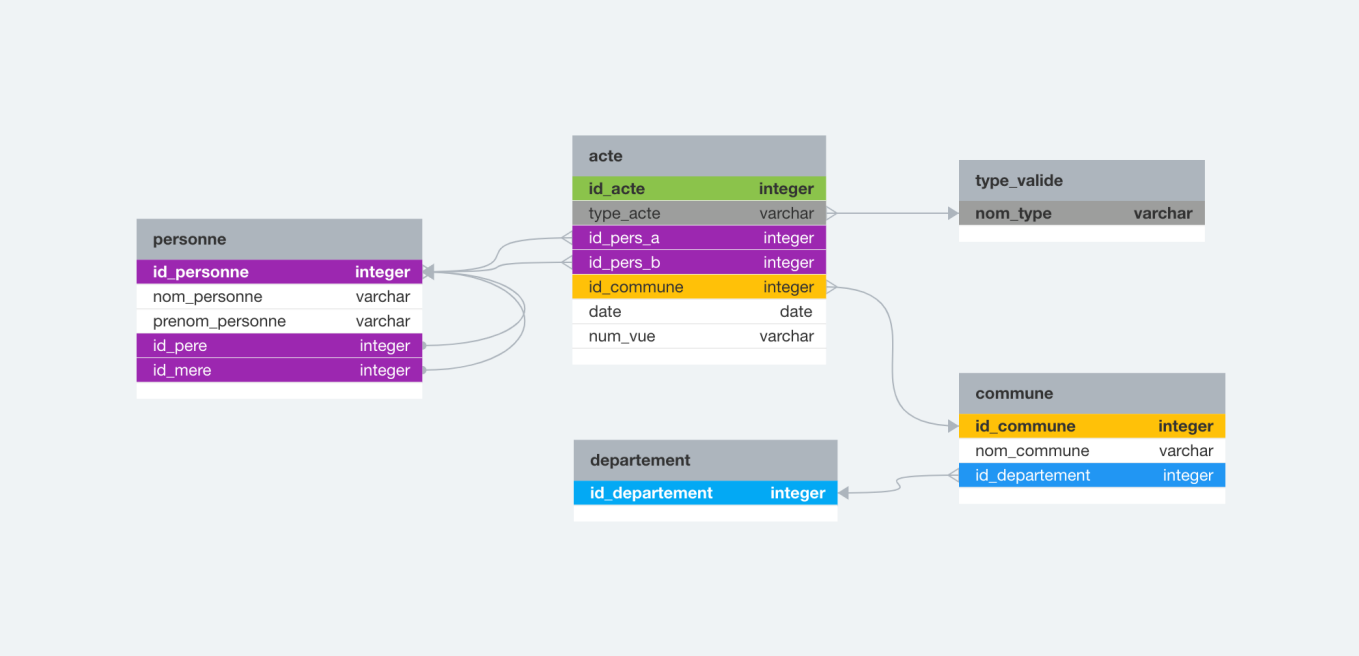
Projet Modelisation bases de données

Binome : FONTENOY Anya & PUGH Joshua

Création et modélisation de la base de donnée

Pour notre projet nous avons fait un shéma de modélisation pour mettre au point notre base de donnée :



Ci dessus nous avons 5 tables :

- acte
- type_Valide
- personne
- commune
- departement

Table Personne

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

-> toute ces personnes sont identifiable par un identifiant unique.

Table Type Valide et Table Departement

Ces deux tables sont très symple en ne contenant qu'**un seul attribut, clé unique.**

Table Commune

Chaque commune possède un seul et unique département associé mais un département peux possédé plusieurs communes.

Table Acte

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

> ***Grace à ces relations nous sommes capables d'obtenir toutes les informations que l'on veux à partir d'une table.***

Par exemple :

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

> Pour ça on récupère l'identifiant de la personne B, puis on récupère l'identifiant de son père, puis on récupère le prénom.

Compréhension et découpage du fichier CSV

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

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

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

Puis nous avons ouvert 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)
```

La suite est relativement simple car nous récupérons les données une à une pour les enregistrer dans la base de données par exemple pour insérer les départements:

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

Puisqu'il était préciser que lorque deux personnes ont le même nom et le même prénom ils sont une même personne, lors de l'insertions des données classique, si l'un des parents était également l'enfant nous décidons de placer la personne en temps que l'enfant et de mettre le parents en NULL.

-> Cela nous évite les problèmes de sélection lorsque deux personne d'une même famille ont le même nom.

Choix et changements

Lors du traitement du grands fichier 'mariages_L3_5K' nous avons dû changer quelques rêgles qui sont les suivantes :

- Beaucoup des actes contenais des **date non valide**. Soit par manque de jour ou de mois (00 à la place, soit ?? à la place, ou encore des **formats qui ne sont pas bon** par exemple des intervals). Pour simplifier le problème nous avons décidé de ne pas traiter les cas particulier et de les stocker dans un ficher "skipped.json" mis à disposition dans l'arborescence.

Amélioration

Nous avons fait le choix de garder les données pour permettre à l'utilisateur de pouvoir **revenir sur les données non valide** pour les inséré avec une date valide. Mais nous pouvons aussi imaginé la perspective de **modifié notre base de donnée** pour pouvoir permettre l'insertion a l'aide d'intervales pour ne pas avoir à modifier les données et de toute les mettre à 01/01/... .

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

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

Amélioration

Lorsque l'utilisateur voudra supprimer ces lignes où ajouté des départements valide afin de reconnecter les communes et les département il pourra le faire de manière très simple à l'aide de requête SQL.

Requêtes et réponses du sujet

1. La quantité de communes par département

On selectionne les quantitées 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)

```
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
85	313
79	51

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'acte `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 acte à Luçon
105

Résulat grand fichier

Nombre acte à Luçon
9572

---

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

On selectionne le nombre de contrats de mariage (acte) `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'`

// -> A vérifier avec le format de la date qui est peut être 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
196

Résulat grand fichier

Nombre acte à Luçon
20601

---

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

On selectionne tout le nom et le nombre de commune `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) ou le nombre le plus petit (ASC) 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 publication

Résulat grand fichier

Nom de commune	Quantité publication
"MONTOURNAIS"	1439

---

5. La date du premier acte et le dernier acte

On selectionne 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

Résulat grand fichier

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