PROJET ÉLECTION

Cours BDM 2023

Thibault NG & Camille DU

I - Organisation du groupe

Travail commun

- Conception de la base de données
- Écriture des requêtes SQL
- Écriture du rapport

Thibault

- Insertion et traitement des données
- Interface graphique (carte)

Camille

- Création de la base de données
- Interface graphique (tableau)

Outils de synchronisation

- Communication par appel Discord pour partage de code, d'écran, etc. Ce qui permet de travailler en simultané et d'échanger rapidement.
- Repo GitHub pour stocker l'ensemble des données manipulées lors de la réalisation du projet et s'assurer d'avoir les mêmes ressources :
    - Code python
    - Script SQL
    - Rapport
    - Diagramme
    - Fichier csv

II - Choix technologiques

Stockage des données

Pour stocker les données, nous avons choisis **Postgresql** comme outil de gestion de bases de données car c'est un outil que nous maîtrisons et qui est donc plus facile à manipuler. Nous avons fait le choix d'apporter la couche multi-dimensionnelle du côté code.

Alimentation de la base de données

Pour alimenter la base, nous avons choisi d'utiliser le langage de programmation **Python** pour la liberté d'offre le langage quant à la manipulation des données dans de simples scripts. Même si celui-ci est moins optimisé que d'autres langages de programmation, c'est la rapidité d'écriture qui a joué en sa faveur étant donné que la performance n'est pas un enjeu, vu que l'alimentation des données est une tâche qui ne se fait qu'une fois.

Pour traiter les données, nous avons utilisé les librairies **pandas** et **geopandas** qui donnent accès à des fonctions facilitant la manipulation de tableaux (DataFrame). Les fonctions qui ont été vraiment utiles dans ces librairies étaient les fonctions de jointure entre tableaux. Aussi, pandas traduit facilement les fichiers csv, ce qui a permis de faciliter l'importation des fichiers csv dans python avant leur insertion.

Au lieu de faire des requêtes directement depuis les scripts python jusqu'à la base, nous avons décidé de passer par l'intermédiaire de script SQL. Nos différents scripts pythons génèrent des scripts d'insertion SQL. Cela avait différents avantages :

- Avoir une sauvegarde de la base sous forme de script
- Pouvoir assurer la bonne insertion des données en examinant les scripts d'insertion directement (notamment pour la résolution de bugs)
- Éviter d'envoyer plein de requêtes depuis les différents script python, mais tout centraliser dans un seul script

Visualisation

En termes de support pour les visualisations nous avons choisis la librairie **Flask** qui permet d'insérer des données python dans des templates **HTML**. Ce qui permet de conserver la gestion des données dans python tout en ayant accès à une interface graphique classique.

Pour les requêtes, nous avons simplement utilisé la librairie **psycopg** qui est la librairie la plus populaire permettant de faire des requêtes SQL depuis python vers une base de données Postgresql.

Les graphiques ainsi que les cartes proviennent quant à eux de la librairie **plotly express** qui possède des classes permettant la création de figures complexes avec peu de paramètres. Cette librairie permet aussi d'exporter ces figures au format HTML, ce qui permet de les intégrer facilement à nos pages HTML avec Flask.

III - Modèle conceptuel des données

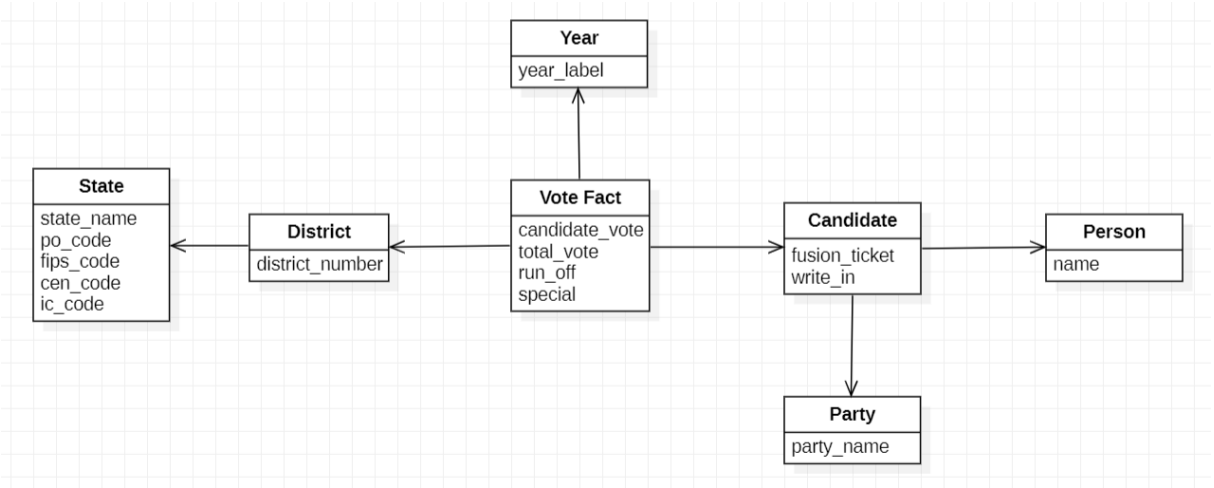

Schéma de la structure interne de l'entrepôt

Structure

Concernant la structure de l'entrepôt, nous avons décidé de prendre un modèle en flocon pour les avantages de la normalisation des dimensions.

Grain

La structure interne de l'entrepôt est d'abord composée d'une table de fait « vote_fact » dont le grain représente les résultats de vote d'un candidat dans un district lors d'une année, sachant qu'un district est situé dans un état et qu'un candidat représente une personne et son parti.

Table de fait

Vote Fact

- « candidate_vote » : nombre de vote pour ce candidat (pour un district et une année)

- « total_vote » : nombre de vote en tout sur cette élection (pour un district et une année)

- « run_off_election » qui signifie qu'il y a eu une élection supplémentaire, ce qui arrive lorsque aucun candidat lors d'une élection primaire ou générale n'a remporté la majorité absolue des votes

- « special_election » signifie que c'est une élection en dehors du calendrier électoral, ce qui arrive lorsqu'il y a des postes vacants pour diverses raisons comme la démission, la mort ou la destitution d'un élu en exercice.

Tables de dimensions

• **Year**

C'est l'année de l'élection

- « year_label » qui représente l'année, i.e. 1976, 1978, 1980

• **District**

La table « district » correspond à une zone géographique délimité et qui compose un état des Etats-Unis, aussi appelés « circonscription électorale » ou « circonscription législative » avec en attribut :

- « district_number » qui correspond à son numéro de district qui dépend de l'état, i.e. 001, 002, 008

• **State**

La table « state » qui correspond à une autre zone géographique englobant les 'districts', aussi appelés états, elle contient les attributs :

- « state_name » qui correspond au nom que porte l'état, i.e. Minnesota, Louisiane, Texas

- « po_code » qui correspond au code postal

- « fips_code » : code fips utilisé pour identifier de manière unique les États et les comtés aux États-Unis. C'est le code que l'on a le plus utilisé car il est présent dans le fichier csv fourni. Il est aussi présent comme attribut des états dans la carte et permet donc de faire la liaison.

- « cen_code » : code cen, utilisé pour identifier de manière unique les entités géographiques dans le contexte du recensement

- « ic_code » ; code ic, utilisé pour identifier les zones de communication interurbaine dans un système de numérotation téléphonique

• **Candidate**

La table « candidate » qui correspond à une personne représentant un parti pour une ou des élections avec en attribut :

- « fusion_ticket » qui correspond au fait que le candidat représente plusieurs partis ou non
- « write_in » : Booléen indiquant si le candidat est write-in. Bien que le choix était au début de porter l'information dans le parti avec un parti « WRITE-IN », on est revenu sur l'idée d'avoir l'information comme attribut à cause des nombreux cas d'exceptions difficiles à gérer (exemple : une personne avec un même parti qui une année est considéré comme write-in puis non, etc.)

• **Person**

La table « person » qui correspond à l'identité du candidat avec en attribut :

- « person_name » qui correspond à son nom et prénom

• **Party**

La table « party » qui correspond à un parti politique avec en attribut :

- « party_name » qui correspond au nom du dit parti

IV – Choix supplémentaires

Cette partie porte sur les choix portés aux données, aux données supplémentaires ajoutées ainsi que les problèmes encourus et leurs solutions.

Choix arbitraires sur le fichier csv initial

- Pour la colonne « unofficial », nous l'avons supprimé et avons décidé de considérer les votes dont les résultats sont non officiels comme tel car ils n'y avaient pas les résultats officiels pour ces élections.

- Pour la colonne « stage », nous avons retiré les lignes correspondant aux valeurs PRI (PRIMARY) car ceux-ci correspondaient aux élections primaires dont le but était d'élire les représentants des différents partis, ce qui n'est pas une donnée qui nous intéresse

Index

- cluster sur candidat avec parti et personne, car en général on appelle le candidat avec son nom et le nom de son/ses parti/s

- bitmap sur les booléens genre run_off_election et special_election, car ce sont des booléens

Agrégats

- party_weight_per_state_aggregate : nombre de vote par parti et ceux que ça représente par rapport à tous les votes dans un state en pourcentage

- winner_party_per_state_per_year_aggregate : le candidat gagnant par district par année (vérif run off inclus ou non)

- winner_party_globally_per_year_aggregate : pour chaque année le parti ayant eu le plus de district avec un vote majoritaire pour son parti

- district_weight_per_year_aggregate : le nombre de vote par année et par district tous partis et candidats confondue / le totale de vote tout district tout partis et tout candidats confondus. Cela permet de voir le poids que représente chaque district dans les élections de manière globale

Données pour la carte

Pour visualiser les données sous forme de carte, nous avons utilisé **plotly.express.Choropleth** qui est une fonction prenant de nombreux paramètres, dont un set de données au format **geojson** (json géographique). Fort heureusement, plotly possède déjà des jeux de données, dont un composé de tout les comtés des États-Unis (1). Nous nous sommes donc basés sur celui-là. Cependant il manquait le comté de Shannon (Oglala Lakota) donc on a dû se baser sur un repo GitHub, en adaptant certaines données au jeu que nous avions (2), notamment les codes fips.

Cependant, les districts sont une composition de plusieurs comtés, donc nous avons fait appel à un autre jeu de données, celui-ci reliant les comtés et les districts (3).

Grâce à tout ces jeux de données, nous avons pu fusionner les différents comtés de la carte en districts en se basant sur les codes fips pour les relier.

Présidents

Pour les graphes, nous avons décidé d'ajouter l'information portant sur le président ainsi que le parti qu'il représente. Pour cela on s'est basé sur un fichier csv provenant d'un repo github (4). On aurait pu le faire à la main, mais après avoir brièvement vérifié, le fichier était fiable.

Lignes dupliquées pour les write-in

Dans le fichier csv fourni, on s'est rapidement rendu compte que certaines lignes étaient dupliquées. Cela était pour les candidats portant le nom de « WRITEIN » et n'ayant pas de parti politique. Il y avait souvent 2 lignes avec des nombres de votes différents. Cependant cela empêchait la sauvegarde en base de données car un fait de vote porte sur une année, un district et un candidat. Cependant dans ce cas-là, pour la même année, le même district et le même candidat (si on peut considérer ça comme un candidat…) on avait 2 lignes, donc impossible.

Le choix a donc été fait de faire la somme des votes et juste porter l'information que les « WRITEIN » ont reçu un tel total de vote. De toute façon sans informations supplémentaires on ne pouvait pas faire mieux.

Pour identifier les lignes atteintes nous ne pouvions pas simplement faire une recherche classique dans le fichier étant donné que certes le début de la ligne était identique mais pas la fin, donc on a utilisé des expressions régulières pour identifier les lignes et les modifier.

Voici ce à quoi ressemblait la recherche :

Étant donné que le nombre de lignes affectées était petit, nous avons simplement fait les sommes à la main, mais si besoin il aurait été possible de réaliser un script python.

La solution apportée nous a permis de conserver le nombre de vote obtenu par les write-in malgré tout, ce qui a été utilisé lors des visualisations portant sur les succès involontaires.

V – Sources

1. Geojson des comtés US https://raw.githubusercontent.com/plotly/datasets/master/geojson-counties-fips.json
2. Geojson du comté de Shannon (manquant au 1) https://github.com/johan/world.geo.json/blob/master/countries/USA/SD/Shannon.geo.json
3. Lien entre comté et district https://www.census.gov/geographies/reference-files/time-series/geo/relationship-files.2020.html

4. Présidents US avec parti https://github.com/awhstin/Dataset-List/blob/master/presidents.csv