UNIVERSITÉ DE NANTES

Rapport de Base de données évoluées

Aniss BENTEBIB, Camille-Amaury JUGE, Aya HAITI, Clément ANTHEAUME

Respectivement M1 Informatique ATAL et DS, ALMA et DS.

Table des matières

1. Introduction

Rappelons brièvement le contexte, ce projet a pour but de créer un entrepôt de données qui doit permettre par la suite de répondre à un contexte fixé par les étudiants. Pour cela, le dataset choisi doit évidemment répondre à des contraintes réalistes et s'adapter aux nouvelles bases de données sur le marché.

Dans ce cadre, nous avons donc choisi d'étudier les accidents liés au trafic routier aux Etats-Unis entre 2016 et décembre 2019. Ce dataset est fourni par des chercheurs [1] et possède un peu moins de 3 millions de lignes. Ainsi, dans la limite de nos machines personnelles, cela représente une base de données correctement fournie et représentative des base de données des entreprises.

Nous nous sommes alors mis dans le contexte d'analystes de données, de chercheurs afin de comprendre quels sont les différents axes, heure de la journée et autres facteurs pouvant influer sur les accidents et comment peut-on prétendre à les réduire une fois que nous avons la connaissance de ces données.

Pour cela, nous vous joignons le lien du github qui vous permettra d'obtenir la procédure d'installation en local de notre environnement. Nous avons choisi de stocker nos données dans un entrepôt MongoDB [2] avec l'interface d'administration Studio 3T [3]. Pour la partie interprétation et visualisation, nous utiliserons un environnement Anaconda [4] et Jupyter Notebook [5] au moyen de Python 3.7 [6]. Pour plus de détails, référez-vous au dépôt Github suivant :

https://github.com/camilleAmaury/BDD_Evoluees

2. La base de données en profondeur

Pour la suite du rapport, nous nous réfèrerons régulièrement au Jupyter Notebook "DataVisualisation.Ipynb" dans le dossier de notre projet. Nous vous donnerons à quelle cellule se réfère notre travail par une simple indication dans le texte.

Dans un premier temps, nous nous devons d'explorer les données que nous avons afin de les comprendre et de percevoir les angles d'interprétation que nous allons ensuite visé. Pour cela, je vous invite à voir les *cellules 3 à 5*. On remarque que notre dataset est composé de 49 colonnes avec des types très variables (booléens, Chaînes de caractères catégoriques, nombres flottants, nombres entiers, …). De plus, nous

possédons des données d'indication géographique (latitude, longitude, ville, pays, …), des données valuées (température, longueur de route, humidité, …) qui vont nous permettre d'effectuer des regroupements et des calculs d'indicateurs.

Ainsi, maintenant que nous avons pris connaissance de la base en elle-même, nous allons justifier notre choix concernant l'utilisation de MongoDB. Etant donné que ce Système de Gestion de Base de Données (SGBD) est spécialisé pour une approche document (Le plus possible dans un unique objet appelé le document), notre collection se prête plutôt bien à cette méthode de traiter les données puisque initialement celles-ci sont déjà présentées sous la forme d'un CSV comprenant un accident et tout ses détails par ligne. Tout peut être regroupé dans un unique document référencé comme étant un accident. MongoDB a la capacité de gérer efficacement ce type de représentation en dénormalisant les schémas SQL classiques.

Par exemple, le lieu (ville et pays) de l'accident aurait pu appartenir à une table indépendante étant donné que ce même lieu puisse se retrouver dans plusieurs accidents. Mais les jointures coûtent particulièrement plus cher en MongoDB qu'en SQL (même si elles ont un coût non négligeable en SQL aussi). Ainsi, il est plus avantageux de tout réunir sous un même document.
Un problème que cela peut poser : si il n'y a pas de table référençant les lieux, on peut imaginer des fautes de saisie sur des enregistrements qui mèneront par la suite à ne pas avoir toutes les données souhaitées.

Finalement, nous avons choisi une approche NoSQL au vu du nombre de données, même si les gros SGBD comme ORACLE ou PostGreSQL aurait pu faire l'affaire pour 3 millions d'enregistrements, il est plus viable niveau performance de préférer l'approche MongoDB.

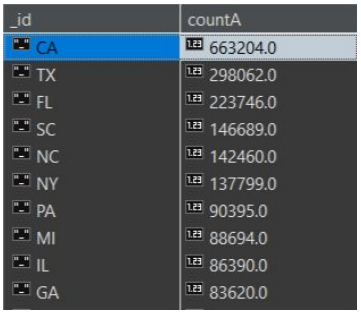
Voici donc la table décrivant nos données. Celle-ci est unique car nos données reposent principalement sur des agrégats, et non sur une structure complexe :

3. Requêtes NoSQL pertinentes

Les requêtes écrites ci-dessous ont été écrites au format mongo shell et exécutée dans l'IntelliShell prévu à cet effet dans le logiciel Studio3T.

Requête 1 :

L'objectif de cette requête est de recenser le nombre d'accident enregistrés en par état. Les résultats de cette requête montrent de grosses inégalités dans le recensement des données. Cela peut s'expliquer soit par le nombre d'accidents.
Résultats :



_id	countA
CA	663204.0
TX	298062.0
FL	223746.0
SC	146689.0
NC	142460.0
NY	137799.0
PA	90395.0
MI	88694.0
IL	86390.0
GA	83620.0

Requête 2 :

```
db.getCollection("US_Accidents_Dec19").find(
{
    Wind_Direction : "North",
    State : "NY",
    Wind_Speed : { $gt : '9.0'}
},
    {
        "Humidity" : 1.0,
        "Temperature" : 1.0,
        "Wind_Chill" : 1.0,
        "Humidity" : 1.0,
        "Pressure" : 1.0,
        "Visibility" : 1.0,
        "Wind_Direction" : 1.0,
        "Wind_Speed" : 1.0,
        "Precipitation" : 1.0,
        "Weather_Condition" : 1.0
    }
);
```

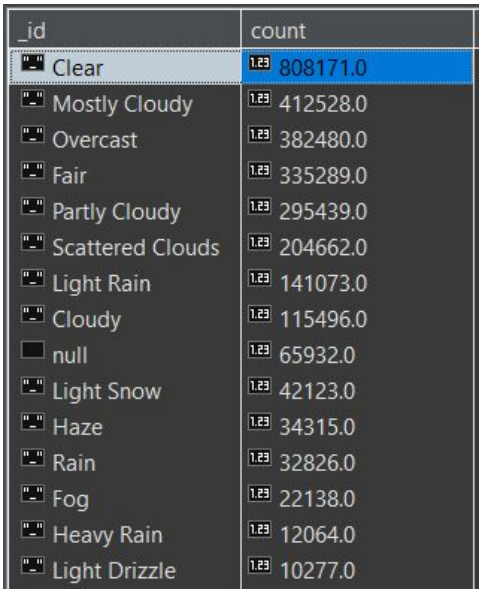
db.getCollection("US_Accidents_Dec19").aggregate([{$group:{_id: "$Wind_Direction",count: { $sum: 1}}},{$sort:{'count':-1}}]);

Objectifs :
Recenser les différents accidents qui ont eu lieu à New York avec un vent très fort (>9mph) allant du sud vers le nord.

Requête 3 :

Objectifs : Voir si un temps arbitrairement jugé difficile pour la conduite augmente les risques d'accident.

Résultats :
On remarque que, contrairement aux idées reçues, un temps clair ou simplement nuageux, qu'on pourrait juger propices à une conduite confortable, n'empêchent pas de nombreux accidents d'arriver.



_id	count
Clear	808171.0
Mostly Cloudy	412528.0
Overcast	382480.0
Fair	335289.0
Partly Cloudy	295439.0
Scattered Clouds	204662.0
Light Rain	141073.0
Cloudy	115496.0
null	65932.0
Light Snow	42123.0
Haze	34315.0
Rain	32826.0
Fog	22138.0
Heavy Rain	12064.0
Light Drizzle	10277.0

Requête 4 :

Nombre d'accidents par Bins de valeurs sur différents champs : (Temperature(F), Wind_Chill(F), Humidity(%), Pressure(in), Visibility(mi), Wind_Direction, Wind_Speed(mph), Precipitation(in), Weather_Condition) télécharger le résultat des requêtes et les résultats sous jupyter → pourquoi pas faire un "top n" à chaque fois
→ Montre l'impact des éléments sur les accidents

Requête 5 : RANK endroits les plus meurtriers (county, street) et ce qui les caractérisent : (Amenity, Bump, Crossing, Give_Way, Junction, No_Exit, Railway, Roundabout, Station, Stop, Traffic_Calming, Traffic_Signal, Turning_Loop)
→ Montre les endroits les plus dangereux et les causes dues à l'affichage routier ou la configuration routière

Requête 6 : Nombre d'Accidents les plus perturbants (severity) par lieu (City, Street) en fonction de l'heure dans la journée (Start_Time) → Plage horaire serait mieux
montre les plus grandes perturbations occasionnées en fonction des moments de la journée

```
var villesaccidentees = db.getCollection("US_Accidents_Dec19").aggregate({$group:{_id: "$City",count: { $sum: 1}}},{$sort:{'count':-1}}]);
-----------------------------------------------------------------------------------------------------------------------------
db.getCollection("US_Accidents_Dec19").find({City : "Houston"}).sort({ "Severity" : -1}).limit(10)
```

Requête 7 : Nombre d'accidents par tranche horaires. Soit utiliser les heures pour découpage, soit utiliser les horaires civils, nautiques et astronomiques (qui sont quasi les mêmes)

Requête 8 : Quels sont les infrastructures de trafic les plus souvent théâtre d'accident : Compter le nombre d'accidents par différente infrastructure (Roundabout, Stop…) et calculer la fréquence, donc nombre accidents avec cette infra/total nombre accidents. Puis classer les infra en fonction de la fréquence d'implication à des accidents

---

4. Note sur l'organisation et le déroulement du projet

Afin d'organiser notre projet et de permettre un travail efficace en équipe, nous avons décidé de suivre les principes de la méthode AGILE (hiérarchie horizontale et fonctionnement par Sprint) grâce notamment à l'outil Trello [7]. Nous avons pu organiser nos différentes tâches par importance et de manière atomique afin de produire des versions augmentant la qualité et la quantité du projet.

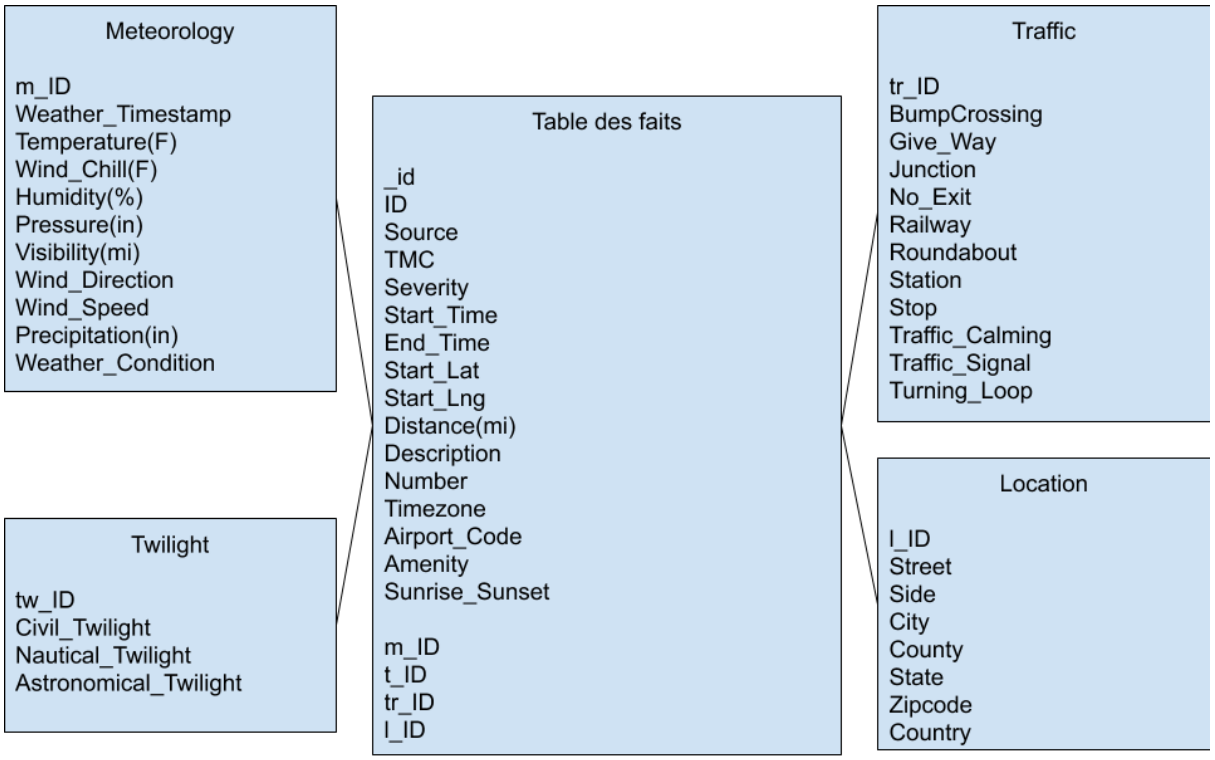
De plus, nous avons choisi Github plutôt que GitLab avec lequel nous sommes plus familier.

Concernant le projet en lui-même, nous avons rencontré certaines difficultés notamment sur ...

5. Conclusion

Ce projet aura été pour nous l'occasion de manipuler un nouvel environnement de gestion de base de données. Se confronter à de nouvelles technologies est toujours un challenge intéressant et enrichissant pour renforcer et élargir nos compétences.

En terme d'amélioration, structurer nos tables en étoile pourrait permettre de gagner de la place. Cependant, au vue des évènements enregistrés et historisés, il n'était pas utile selon nous d'adapter la structure ainsi car le gain aurait été beaucoup plus limité que sur des données qui se manifestent plus régulièrement. Le modèle est décrit par les tables suivantes :



6. Sources

---

[1] https://www.kaggle.com/sobhanmoosavi/us-accidents, visité le 17/02/2020. Se réfère aux travaux :

- Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, and Rajiv Ramnath. "A Countrywide Traffic Accident Dataset.", 2019.
- Moosavi, Sobhan, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, Radu Teodorescu, and Rajiv Ramnath. "Accident Risk Prediction based on Heterogeneous Sparse Data: New Dataset and Insights." In proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM, 2019.

[2] https://www.mongodb.com/fr, visité le 19/02/2020. Site portail de MongoDB.

[3] https://studio3t.com/, visité le 19/02/2020. Site portail de Studio 3T.

[4] https://www.anaconda.com/, visité le 19/02/2020. Site portail de Anaconda.

[5] https://jupyter.org/, visité le 19/02/2020. Site portail de Jupyter Notebook.

[6] https://www.python.org/, visité le 19/02/2020. Site portail de Python.

[7] https://trello.com/, visité le 19/02/2020. Site portail de Trello.