Rapport du Projet Entrepôt de Données

Florent ALAPETITE - Alexandre BOUDINE - Florent GAILLARD

Bases de données évoluées

15 novembre 2017

Table des matières

1 Introduction

Dans le cadre de notre cours de bases de données évoluées, nous avons travaillé sur un projet de réalisation d'un entrepôt de données analysable à l'aide de requêtes OLAP. Pour cela, nous avons cherché un dataset libre de droit afin de pouvoir compter sur des données exploitables aves des faits rééls pour réaliser notre entrepôt.

Le dépôt GitHub contenant les scripts de création et d'interrogation de notre entrepôt de données ainsi que notre solution d'intégration est disponible à l'adresse :

https://github.com/FlorentALAPETITE/OLAP_Database_Project

2 Description du dataset utilisé

Le dataset que nous avons choisi pour notre entrepôt de données recense les homicides survenus aux Etats Unis entre 1980 et 2014, qu'ils soient résolus ou non.

Chaque crime possède des informations concernants le ou les victimes, le ou les auteurs ainsi que l'agence qui est chargée de résoudre l'affaire. De plus, il est précisé le lieu, la date ainsi que d'autres informations importantes relatives à l'incident comme le type d'arme utilisé.

Le dataset complet est disponible à l'adresse suivant :

https://www.kaggle.com/murderaccountability/homicide-reports

Pour des raisons de place, le dataset que nous avons utilisé tout au long du développement de notre entrepôt est un sous-dataset comprenant l'intégralité des homicides de l'années 2013 ainsi qu'un certain nombre d'homicides des années 2012 et 2014 (environ 16 000 homicides sur les 650 000 disponibles dans le dataset originel).

3 Création de l'entrepôt de données

3.1 Choix techniques

Nous avons décidé de nous orienter vers la réalisation d'un entrepôt de données fonctionnant avec le système de gestion de bases de données **Oracle**. Ainsi, nous avons pu travailler autour de la réalisation d'un schéma d'entrepôt de données de type Entité-Relation.

De plus, nous avons pu compter sur l'implémentation des différents opérateurs OLAP et d'agrégation présents nativement dans le SGBD Oracle pour réaliser nos requêtes d'analyse sur notre entrepôt de données.

3.2 Conception de l'entrepôt de données

Nous avons débuté l'étape de conception de l'entrepôt de données par une phase d'analyse des données présentes dans notre dataset et des faits rapportés dans celui-ci. Nous avons décidé de choisir un schéma d'entrepôt de type **schéma en étoile**, c'est à dire un modèle entité-relation composé d'une table de **fait** (table centrale de l'entrepôt contenant un nombre important d'attributs) relié à des **dimensions** (tables secondaires de l'entrepôt contenant un plus faible nombre d'attributs et permettant d'orienter l'analyse). Le schéma de notre entrepôt de données est disponible en annexe (Annexe 1).

Nous en avons profité pour définir la granularité de notre table de fait : un tuple correspond à un homicide (rapporté le mois d'une année par une agence précise dans un lieu précis). De ce choix de granularité a découlé le choix de 4 dimensions :

- Une dimension **DimDate**{month,year,season} regroupant les informations temporelles des homicides.
- Une dimension **DimProfile**{sex,age,ethnicity,race} regroupant des profiles de personnes impliquées dans les homicides (aussi bien auteurs que victimes).
- Une dimension **DimAgency**{agencyCode,agencyName,agencyType} regroupant les informations de l'agence ayant constaté l'homicide.
- Une dimension **DimPlace**{city, state} regroupant les données de localisation des homicides.

La table de fait, quant à elle, conserve :

- Un identifiant unique relatif au crime
- L'identifiant de l'agence chargée de l'affaire
- L'identifiant de la date (mois, année)
- L'identifiant du lieu du crime (ville, état)
- Le nombre de victimes supplémentaires (zéro par défaut)
- Les informations concernant la victime (sexe, âge, ethnicité, race)
- Le nombre d'agresseurs supplémentaires (zéro par défaut)
- Les informations concernant l'auteur du crime(sexe, âge, ethnicité, race)
- L'arme utilisée
- Le lien entre la victime et l'agresseur
- La source du crime
- Un booléen indiquant si le crime est résolu ou non

3.3 Intégration des données

Une fois la conception de notre entrepôt de données réalisée, nous nous sommes concentrés sur l'intégration des données de notre dataset. Nous avons choisi de partir du dataset au format CSV. Après examen de celui-ci, nous avons constaté qu'il était globalement bien formé et comportait la plupart des attributs que nous souhaitons utiliser dans notre entrepôt. Nous avons donc choisi d'utiliser une solution d'intégration simple et efficace : **JDBC**.

JDBC (Java DataBase Connectivity) est une solution permettant de réaliser une interface entre un programme Java et une SGBD comme Oracle. Nous avons décidé de l'utiliser car il est relativement simple d'utilisation et il permet d'effectuer un grand nombre d'opérations d'intégration de données en se basant sur les fonctionnalités de Java.

Notre intégration des données débute par la lecture de notre Dataset CSV par un programme Java très léger. Chaque ligne du dataset est alors transformé en de multiples requêtes d'insertion (pour chaque dimension et la table des faits) via l'utilisation de PreparedStatement couplé à du "Batch processing" (execution directe d'un ensemble de requêtes au lieu d'une par dialogue programme/SGBD).

Nous avons profité des avantages du langage Java pour faciliter l'intégration des données de notre CSV par l'utilisation d'expression régulières dans la manière de découper les informations de chaque ligne. En effet, le fichier CSV du dataset que nous utilisons comporte un léger problème : les noms d'agences peuvent comprendre des virgules. La virgule étant le séparateur des données du CSV, la lecture puis la découpe des différentes informations n'était pas conforme

lors de notre intégration de données. Nous avons ainsi pu résoudre le problème par l'utilisation d'expressions régulières.

L'utilisation de JDBC comporte également l'avantage de pouvoir facilement ajouter des données non-présentes nativement dans le dataset mais pouvant s'avérer pratique pour l'analyse de notre entrepôt. Nous avons par exemple décidé d'ajouter les informations sur la saison (été,automne,hiver,printemps) directement dans le programme Java réalisant l'intégration des données afin de pouvoir compter par la suite sur un autre niveau d'agrégation (par saison).

3.4 Requêtes et analyse de l'entrepôt de données

Dans la dernière partie du projet, nous avons réalisé différentes requêtes nous permettant d'analyser les données contenues dans notre entrepôt. Pour cela, nous avons utilisé les outils implémentés en SQL par le SGBD Oracle. Nous avons essayé d'utiliser un panel complet d'opérateurs OLAP afin d'avoir un maximum de requêtes interressantes.

3.4.1 Requête 1

```
1 SELECT city,state,month,year, count(*) as nbIncident
2 FROM admi2.Fact NATURAL JOIN admi2.DimDate NATURAL JOIN admi2.DimPlace
3 GROUP BY ROLLUP(year,month,state,city);
```

Cette requête permet d'obtenir le nombre d'homicides selon plusieurs agrégats :
- année, mois, état et ville
- année, mois et état
- année et mois
- année
- total de tous

3.4.2 Requête 2

```
1 SELECT * FROM (
2         SELECT SUM(victimCount+1) as nbVictimes, city, state
3         FROM admi2.Fact NATURAL JOIN admi2.DimPlace
4         GROUP BY city, state
5         ORDER BY nbVictimes DESC)
6 WHERE ROWNUM<=5;
```

La requête retourne le TOP 5 (avec ROWNUM) des villes avec le plus de de victimes tout au long de la période de temps.

3.4.3 Requête 3

```
1 SELECT perpetratorAge as age, perpetratorSex as sex,
2     perpetratorRace as race, perpetratorEthnicity as ethnicity,
3     SUM(victimCount+1) as nbVictimes
4 FROM admi2.Fact, admi2.DimProfile
5 WHERE perpetratorAge = admi2.DimProfile.age
6     AND perpetratorSex = admi2.DimProfile.sex
7     AND perpetratorRace = admi2.DimProfile.race
8     AND perpetratorEthnicity = admi2.DimProfile.ethnicity
```

```
GROUP BY CUBE (perpetratorAge, perpetratorSex, perpetratorRace,
    perpetratorEthnicity);
```

Cette requête renvoie le nombre de victimes en fonction du profil de l'agresseur. Tous les regroupements possibles par rapport à l'âge, le sexe, la race et l'origine de l'agresseur sont faits.

3.4.4 Requête 4

```
SELECT month,year, sum(victimCount+1) as nbVictimes, sum(sum(victimCount+1))
    over (order by year,month) as accumulationVictimes
FROM admi2.FACT natural join admi2.DimDate
GROUP BY year,month;
```

La requête renvoie le nombre de victimes par mois et année ainsi que le cumul de victimes par mois, trié par ordre de mois et d'année.

3.4.5 Requête 5

```
SELECT weapon, sum(1+victimCount) as victimes
FROM admi2.Fact
GROUP BY weapon
ORDER BY victimes DESC;
```

Cette requête liste toutes les armes impliquées dans des crimes ainsi que leur nombre d'implication.

3.4.6 Requête 6

```
SELECT admi2.Fact.year, admi2.Fact.state, admi2.Fact.victimSex,
    SUM(victimCount+1) as victimes
FROM admi2.Fact, admi2.DimProfile, admi2.DimDate
WHERE admi2.Fact.year = admi2.DimDate.year
    AND admi2.Fact.month = admi2.DimDate.month
    AND victimAge = admi2.DimProfile.age
    AND victimSex = admi2.DimProfile.sex
    AND victimRace = admi2.DimProfile.race
    AND victimEthnicity = admi2.DimProfile.ethnicity
GROUP BY GROUPING SETS ((admi2.Fact.state, admi2.Fact.year, admi2.Fact.victimSex),
    (admi2.Fact.year,admi2.Fact.victimSex),
    (admi2.Fact.state, admi2.Fact.victimSex),
    (admi2.Fact.victimSex));
```

La requête recense le nombre de victimes selon les regroupements suivants :
- l'etat, l'année et le sexe de la victime
- l'année et la sexe de la victime
- l'état et le sexe de la victime
- le sexe de la victime

3.4.7 Requête 7

```
SELECT admi2.DimDate.month, admi2.DimPlace.state,
    sum(1+victimCount) as victims, GROUPING(admi2.DimDate.month) as monthB,
    GROUPING(admi2.DimPlace.state) as stateB
FROM admi2.Fact, admi2.DimDate, admi2.DimPlace
WHERE admi2.Fact.year = admi2.DimDate.year
    AND admi2.Fact.month = admi2.DimDate.month
    AND admi2.Fact.city = admi2.DimPlace.city
    AND admi2.Fact.state = admi2.DimPlace.state
GROUP BY ROLLUP(admi2.DimDate.month,admi2.DimPlace.state);
```

Le résultat de la requête regroupe le nombre de victimes selon le mois, le mois et l'état, ainsi que tous confondus. Deux colonnes sont ajoutées afin de savoir si un agrégat est fait sur le mois ainsi que sur l'état.

3.4.8 Requête 8

```
SELECT season, sum(1+victimCount) as victims,
        RANK() OVER (ORDER BY sum(1+victimCount) DESC) as rank
FROM admi2.Fact, admi2.DimDate
WHERE admi2.Fact.year = admi2.DimDate.year
    AND admi2.Fact.month = admi2.DimDate.month
GROUP BY season;
```

La requête retourne un classement des saisons par rapport au nombre de victimes.

3.4.9 Requête 9

```
SELECT admi2.DimDate.year, admi2.DimDate.season,
    sum(1+victimCount) as victims,
        RANK() OVER (PARTITION BY admi2.DimDate.year
        ORDER BY sum(1+victimCount) DESC) as rank
FROM admi2.Fact, admi2.DimDate
WHERE admi2.Fact.year = admi2.DimDate.year
    AND admi2.Fact.month = admi2.DimDate.month
GROUP BY (admi2.DimDate.year,admi2.DimDate.season);
```

Cette requête retourne un classement des saisons par rang (RANK) au sein d'une même année, basé sur le nombre de victimes. De plus, le résultat est partitioné (PARTITION BY) par années.

3.4.10 Requête 10

```
SELECT agencyType, COUNT(crimeSolved) as nbCrimeSolved,
    NTILE(4) over(order by COUNT(crimeSolved) desc) as quarter
FROM admi2.Fact, admi2.DimAgency
WHERE admi2.Fact.agencyCode = admi2.DimAgency.agencyCode
    AND crimeSolved = 'Yes'
GROUP BY agencyType;
```

La requête retourne le type des agences ainsi que leur nombre de crimes résolus et est ordonné par quart via la fonction NTILE.

3.5 Analyse des résultats

Nous avons créé un script permettant de récupérer les résultats de chaque requête sous la forme d'un fichier .csv. De cette manière, il est très simple de réaliser une analyse détaillée des données récupérées en résultat de la consultation de l'entrepôt de données.

Malheuresement par manque de temps, nous n'avons pas pu fournir de script permettant de générer automatiquement des diagrammes à partir de nos résultats (avec par exemple le langage R).

4 Annexe

4.1 Annexe 1 : schéma en étoile de notre entrepôt de données