GUINLE
LOIC
BUT 3
Irit groupe 2

13/10/2023

Compte rendu d'activité

1. *Introduction*

Le but de ce projet est de créer **une interface graphique** pour un prototype d'interrogation de polystore, un système de gestion de données intégrant divers types de stockages. Le but est **d'améliorer la visualisation des requêtes et de pouvoir réécrire des sous-requêtes,** simplifiant ainsi l'accès aux données stockées de manière hétérogène dans ces systèmes.
L'application est réalisée en **Java** en utilisant la librairie **JavaFX** pour le côté graphique.

Lors de mon arrivée dans le projet en tant qu'alternant le projet était au niveau de la conception de la v1 avec la représentation graphique des arbres algébriques en fonction d'une requête renseignée.

Voici l'interface lors de notre arrivée :

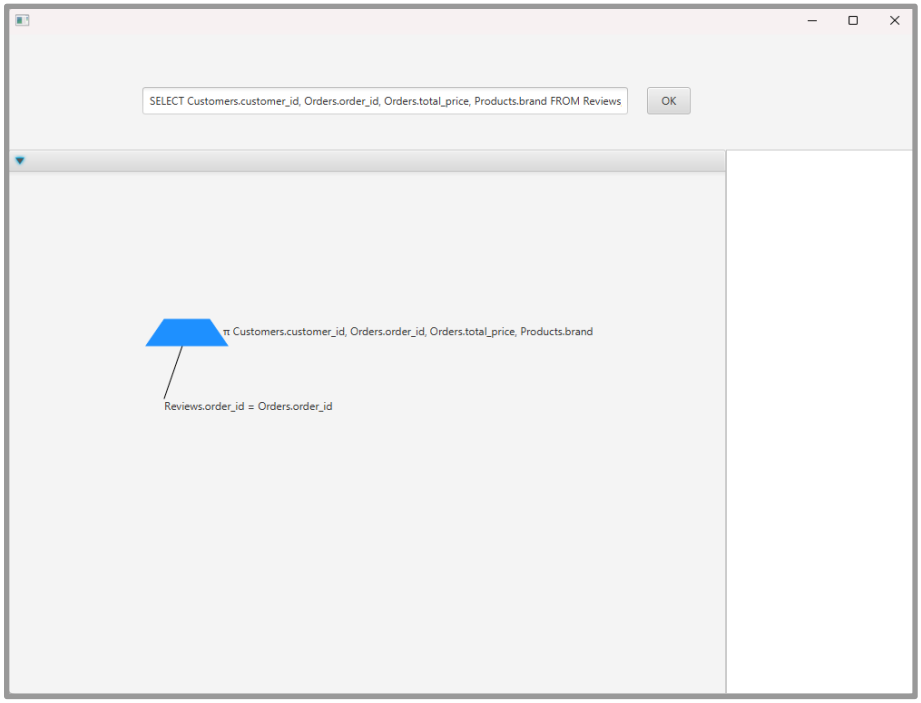

Figure 1 Visuel de l'application à notre arrivée

2. *Les issues*

L'issue principale qui m'a été attribuée est la représentation des arbres sous forme d'arborescence de fichier.
En effet le but était de proposer une **vue alternative** de l'arbre à l'utilisateur.
L'arborescence dépend du type d'arbres algébrique affiché
Pour ce faire j'ai utilisé le composant TreeView de javaFX.

https://github.com/AurelienSP/SAE-S5-IRIT-G2/issues/15.

Ce code permet d'alimenter un **TreeItem** qui servira de root au **TreeView** :
Pour ce faire j'utilise une **méthode récursive** qui va parcourir l'arbre et ajouter le nom du TreeNode au TreeItem.

```
/**
* Création d'une treeView avec comme paramètre node ainsi que la TreeItem parente
*
* @param node       noeud de l'arbre a partir duquel est crée un TreeItem
* @param treeParent TreeItem parent sur lequel on va add les enfants
*/
private void populateTreeView(ETreeNode node, TreeItem<String> treeParent) {
   if (node.getClass().equals(EProjection.class)) {
       //Boucle sur le nombre d'enfant de la projection initial
       int nbChild = node.getChild().length;
       for (int i = 0; i < nbChild; i++) {
           TreeItem<String> child = new TreeItem<>(node.getChild()[i].toString());
           child.setExpanded(true);
           treeParent.getChildren().add((child));
           this.populateTreeView(node.getChild()[i], child);
       }
   } else if (node.getClass().equals(EJoin.class)) {
       // Si le node est de type EJoin il a un left et un right child qu'on va créer en tant que
TreeItem et ajouter au parent
       TreeItem<String> childLeft = new TreeItem<>(( (EJoin) node).getLeftChild().toString());
       childLeft.setExpanded(true);
       TreeItem<String> childRight = new TreeItem<>(((EJoin) node).getRightChild().toString());
       childRight.setExpanded(true);
       treeParent.getChildren().addAll(childLeft, childRight);

       populateTreeView(((EJoin) node).getRightChild(), childRight);
       populateTreeView(((EJoin) node).getLeftChild(), childLeft);
   } else if (node.getClass().equals((ESelection.class))) {
       // Si le node est de type ESelection il n'aura qu'un enfant
       TreeItem<String> child = new TreeItem<>(node.getChild()[0].toString());
       child.setExpanded(true);
       treeParent.getChildren().add(child);

       populateTreeView(node.getChild()[0], child);
   }
}
```

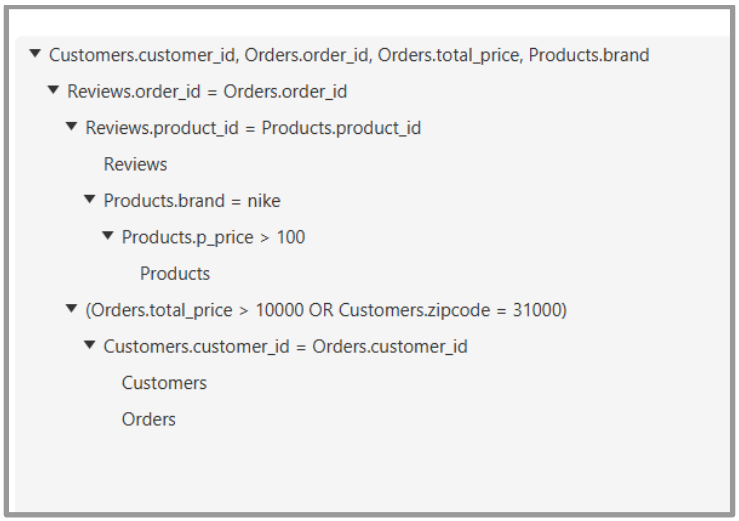

Figure 2 Arborescence

J'ai aussi participé à la résolution de cette issue avec deux de mes camarades :

https://github.com/AurelienSP/SAE-S5-IRIT-G2/issues/27

Le but était de **clarifier le code** et de le rendre plus compréhensible (ajout de commentaires, refactorisation, nettoyage global du code).

https://github.com/AurelienSP/SAE-S5-IRIT-G2/issues/40

L'autre issue sur laquelle j'ai travaillé est la **représentation structurelle des bases**. L'ajout de cette fonctionnalité a été demandé par le client après une réunion afin **d'aider l'utilisateur dans son requêtage.**
Les différentes bases sont représentées dans un fichier JSON « relationnelUnifiedView.json » que j'ai parser grâce à la librairie **JACKSON** pour pouvoir l'exploiter.
Pour pouvoir générer la structure j'ai réalisé une **méthode récursive** permettant de parcourir les différents niveaux, mais le souci rencontré est que le nom des tables n'est pas au même niveau que les données relatives à celles-ci. Pour remédier à ce problème j'ai dû créer deux tableaux, un contenant les noms des tables et un autre contenant leurs informations (la base d'origine, nom des colonnes, types)

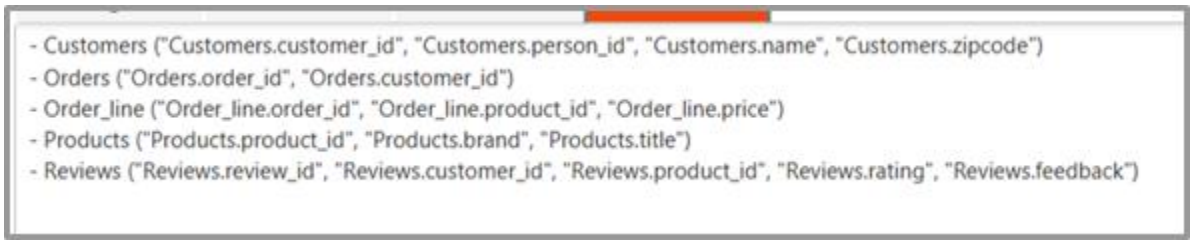

Figure 3 Structure des bases selon le fichier json

3. *Problèmes rencontrés*

Les principaux soucis rencontrés ont été la **difficulté de compréhension du sujet** accompagné du manque d'expérience sur les arbres algébriques ainsi que sur le langage Java.

4. *Conclusion*

Ce projet m'a permis de découvrir les arbres algébriques ainsi qu'un travail d'équipe hors pair. Malgré la complexité du sujet, le travail accompli a satisfait le client. Je crois qu'une fusion des deux projets, en incorporant les meilleures idées de chaque groupe, pourrait aboutir à **une application qui dépasserait les attentes du client**.

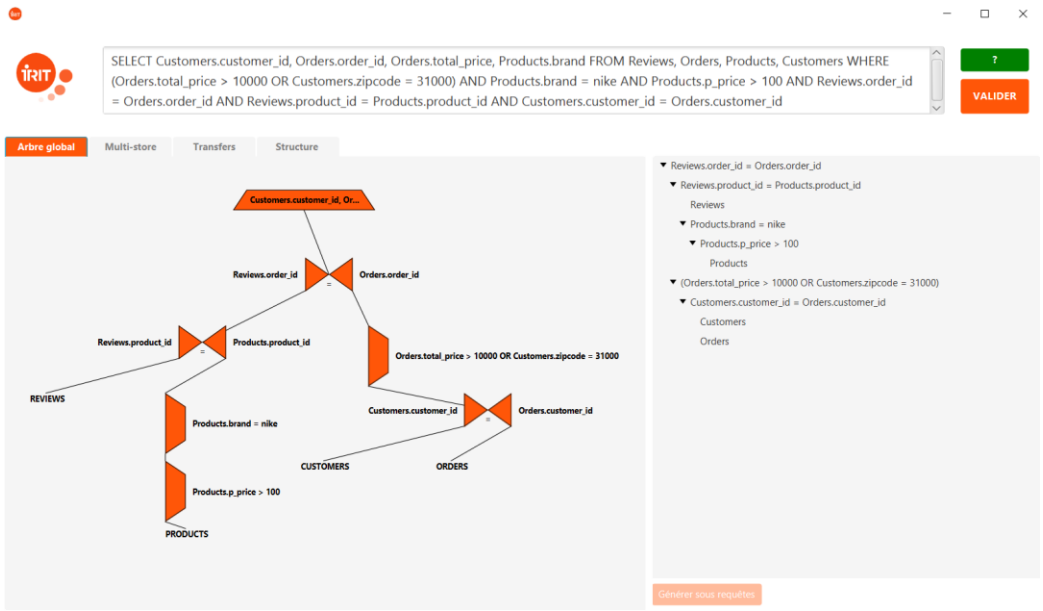

Figure 4 Visuel de l'application à notre départ en entreprise