OLAP Movie Database Coursework
-
Report

·X1I1050·

Roxane Bellot, Hippolyte Dubois, Marvin Jean

Novembre 2017

Table des matières

Introduction

Dans le cadre du cours X1I1050, il nous a été demandé de concevoir, implémenter, peupler et utiliser un entrepôt de données. Pour cet exercice, nous avons choisi d'analyser les entrées et recettes de films en fonction des genres et années.

Chapitre 1

Travail réalisé

1.1 Conception

Pour démarrer la phase de conception, nous avons suivi la méthodologie donnée en cours :

1. *Définir le système* : le succès des films (nombre d'entrées, bénéfice, notation), afin de pouvoir réaliser un étude sur les propriétés importantes des films. Mettre en évidence les éventuelles caractéristiques qui augmenteraient le succès d'un film.
2. *Choisir un grain* : un film. Pour analyser le succès nous avons besoin premièrement des propriétés du film, principales données sur lequel nous croiserons nos faits. Nous aurons aussi besoin de la date, pour étudier les dates de sortie et plus généralement les périodes (mois, saison, année, décennie). Nous aurons aussi besoin des critiques et/ou des notes données aux films, qui sont, avec le nombre d'entrées, les critères permettant de déterminer le succès d'un film. Nous aurons aussi une dimension pour les zones géographiques et linguistiques, pour analyser les tendances.
3. *Choisir un fait* : les faits correspondront à ce qui peut varier en fonction du film, le nombres d'entrée et le bénéfice qu'il a généré, la durée pendant laquelle il est resté à l'affiche, le genre, la boite de production. Avec tout ceci, il sera possible de faire différent traitements en fonction des films.

1.2 Récolte de données

Nous exploitons des données tirées du dataset du site `https://www.themoviedb.org/`, obtenues via des appels sur son api. Les appels sont effectués

au moyen d'un script Python qui récupère les informations des films référencés par une liste d'ID propres à l'API, puis les compile dans un seul fichier JSON.

Ce fichier JSON est ensuite utilisé pour construire les instructions *INSERT* qui seront éxécutées par Oracle et qui correspondent au modèle de notre base de données.

1.3 Requêtes

Nous avons interrogé notre entrepôt de données via quelques requêtes, détaillées ici. Note : Pour des raisons de lisibilité, les résultats comportant trop de lignes ont été tronqués.

1.3.1 Affichage des films pour adultes

Requête

```
1 SELECT fi.title
2 FROM fait fa, d_film fi, d_genre ge
3 WHERE fa.d_genre_id=ge.id
4     AND fa.d_film_id = fi.id
5     AND ge.adult = 1;
```

Commençons avec une requête simple pour prendre en main notre base de données.

Résultat

```
1 TITLE
2 --------------------------------------------------
3 Rocco - Perfect Girls 5
```

1.3.2 Entrées en fonction des compagnies et du genre

Requête

```
1 SELECT ge.genre_name, co.name_, sum(fa.admissions) AS "nb_entrees
      "
2 FROM fait fa, d_genre ge, d_company co
3 WHERE fa.d_genre_id = ge.id
4     AND fa.d_company_id = co.id
5     AND ge.genre_name !='NULL'
6     AND co.name_ !='NULL'
7 GROUP BY ROLLUP(ge.genre_name,co.name_);
```

Donne le total des entrées pour chaque genre pour chaque compagnie de production, mais aussi pour chaque genre en tout.

GENRE_NAME	NAME_	nb_entrees
Crime	Televisiõn Española TVE	6701026
Crime		6701026
Drama	El Deseo	1617375
Drama	DreamWorks	14143624
Drama	Véa Digital	8526951
Drama	Hotshot Films	0
Drama	United Artists	14871212
Drama	France 3 Cinéma	0
Drama	P.P. Film Polski	0
Drama	Fine Line Features	6665199
Drama	Detour Film Production	921348

Résultat

1.3.3 TOP 10 des films qui ont généré le plus de revenu

Requête

```
1  SELECT fi.title
2  FROM d_film fi, fait fa
3  WHERE fa.d_film_id = fi.id
4      AND ROWNUM <= 10
5  ORDER BY fa.revenue;
```

Résultat

```
1  TITLE
2  --------------------------------------------------
3  Three Colors: Blue
4  Italian for Beginners
5  The Rolling Stones: Gimme Shelter
6  No End
7  Run Lola Run
8  Lock, Stock and Two Smoking Barrels
9  Match Point
10 Princess Mononoke
11 The Lord of the Rings: The Fellowship of the Ring
12 Finding Nemo
```

1.3.4 Entrées d'argent par genres en fonction de l'année

Requête

YEAR	GENRE_NAME	SUM(F.REVENUE)
		0
		3260793431
	NULL	0
	Crime	40266982
	Drama	514209432
	Action	488181621
	Comedy	37856188
	Adventure	1030743672
	Animation	940335536
	Documentary	0
	Science Fiction	209200000
2000		40031879
2000	Drama	40031879
2000	Comedy	0

```
1 SELECT dt.year, dg.genre_name, SUM(f.revenue)
2 FROM fait f, d_time dt, d_genre dg
3 WHERE f.d_time_id = dt.id
4     AND f.d_genre_id = dg.id
5 GROUP BY CUBE (dt.year, dg.genre_name);
```

Cette requête nous permet d'avoir un regard assez complet sur les revenus des films, en croisant les genres et les années. Grâce à l'operateur `CUBE`, nous avons aussi les revenus sur les années tout genres confondus et sur les genres, indépendamment des années.

Résultat

1.3.5 Réussite moyenne (entrées, revenu, popularité, vote moyens) et nombre de films des boites de production, par trimestre

Requête

```
1 SELECT dt.season, dt.year, dc.name_, AVG(f.admissions),
2   AVG(f.popularity), AVG(f.revenue), AVG(f.vote_average),
3   count(f.id)
4 FROM fait f, d_time dt, d_company dc
5 WHERE f.d_time_id = dt.id
6     AND f.d_company_id = dc.id
7 GROUP BY ROLLUP(dc.name_, dt.year, dt.season);
```

Ici, nous demandons un compte rendu complet de la réussite moyenne des compagnies de production. Chaque indicateur (moyenne des entrées, revenu, popu-

larité, votes) est calculé non seulement pour chaque entreprise, mais aussi pour chaque entreprise pour chaque année (où au moins un film à été produit), et pour chaque année pour chaque saison, grâce au fonctionnement de l'opérateur `ROLLUP`

Résultat

Certaines colonnes ont été retirées pour rendre le tableau lisible.

Saison	Année	Production	Nombre d'entrées moyen	Note moyenne
	1987	Paramount Pictures	49739707	6.1
SP	1989	Paramount Pictures	11769827	5.6
	1989	Paramount Pictures	11769827	5.6
		Paramount Pictures	21572331.3	6.375
WI	1998	Handmade Films Ltd.	4702374	7.5
	1998	Handmade Films Ltd.	4702374	7.5
		Handmade Films Ltd.	4702374	7.5
SU	2002	Les Films du Losange	0	6.8
	2002	Les Films du Losange	0	6.8
		Les Films du Losange	0	6.8
SU	1998	X-Filme Creative Pool	1211807	7.2

1.3.6 Classement des genres qui engendrent un temps moyen a l'affiche le plus long

Requête

```
1 SELECT ge.genre_name,avg(fa.runtime) AS "Moyenne de temps en
      salle (j)", rank() over (order by avg(fa.runtime) desc) AS "
      Rang"
2 FROM fait fa, d_genre ge
3 WHERE fa.d_genre_id = ge.id
4 GROUP BY ge.genre_name;
```

Cette requête utilise `rank() over`, qui fournit une numérotation correspondant à un classement.

Résultat

1.3.7 Cumul des budgets des films anglophones depuis 2000

Requête

GENRE_NAME	Durée Moyenne en salle (en jours)	Rang
NULL	210	1
Adventure	156	2
Science Fiction	119.5	3
Drama	110.07	4
Comedy	106.66	5
Crime	106	6
Action	97.33	7
Documentary	66.5	8
Animation	60.5	9

```
SELECT ti.year, sum(fa.budget) AS, sum(sum(fa.budget)) over(order
     by ti.year rows unbounded preceding)
FROM fait fa, d_time ti, d_zone zo
WHERE fa.d_time_id = ti.id AND fa.d_zone_id = zo.id AND ti.year
    >= 2000 AND zo.original_language = 'en'
GROUP BY ti.year;
```

On note ici l'utilisation d'une fenêtre glissante pour voir progressivement l'accumulation totale du budget investi dans l'industrie cinématographique au cours des années.

Résultat

```
\begin{table}[]
  \centering
  \texttt{
  \begin{tabular}{l l l}
  YEAR & SUM(FA.BUDGET) & budget cumul?? \\ \hline
  2000 & 12800000 & 12800000 \\
  2001 & 93000000 & 105800000 \\
  2003 & 124000000 & 229800000 \\
  2004 & 2700000 & 232500000 \\
  2005 & 20000000 & 252500000 \\
  2006 & 42000 & 252542000 \\

  \end{tabular}
}
\end{table}
```

1.3.8 Categorisation des pays où sont produit les films générant le plus de revenus avec leur films

Requête

```
1 SELECT zo.production_country, NTILE(4) over(order by sum(fa.
      revenue) desc) as "Groupe"
2 FROM fait fa, d_zone zo
3 WHERE fa.d_zone_id = zo.id
4 GROUP BY zo.production_country;
```

Pour finir, une division en quatre parties avec `NTILE(4)` des pays proposant les films les plus rentables.

Résultat

PRODUCTION_COUNTRY	Groupe
United States of America	1
New Zealand	1
Japan	1
Spain	1
Ireland	2
Argentina	2
United Kingdom	2
Germany	2
Canada	3
Austria	3
Denmark	3
Poland	4
France	4

Chapitre 2

Annexe

2.1 Script de récolte des données

2.1.1 Requêtes à l'API

```
1  #! /usr/local/bin/python3
2
3  import urllib.request
4  import urllib.parse
5  import json
6
7  movies_id = [] # Trop long pour le rapport
8  movies_array = []
9  api_key = "40d11b52ba315a0c925a73ee719f595d"
10
11 for id in movies_id:
12   request = "https://api.themoviedb.org/3/movie/"+ id + "?api_key
         =" + api_key
13   print("issuing request: " ,request)
14
15   results = urllib.request.urlopen(request)
16   file_w = open("./movies/" + id + ".json","w+")
17   file_w.write(results.read().decode("utf-8"))
18   print(results.read().decode("utf-8"))
19   movies_array.append(results.read().decode("utf-8"))
20   file_w.close()
```

2.1.2 Compilation des données dans un seul fichier

Afin d'avoir des données du nombre d'entrées, nous en générons aléatoirement en se basant sur le chiffre d'affaire du film.

```
#! /usr/local/bin/python3

import os, json, random

directory_in_str = "/Users/hippolytedubois/Documents/MASTER/M1/BDD/Projet_OLAP/API/movies"
directory = os.fsencode(directory_in_str)

movies_arr = []

for file in os.listdir(directory):
  filename = os.fsdecode(file)
  f = open(directory_in_str + '/' + filename)
  json_o = json.loads(f.read())
  json_o["admissions"] = int(json_o["revenue"] / 6 + ((json_o["revenue"] / 500) * (random.random() - 0.5)))
  movies_arr.append(json_o)
  f.close()

f = open("every_movies.json","w+")
f.write(json.dumps(movies_arr, sort_keys=True, indent=4, separators=(',', ': ')))
f.close()
```

2.2 Création de la base de donnée

```
CREATE TABLE d_zone (
  id NUMBER PRIMARY KEY,
  original_language VARCHAR(2),
  production_country VARCHAR(24)
);

CREATE TABLE d_time (
  id NUMBER PRIMARY KEY,
  release_date DATE,
  month NUMBER, -- 01, 02...12
  season VARCHAR(2), -- FA, WI, SP, SU
  year NUMBER,
  decade NUMBER -- 1910, 1920... 2010
```

```
14 );
15
16 CREATE TABLE d_genre (
17   id NUMBER PRIMARY KEY,
18   genre_name VARCHAR(15),
19   adult NUMBER(1,0) -- Adult = 1, kid = 0
20 );
21
22 CREATE TABLE d_film (
23   id NUMBER PRIMARY KEY,
24   title VARCHAR(49),
25   overview VARCHAR(955),
26   imdb_id VARCHAR(9),
27   original_title VARCHAR(49),
28   tagline VARCHAR(136),
29   status_ VARCHAR(8)
30 );
31
32 CREATE TABLE d_company (
33   id NUMBER PRIMARY KEY,
34   name_ VARCHAR(43)
35 );
36
37 CREATE TABLE fait (
38   id NUMBER PRIMARY KEY,
39   admissions NUMBER,
40   popularity FLOAT,
41   revenue NUMBER,
42   runtime NUMBER,
43   budget NUMBER,
44   vote_average FLOAT,
45   vote_count NUMBER,
46
47   -- FOREIGN KEYS
48   d_zone_id NUMBER, -- origin
49   d_time_id NUMBER, -- release date
50   d_genre_id NUMBER,
51   d_film_id NUMBER,
52   d_company_id NUMBER,
53
54   CONSTRAINT fk_d_zone FOREIGN KEY (d_zone_id) REFERENCES d_zone(
         id),
55   CONSTRAINT fk_d_time FOREIGN KEY (d_time_id) REFERENCES d_time(
         id),
56   CONSTRAINT fk_d_genre FOREIGN KEY (d_genre_id) REFERENCES
         d_genre(id),
```

```
57   CONSTRAINT fk_d_film FOREIGN KEY (d_film_id) REFERENCES d_film(
         id),
58   CONSTRAINT fk_d_company FOREIGN KEY (d_company_id) REFERENCES
         d_company(id)
59 );
```

2.2.1 Conversion des données au modèle de la base de donnée

Ce script Python génère un fichier d'instructions SQL qui remplit la base de donnée. On remplie un fichier au moyen d'une redirection du stdout shell.

```
1  #! /usr/local/bin/python3
2
3  import os, json
4
5  fname = "../API/every_movies.json"
6  seasons_arr = ["WI", "SP", "SU", "FA"]
7
8  insert_zone = "INSERT INTO d_zone(id, original_language,
       production_country) VALUES ({}, {}, {});\n"
9  insert_time = "INSERT INTO d_time(id, release_date, month, season
       , year, decade) VALUES ({}, to_date({} , \'YYYY-MM-DD\'), {},
        {}, {}, {});\n"
10 insert_genre = "INSERT INTO d_genre(id, genre_name, adult) VALUES
        ({}, \'{}\', {});\n"
11 insert_film = "INSERT INTO d_film(id, title, overview, imdb_id,
       original_title, tagline, status_) VALUES ({}, \'{}\', \'{}\',
        \'{}\', \'{}\', \'{}\', \'{}\');\n"
12 insert_company = "INSERT INTO d_company(id, name_) VALUES ({},
       \'{}\');\n"
13 insert_fait = "INSERT INTO fait(id, admissions, popularity,
       revenue, runtime, budget, vote_average, vote_count, d_zone_id
       , d_time_id, d_genre_id, d_film_id, d_company_id) VALUES ({},
        {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {}, {});\n"
14
15 if os.path.exists(fname):
16 with open(fname, 'r') as f:
17 json_arr = json.load(f)
18 id = 0
19 for movie in json_arr:
20 # Zone insert
21 country = "NULL"
22 if movie['production_countries']:
23 country = "\'" + movie['production_countries'][0]['name'] + "\'"
```

```
print(insert_zone.format(id, "\'" + movie['original_language'] + "\'", country))

# Date insert
date = "NULL"
month = "NULL"
season = "NULL"
year = "NULL"
decade = "NULL"
if movie['release_date']:
date = "\'" + movie['release_date'] + "\'"
month = movie['release_date'].split('-')[1]
season = "\'" + seasons_arr[int(movie['release_date'].split('-')[1]) // 4] + "\'"
year = movie['release_date'].split('-')[0]
decade = movie['release_date'].split('-')[0][:3] + "0"
print(insert_time.format(id, date, month, season, year, decade))

# Genre insert
genre_name = "NULL"
if movie['genres']:
genre_name = movie['genres'][0]['name']
adult = "0"
if movie['adult']:
adult = "1"
print(insert_genre.format(id, genre_name, adult))

# Film insert
print(insert_film.format(id, movie['title'], movie['overview'], movie['imdb_id'], movie['original_title'], movie['tagline'], movie['status']))

# Company insert
company = "NULL"
if movie['production_companies']:
company = movie['production_companies'][0]['name']
print(insert_company.format(id, company))

# Fact insert
print(insert_fait.format(id, movie['admissions'], movie['popularity'], movie['revenue'], movie['runtime'], movie['budget'], movie['vote_average'], movie['vote_count'], id, id, id, id, id))

id+=1
```

2.2.2 Exemple de commande INSERT ainsi générées

```
1  INSERT INTO d_zone(id, original_language, production_country)
       VALUES (0, 'en', 'United Kingdom');
2
3  INSERT INTO d_time(id, release_date, month, season, year, decade)
        VALUES (0, to_date('1998-03-05' , 'YYYY-MM-DD'), 03, 'WI',
       1998, 1990);
4
5  INSERT INTO d_genre(id, genre_name, adult) VALUES (0, 'Comedy',
       0);
6
7  INSERT INTO d_film(id, title, overview, imdb_id, original_title,
       tagline, status_) VALUES (0, 'Lock, Stock and Two Smoking
       Barrels', 'A card sharp and his unwillingly-enlisted friends
       need to make a lot of cash quick after losing a sketchy poker
        match. To do this they decide to pull a heist on a small-
       time gang who happen to be operating out of the flat next
       door.', 'tt0120735', 'Lock, Stock and Two Smoking Barrels', '
       A Disgrace to Criminals Everywhere.', 'Released');
8
9  INSERT INTO d_company(id, name_) VALUES (0, 'Handmade Films Ltd
       .');
10
11 INSERT INTO fait(id, admissions, popularity, revenue, runtime,
       budget, vote_average, vote_count, d_zone_id, d_time_id,
       d_genre_id, d_film_id, d_company_id) VALUES (0, 4702374,
       4.880012, 28356188, 105, 1350000, 7.5, 1678, 0, 0, 0, 0, 0);
12
13 INSERT INTO d_zone(id, original_language, production_country)
       VALUES (1, 'de', 'Germany');
14
15 INSERT INTO d_time(id, release_date, month, season, year, decade)
        VALUES (1, to_date('1998-08-20' , 'YYYY-MM-DD'), 08, 'SU',
       1998, 1990);
16
17 INSERT INTO d_genre(id, genre_name, adult) VALUES (1, 'Action',
       0);
18
19 INSERT INTO d_film(id, title, overview, imdb_id, original_title,
       tagline, status_) VALUES (1, 'Run Lola Run', 'Lola receives a
        phone call from her boyfriend Manni. He lost 100,000 DM in a
        subway train that belongs to a very bad guy. She has 20
       minutes to raise this amount and meet Manni. Otherwise, he
       will rob a store to get the money. Three different
       alternatives may happen depending on some minor event along
```

```
    Lolas run.', 'tt0130827', 'Lola rennt', 'Every second of
    every day youre faced with a decision that can change your
    life.', 'Released');
20
21  INSERT INTO d_company(id, name_) VALUES (1, 'X-Filme Creative
    Pool');
22
23  INSERT INTO fait(id, admissions, popularity, revenue, runtime,
    budget, vote_average, vote_count, d_zone_id, d_time_id,
    d_genre_id, d_film_id, d_company_id) VALUES (1, 1211807,
    7.326948, 7267585, 81, 1530000, 7.2, 678, 1, 1, 1, 1, 1);
```