Université de Cergy-Pontoise
Faculté des Sciences et Techniques
Département d'Informatique
Master 1 IISC

Rapport de projet
Intégration et Entrepôt de données

29 Avril 2019

Réalisé par :
Hong Phuc VU
Thiziri TENSAOUT

Encadré par :
Dan Vodislav
Manos Katsomallos

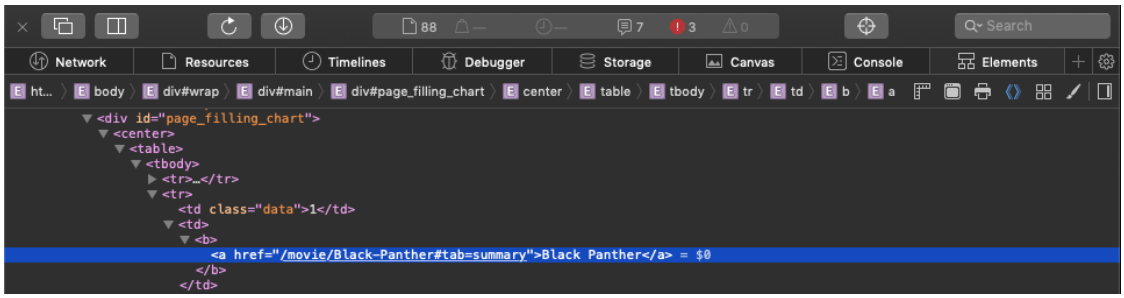
Table des matières

1. Extraction de données avec Jsoup

Il faut extraire les informations sur les films à partir d'un tableau HTML. L'accès à l'URL sous la forme suivante génère les informations du film dans un tableau HTML

```
http://www.the-numbers.com/market/<année>/genre/<Genre>
```

Prend un exemple pour l'URL https://www.the-numbers.com/market/2018/genre/action
Cet URL nous permet de consulter les films du genre action de l'année 2018. On inspecte le contenu du page web grâce à un outil intégré dans le navigateur, les informations visibles sur la page peuvent facilement se trouver dans le code source. Le code source est comme suit



En effet, le résultat de la recherche situé dans un tableau sous la section div avec id : page_filling_chart. On stocke le tableau dans un variable de type Elements dans Jsoup

```
Elements table = doc.select("div#page_filling_chart table tbody tr")
```

Chaque ligne y compris l'entête est englobé par la balise <tr>. On s'intéresse que les données du film donc on ne prend pas la première <tr> qui est l'entête de tableau et les deux dernières lignes qui sont des informations inutiles. La sélection avec Jsoup est donc

```
doc.select("tr:gt(0)" + ":lt" + "(" + (table.size()-2) + ")" )
```

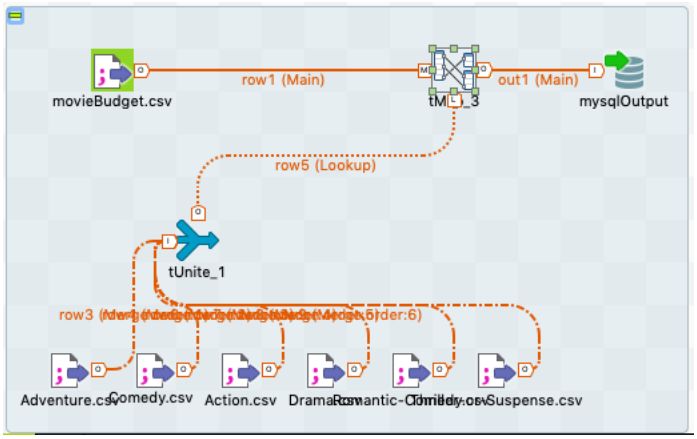
On avait 7 éléments dans une ligne divisé par les colonnes <td>, L'élément Rank pour l'indice 0, le titre du film pour l'indice 1, la date de sortie pour l'indice 2, le distributeur pour celle de 3, le rating pour l'indice 4, le revenu pour la 5 et les billets vendues pour la 6. La commande :eq() permet de préciser l'indice à laquelle qu'on veut récupérer la valeur.

2. Transformation et stockage de données

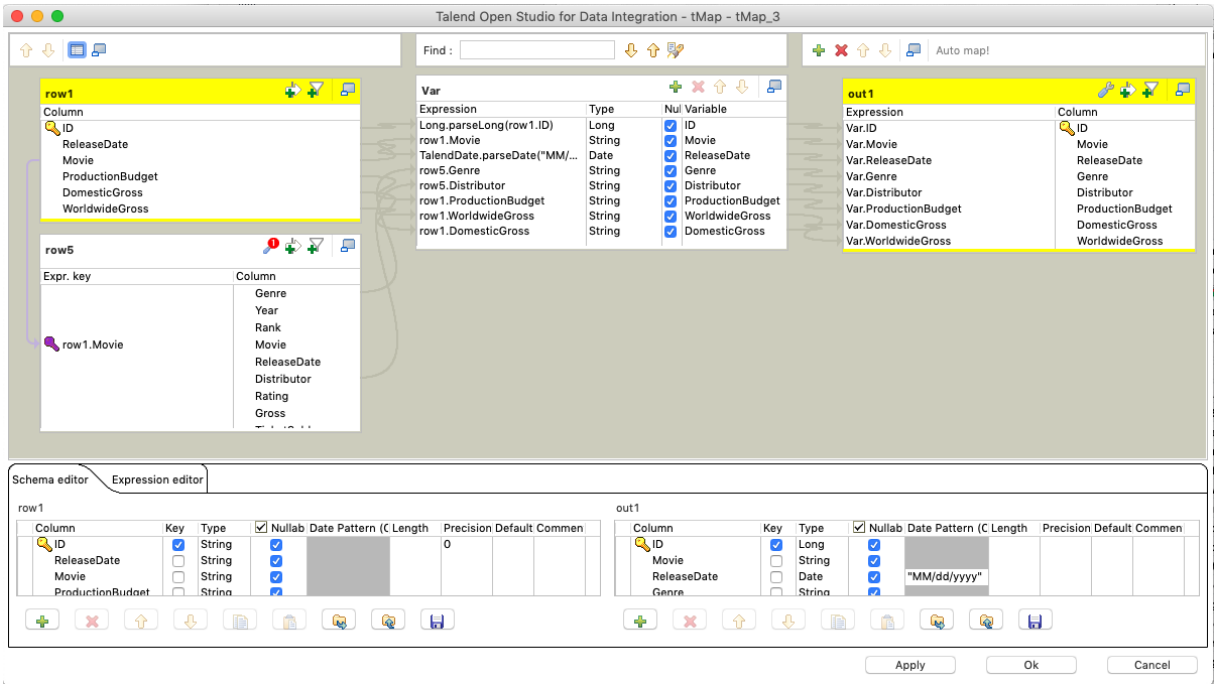
2.1. TALEND

Le job pour transformer les données et envoyer à la base de données ont les éléments suivants

- Un tFileInputDelimited pour extraire les données d'un fichier csv, on utilise \t comme déliminateur. Pour éviter le caractère d'espace qui peut fausser le résultat, on choisir l'option « Trim all columns » dans Advanced settings.
- Six tFileInputDelimited pour les fichiers scrappés par Jsoup. Ces fichiers se servent à compléter l'information de genre et de la distribution.
- tUnite pour fusionner les données des fichiers triés par genre
- Un tMap pour mapper les données provenues de fichiers csv différents et envoyer l'information à la base de données MySQL
- Un tDBOutput mysqlOutput pour stocker les résultats traités par tMap dans une base de données de type MySQL



On fait la jointure sur le titre du film Movie entre le tableau row1 de fichier movieBudget et le tableau row5 de la fusion de 6 fichiers caractérisé par genre. On transforme l'id du film en type Long et la date de sortie ReleaseDate dans la table Var. On liée les informations besoins pour alimenter la base de donnée de la table Var à la table out1 qui s'agit le schéma de la base de données MySQL. Le mapping est comme la photo ci-dessus



3. Médiateur & Source

Le mapping entre le médiateur et les trois sources est décomposé par source

3.1. DBpedia

La classe permet de récupérer l'information de DBPedia est LinkedMovieJob. A l'aide de l'API Jena, les méthodes pour extraire les données sur le titre du film, le réalisateur, l'acteur et le producteur sont les suivants

directorQuery(String movieName, String lang): Cette méthode s'effectue une requête SPARQL qui retourne le nom du réalisateur à partir de nom du film dans lequel qu'il dirige

Requête effectuée

```
SELECT DISTINCT ?director
```

```
WHERE {
?film a <http://dbpedia.org/ontology/Film> ;
foaf:name ?movieLiteral ;
dbo:director ?directorObject .
?directorObject foaf:name ?director .
}
```

producerQuery(String movieName, String lang): producerQuery fait une requête en SPARQL permettant de récupérer la nom du producteur par le nom du film en paramètre
Requête effectuée

```
SELECT DISTINCT ?producer
WHERE {
?film a <http://dbpedia.org/ontology/Film> ;
foaf:name ?movieLiteral ;
dbo:producer ?producerObject .
?producerObject foaf:name ?producer .
}
```

actorQuery(String movieName, String lang): Elle passe le nom du film et sa langue en paramètre et retourne le nom d'acteur en exécutant la requête
Requête effectuée

```
SELECT DISTINCT ?actor
WHERE {
?film a <http://dbpedia.org/ontology/Film> ;
foaf:name ?movieLiteral ;
dbo:starring ?actorObject .
?actorObject foaf:name ?actor .
}
```

movieQuery(String actor, String lang): Cette méthode prend le nom d'acteur en paramètre et retourne à la fin le nom du film auquel il joue
Requête effectuée

```
SELECT DISTINCT ?movie
WHERE {
?film a <http://dbpedia.org/ontology/Film> ;
dbo:starring ?actorObject ;
foaf:name ?movie .
?actorObject foaf:name ?actor .
```

3.2. Open Movie Database

Afin de récupérer l'information de ce site. Il suffit d'appeler le service REST par un appel GET avec l'exemple d'URI suivant
http://www.omdbapi.com/?apikey=8f60a1f&r=xml&t=Iron+Man
Le service nécessite une clé API qui s'agit le paramètre apikey dans l'URI pour utiliser. On précise également le format de résultat retourné est XML. Il facilite l'interrogation de donnée par XPath. On met enfin le titre du film recherché avec le paramètre t.

On dispose la classe Java XPathQuery pour extraire l'information sur le résumé. Elle contient les méthodes ci-dessus :

Object XPath(String uri, String requete, Qname typeRetour) prend en paramètre l'URI pour le service REST, le chemin XPath pour interroger la donnée une fois qu'on obtient le résultat sous forme XML et le type de retour typeRetour. Cette méthode sort toujours le résultat en type Object dans Java. Il faut transformer plus tard cet objet en type indiqué par typeRetour pour obtenir la donnée

void getXPathResult(MovieInfo mi, String movie)
Pour extraire la donnée de la méthode XPath(), on utilise getXPathResult()

La requete XPath à mettre dans la méthode XPath() pour récupérer la résumé est

```
/root/movie/@plot
```

On transforme ensuite le résultat de XPath() en type Node et utilise getTextContent pour récupérer l'information de type String

3.3. JDBC MySQL

Pour accéder la base de données locale MySQL, on utilise l'API JDBC

MovieInfo readDataBase(String movie) effectue une requête SQL avec la condition imposé sur le nom du film. La requête effectuée en SQL avec le nom du film $movie est la suivante :

```
SELECT * FROM Film WHERE Movie LIKE '$movie'
```

3.4. Médiateur

Le médiateur s'agit MainProgram dans Java. Il appelle les trois source Open Movie Database, DBPedia et base de données locale. Les méthodes ci-dessus sont utilisées

MovieInfo searchByName(String movie, String lang)

Elle s'affiche toutes les informations disponibles sur le film

On instance la variable jdbc pour la classe JDBCJob. Cela permet de récupérer l'information depuis la base de données locale. La variable xq pour la classe XPathQuery est créer pour extraire la donnée provenue de Open Movie Database. On met en place également la variable l de la classe LinkedMovieJob. Elle permet d'effectuer la requête SPARQL pour récupérer l'information de DBPedia. Les résultats extraits de ces trois sources sont stockés dans un objet MovieInfo. Au premier temps, on tente d'obtenir le titre, la date de sortie, la genre, le distributeur, le budget, la revenue domestique aux États-Unis et la revenue mondiaux en utilisant readDataBase() par jdbc. Pour la résumé, on appelle getXPathResult(). On utilise successivement producerResult(), directorResult(), actorResult() pour récupérer le producteur, le directeur et l'acteur

void searchByActor(String actor, String lang)

Cela permet d'afficher la liste des films ou l'acteur a joué. On crée un ArrayList movieResults pour stocker le nom des films auxquels l'acteur joue. Cette requête elle nécessite que l'appel de la classe LinkedMovieJob pour extraire l'information de film ou joue l'acteur à partir de la base DBPedia. movieQuery est utilisé. Pour chaque film obtenu dans la liste, on applique la méthode searchByName. Si cela renvoie le résultat autre que null, on l'affiche l'information du film, sinon on met un message pour dire qu'il n'y a aucune information pour le film recherché

void main(String[] args) : Le cœur du programme de médiateur.

Le programme prend l'entrée de l'utilisateur. S'il tape 'M' dans la console, c'est bien la requête de chercher les informations du film à partir le titre. On demande ensuite l'entrée sur le titre et appeler la méthode searchByName(). Le résultat sera affiché si il n'est pas null par showInfoMovie(). Si l'utilisateur choisit 'A' comme entrée, on fait l'autre requête. Le programme appelle la méthode searchByActor() et afficher le titre, la date de sortie, le genre, le distributeur, le réalisateur et le producteur pour chaque film ou l'acteur demandé joue