici l'identifiant de la personne A est 8, donc nous cherchons la personne avec l'identifiant 8.

id	nom	prenom	id_pere	id_mere
8	"ABADIE"	"Jean Auguste"	6	7

-> Ici l'identifiant du père est 6, donc nous cherchons la personne avec l'identifiant 6.

id | nom | prenom | id_pere | id_mere |:-: |:-: |:-: |:-: 6 | "ABADIE" | "Jean François" | NULL | NULL

Maintenant, il n'y a plus qu'à récupérer le prénom.

Compréhension et découpage du fichier CSV

Pour découper le fichier CSV, nous avons fait un programme python qui permet d'insérer les informations.

Pour cela, nous nous connectons à notre base de donnée avec le login et le mot de passe nécéssaire et créons un curseur qui va pouvoir exécuter des requetes à la base de données :

```
conn = psycopg2.connect("dbname=projet_modelisation user=postgres
password=motdepasse")
```

Puis, nous ouvrons le fichier voulu pour le parcourir ligne par ligne et récupérer les données :

```
with open('mariages\mariages_L3_5K.csv', 'r', encoding='utf-8') as f:
    reader = list(csv.reader(f))
    row_count = len(reader)
```

Nous pouvons à présent insérer des données comme celles présente dans la table type_acte et departement :

```
cur.execute(
        "INSERT INTO departement (id_departement) VALUES (%s),(%s),(%s),(%s)",
(44,49,79,85)
    )
```

La suite est relativement simple car nous récupérons les données une à une pour les tester puis les enregistrer dans la base de données. Par exemple, pour insérer les communes, voici le fonctionnement :

```
test = "="

if row[13] == "n/a" or row[13] not in ["44", "49", "79", "85"]:

    row[13] = None
    test = "IS"

cur.execute(
    """
    INSERT INTO commune (nom_commune, id_departement)
    SELECT %s, %s
    WHERE NOT EXISTS (SELECT 1 FROM commune WHERE nom_commune = %s AND
id_departement """+test+""" %s);
    """, (row[12], row[13], row[12], row[13])
)
```

Ci-dessus, nous avons récupéré la valeur de la colone 13 qui est associée à la commune. Nous vérifions si elle n'est pas `n/a` ou hors département valide. Si c'est le cas, nous mettons le résulat en `None` et la variable test en `IS` ce qui va nous servir après à insérer les données.

Précision

Comme il était précisé, lorque deux personnes ont le même nom et le même prénom, elles sont une même personne. Lors de l'insertions des données classique, si l'un des parents est également l'enfant, nous décidons donc de placer la personne en temps qu'enfant et de mettre le parents en NULL.

-> Cela nous évite les problèmes de sélection lorsque deux personnes ont le même nom.

Choix et changements dus aux difficultés

Lors du traitement du grands fichier 'mariages_L3' nous avons dû changer quelques règles qui sont les suivantes :

- Beaucoup des actes contenaient des **dates non valides**. Soit par manque de jour ou de mois (00 ou ?? à la place), soit par cause de **formats qui ne sont pas bon** (par exemple des intervals). Pour simplifier le problème, nous avons décidé de ne pas traiter les cas particuliers et de les stocker dans un ficher

"skipped.json" mis à disposition dans l'arborescence, en plus d'enregistrer tout le même l'acte en passant la date en `NULL`.

Amélioration

Nous avons fait le choix de garder les données pour permettre à l'utilisateur de pouvoir **revenir sur les données non valides** pour les insérer avec une date valide. Mais nous pouvons aussi imaginer la perspective de **modifier notre base de données** pour pouvoir permettre l'insertion à l'aide d'intervales.

- Certains actes avaient des départements **n'appartenant pas au registre autorisé** ou alors étant enregistrés comme étant "n/a". Nous avons décidé ici de **basculer** tout les id de département non valides **en NULL** et de les insérer tout de même dans la base de données pour **évité de perdre les données**.

> Lorsque l'utilisateur voudra supprimer ces lignes ou ajouter des départements valide afin de reconnecter les communes et les départements, il pourra le faire de manière très simple à l'aide de requêtes SQL.

- Il y avait également beaucoup d'actes dont le type (type_acte) n'était pas renseigné dans les types valides. Là encore, nous avons fait le choix de stocker les données dans le fichier "skipped.json" mais cette fois-ci de ne pas les insérer dans la base de données et de laisser libre choix à l'utilisateur de revenir sur ces données plus tard.

Requêtes et réponses du sujet

1. La quantité de communes par département

On selectionne les quantités de communes `count(commune.nom_commune)` et leur départements `departement.id_departement` . On les rassemble par paquet de département `group by departement.id_departement` (et puis on trie par ordre croissant d'id pour être plus propre).

```sql
SELECT departement.id_departement,count(commune.nom_commune) FROM commune
JOIN departement ON commune.id_departement = departement.id_departement
GROUP BY departement.id_departement
ORDER BY departement.id_departement
```

Résultat petit fichier

N° département	Nombre de commmunes
44	9
49	2
79	51
85	313

Résultat grand fichier

N° département	Nombre de commmunes
44	21
49	25
85	69
79	353

---

2. La quantité de actes à LUÇON

On selectionne le nombre d'actes `COUNT(id_acte)` et on rajoute la contrainte qu'ils viennent de LUÇON `WHERE commune.nom_commune = 'LUÇON'`

```
SELECT COUNT(id_acte) FROM acte
JOIN commune ON acte.id_commune = commune.id_commune
WHERE commune.nom_commune = 'LUÇON';
```

Résulat petit fichier

Nombre d'actes à Luçon
105

Résulat grand fichier

Nombre d'acte à Luçon
9572

---

3. La quantité de “contrats de mariage” avant 1855

On selectionne le nombre de contrats de mariage (actes) `COUNT(id_acte)` et on vérifie que ce sont bien des contrats de mariage `WHERE type_acte = 'Contrat de mariage'` et que la date est inférieur à 1855 `date < '1855-01-01'`

```
SELECT COUNT(id_acte) FROM acte
WHERE type_acte = 'Contrat de mariage' AND date < '1855-01-01';
```

Résultat petit fichier

Mariage avant 1855

Mariage avant 1855
196

Résulat grand fichier

Nombre acte à Luçon
20601

---

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

On selectionne tout les nom et le nombre de communes `commune.nom_commune, COUNT(*)`. On fait une jointure avec la table acte pour obtenir les types d'actes, puis on met la contrainte pour ne prendre que les publications de mariage. On regroupe les actes par leur commune, puis pour obtenir le nombre le plus grand (DESC) on tri notre ensemble `ORDER BY COUNT(*) DESC` et on récupère le premier élément `LIMIT 1`.

```
SELECT commune.nom_commune, COUNT(*) FROM commune
JOIN acte ON commune.id_commune = acte.id_commune
WHERE acte.type_acte = 'Publication de mariage'
GROUP BY commune.nom_commune
ORDER BY COUNT(*) DESC
LIMIT 1;
```

Résultat petit fichier

Nom de commune	Quantité publications
"SAINT PIERRE DU CHEMIN"	20

Résulat grand fichier

Nom de commune	Quantité publications
"MONTOURNAIS"	1439

---

5. La date du premier acte et le dernier acte

On sélectionne la plus petite et la plus grande date

```
SELECT MIN(date), MAX(date) FROM acte;
```

Résultat petit fichier

Date la plus petite	Date la plus grande

Date la plus petite	Date la plus grande
"1581-12-23"	"1915-09-14"

Résulat grand fichier

Date la plus petite	Date la plus grande
"1535-06-30"	"1915-12-22"