LE BARS Yannis
FORTIN Guillaume
TRAVERS Clément

Groupe de TD : B

Rapport de projet
Base de données évoluées

UFR Sciences et Techniques - Nantes

Master 1 ALMA - 2017-2018

UNIVERSITÉ DE NANTES

Sommaire

I - Présentation du dataset

Ce projet a pour but de créer un entrepôt de données basé sur le système OLAP, nous devons ainsi créer un schéma en étoile ou en flocon et interroger les données à l'aide de requêtes OLAP.

Pour concevoir cet entrepôt il nous a été demandé de choisir plusieurs dataset et de les lier entre eux. Nous avons donc retenu les dataset suivants :

- https://grouplens.org/datasets/movielens/
- http://www.omdbapi.com/

Un troisième dataset était initialement prévu dans l'entrepôt de données, mais étant disponible dans un format peu pratique pour la transformation en base de données (.xlsx) et possédant plusieurs feuilles de calculs dans le même fichier, nous avons fait le choix d'abandonner son utilisation.

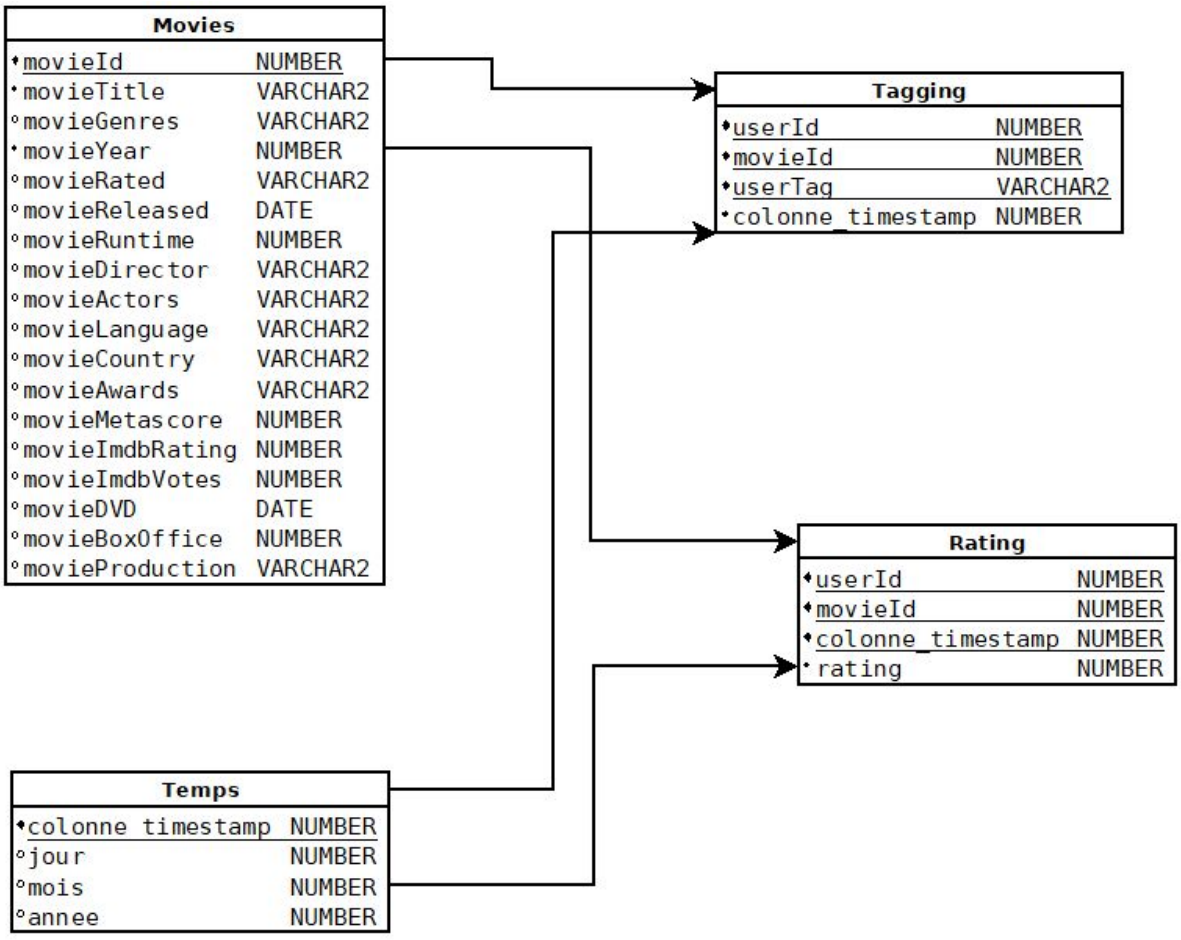
Le premier dataset regroupe des films possédant des tags et des notes données par des utilisateurs avec des informations succinctes sur les films.
Tandis que le deuxième dataset regroupe une liste de film avec les données associées à chaque film (année, genres, réalisateur, acteurs, etc), celui-ci nous a permis de compléter les informations du premier dataset.

II - Conception de l'entrepôt de données

Nous avons réalisés l'entrepôt de données sous la forme relationnelle, en utilisant Oracle. Nous avons choisis de représenter nos données sous la forme d'une constellation de faits, composée de deux tables de faits, appelées Rating et Tagging, et de deux dimensions, appelées Movies et Temps.

- Rating regroupe les notes donnés par les utilisateurs à un film.
- Tagging regroupe les tags attribués à un film, encore une fois par les utilisateurs

- Movies regroupe toutes les informations concernant les films, telles que le nom du film, les genres du film, son réalisateur, etc.
- Temps regroupe les informations de datage, à savoir le jour, le mois et l'année correspondant à un timestamp, utilisé pour dater les informations contenues dans les deux tables de faits. Le jour, le mois et l'année sont mis à jour automatiquement à l'insertion d'un timestamp grâce à un trigger.



Pour arriver à cette constellation nous avons omis certains attributs présents dans les datasets qui nous ont paru non pertinent et nous avons enlevé certaines tables de base car nous avons jugés pouvoir les retrouver via des requêtes. Par exemple, la table qui associe un film avec tous ses tags reçus peut être retrouvé avec une requête.

L'insertion dans l'entrepôt a été réalisé à l'aide de scripts C++ permettant d'extraire les données des dataset et de les insérer automatiquement dans l'entrepôt déjà créé. Ces scripts analysent le contenu du dataset afin de créer un fichier sql réalisant à l'exécution l'insertion dans les tables de notre entrepôt de données.

III - Requêtes OLAP

L'autre objectif de ce projet est la création de plusieurs requêtes utilisant les différents opérateurs OLAP, tels que GROUP BY ROLL UP, RANK() ou encore PARTITION BY.

Première requête

On utilise GROUP BY pour obtenir la moyenne des notes des utilisateurs pour un tag.

SELECT AVG(rating) **as** moyenne, userTag
FROM (Rating **JOIN** Tagging **ON** (Rating.userId = Tagging.userId **and** Rating.movieId = Tagging.movieId)) **NATURAL JOIN** Movies
GROUP BY (userTag)
ORDER BY (moyenne);

Deuxième requête

On compte le nombre de films sortis en DVD par an et par pays, en utilisant l'opérateur OLAP GROUP BY CUBE.

SELECT YEAR(movieDVD) **as** anneeDVD, movieCountry, **count**(movieId)
FROM Movies
GROUP BY CUBE (anneeDVD, movieCountry);

Les films n'ayant pas de date de sortie en DVD renseignée permettront de connaître le nombre de films n'étant pas sortis en DVD.

Troisième requête

On calcule la somme des entrées du box office pour tous les films, par année.

SELECT movieYear **AS** annee, **SUM**(movieBoxOffice) **AS** boxOffice
FROM Movies
WHERE movieBoxOffice **IS NOT NULL**
GROUP BY GROUPING SETS (movieYear);

GROUPING SETS nous permet d'isoler le calcul pour chaque année, afin d'obtenir le résultat souhaité.

Quatrième requête

On récupère les 10 films ayant reçus la meilleure moyenne, accompagné de leur rang général dans le classement des chiffres les plus élevés du box office

```
SELECT movieTitle, MAX(avg_rating), RANK() over (ORDER BY movieBoxOffice DESC) AS rangBoxOffice
FROM (
   SELECT movieId, movieTitle, AVG(rating) AS avg_rating, movieBoxOffice
   FROM Movies NATURAL JOIN Rating
   GROUP BY movieId
)
WHERE rownum <= 10
ORDER BY rating DESC;
```

Cinquième requête

On sélectionne les 20 meilleurs succès au box office, en réalisant une partition sur leurs producteurs

```
SELECT movieProduction, movieTitle, movieYear, movieBoxOffice, rank() OVER (PARTITION BY movieProduction ORDER BY movieBoxOffice DESC) AS rank
FROM Movies NATURAL JOIN Rating
WHERE rownum <= 20;
```

Sixième requête

On récupère le film ayant reçu le meilleur rating moyen par les utilisateurs pour chaque année

```
SELECT movieTitle, movieYear AS annee, MAX(avg_rating)
FROM (
   SELECT movieId, movieTitle, movieYear, AVG(rating) AS avg_rating
   FROM Movies NATURAL JOIN Rating
   GROUP BY movieId
)
WHERE movieYear IS NOT NULL
GROUP BY GROUPING SETS (movieYear);
```

IV - Conclusion

Pour conclure, la plupart des problèmes liés à l'intégration des données des datasets ont été résolu, néanmoins, il reste un problème non résolu :
Le premier dataset possède un timestamp pour indiquer la date, or il nous a été impossible de caster l'entier récupéré du fichier csv pour le transformer en timestamp, de ce fait, il nous a été impossible de rentrer les timestamp dans l'entrepôt. Pour permettre de pouvoir faire des test et rendre l'entrepôt fonctionnel nous avons remplacer l'insertion de chaque timestamp par CURRENT_TIME.

Pour l'extension de l'entrepôt, nous avons pensé qu'il aurait pût être intéressant de trouver ou ajouter manuellement une table contenant des informations sur les utilisateurs et ainsi pouvoir augmenter le nombre de requêtes intéressantes possibles avec cet entrepôt. Notamment car nous avons eu du mal à trouver des requêtes qui se différencient vraiment les unes des autres, nous empêchant ainsi d'atteindre le nombre de requêtes demandées.