Requêtes en SQL et leur graphe de dépendance

Projet Long 2019/2020

1. Introduction

- **Les objectifs du projet**
  Quel est le but du projet ?
- **Que fait notre logiciel ?**
  En quoi est-il utile pour répondre au sujet du projet ?
- **Comment peut-on l'utiliser ?**
  Différent scénarios d'utilisation, exemple d'utilisation ....

Les objectifs du Projet :

Le but de ce projet est de fournir un outil d'analyse statique de base de données.

Celui-ci doit procéder à l'analyse de requêtes SQL en passant par la génération du graphe de dépendance de ces dernières.

Les objectifs du Projet :

Quels étaient les principaux jalons pour ce projet :

- Analyser un ensemble de requêtes en calculant leur graphe de dépendance
- Créer une Interface utilisateur permettant d'analyser le graphe produit
- Ecrire un programme qui prend en entrée un graphe de dépendance et qui, en sortie, informe du nombre de cycle critique présent dans le graphe

Que fait notre logiciel ?

Les fonctionnalités de l'application fournis avec le sujet du projet sont :

- l'étude de requêtes écrites en un langage proche de PGSQL
- exportation du résultat sous forme de graphe au format pdf ou dot

Les fonctionnalités de l'application réalisées pour cette année de PLONG sont:

- l'étude des requêtes écrites en langage PL/PGSQL
- génération d'un graphe dans une Interface Utilisateur
- étude du graphe sous divers critères
- exportation du graphe généré sous divers formats :

pdf, svg, png, gif, jpg, jpeg, xdot, dot, ps, fig, json.

Quels sont les principaux scénarios d’utilisation ?

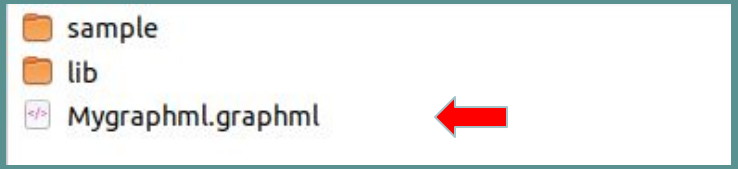
Sous forme de récit utilisateur, les principaux scénarios d’utilisations sont :

- **En tant qu’utilisateur, je veux visualiser un graphe afin d’analyser le contenu d’un fichier .graphml**

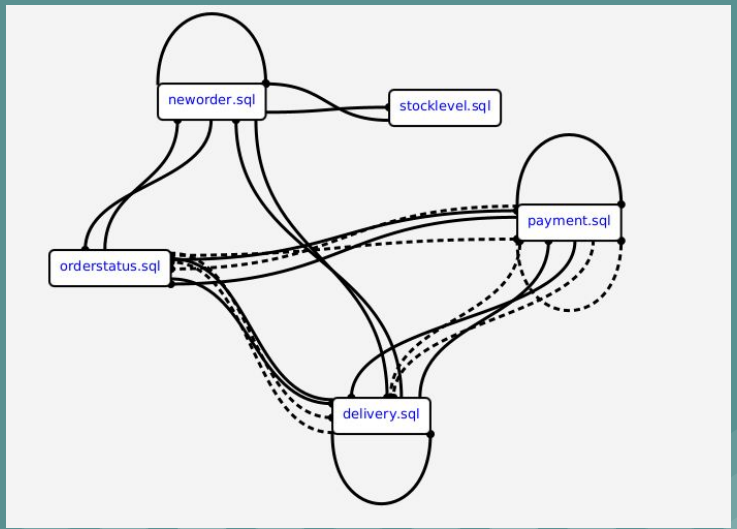
- **En tant qu’utilisateur, je veux pouvoir générer un graphe de dépendance afin d’analyser des Transactions SQL.**

<u>User story 1 :</u> En tant qu'utilisateur, je veux visualiser un graphe afin d'analyser le contenu d'un fichier .graphml

1) Choix d'un fichier :

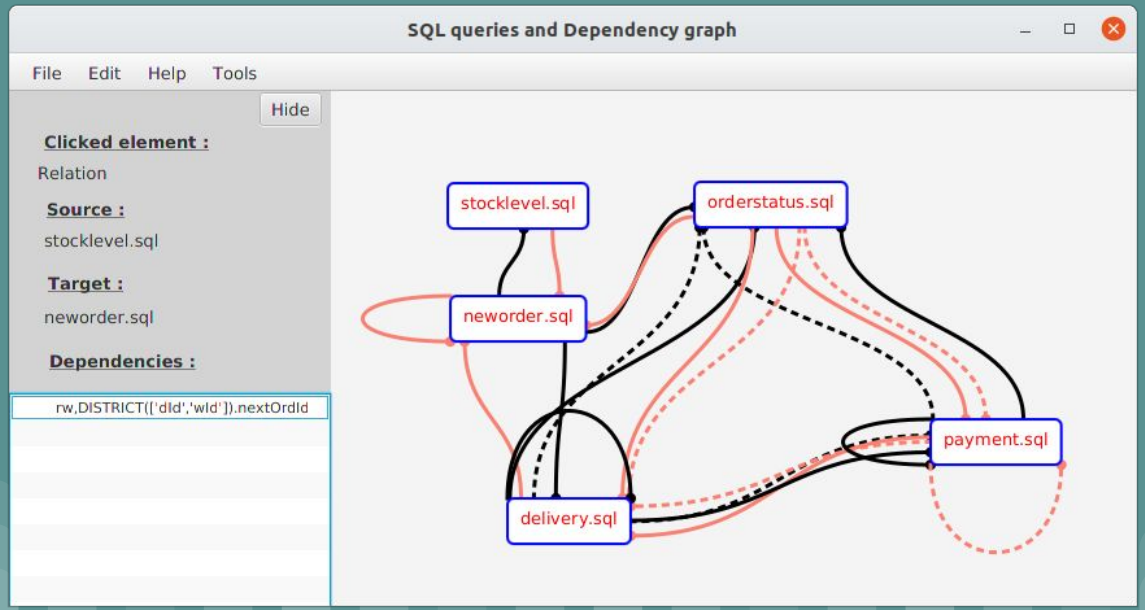


2) génération du graphe correspondant



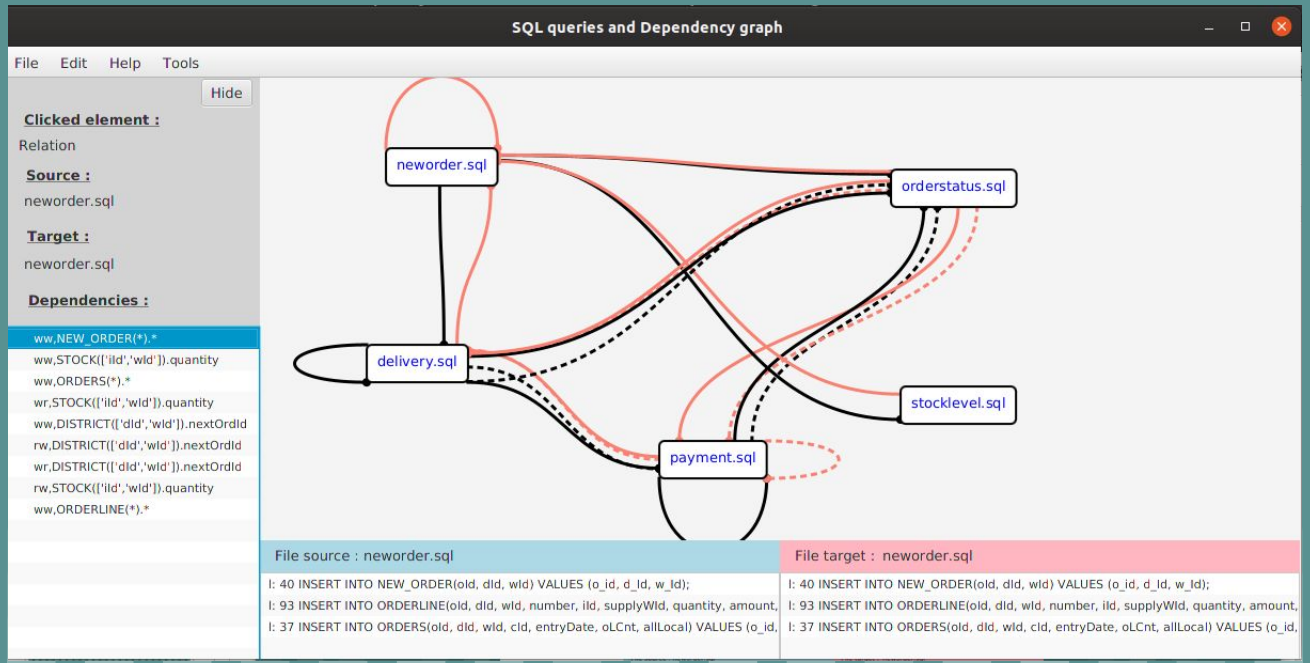
User story 2 : En tant qu'utilisateur, je veux pouvoir générer un graphe de dépendance afin d'analyser des Transactions SQL

1) Génération d'un graphe de dépendance à partir d'un ensemble de Transactions



<u>User story 2 :</u> En tant qu'utilisateur, je veux pouvoir générer un graphe de dépendance afin d'analyser des Transactions SQL

1) Génération d'un graphe de dépendance à partir d'un ensemble de Transactions
2) **Etude du graphe et de dépendance**

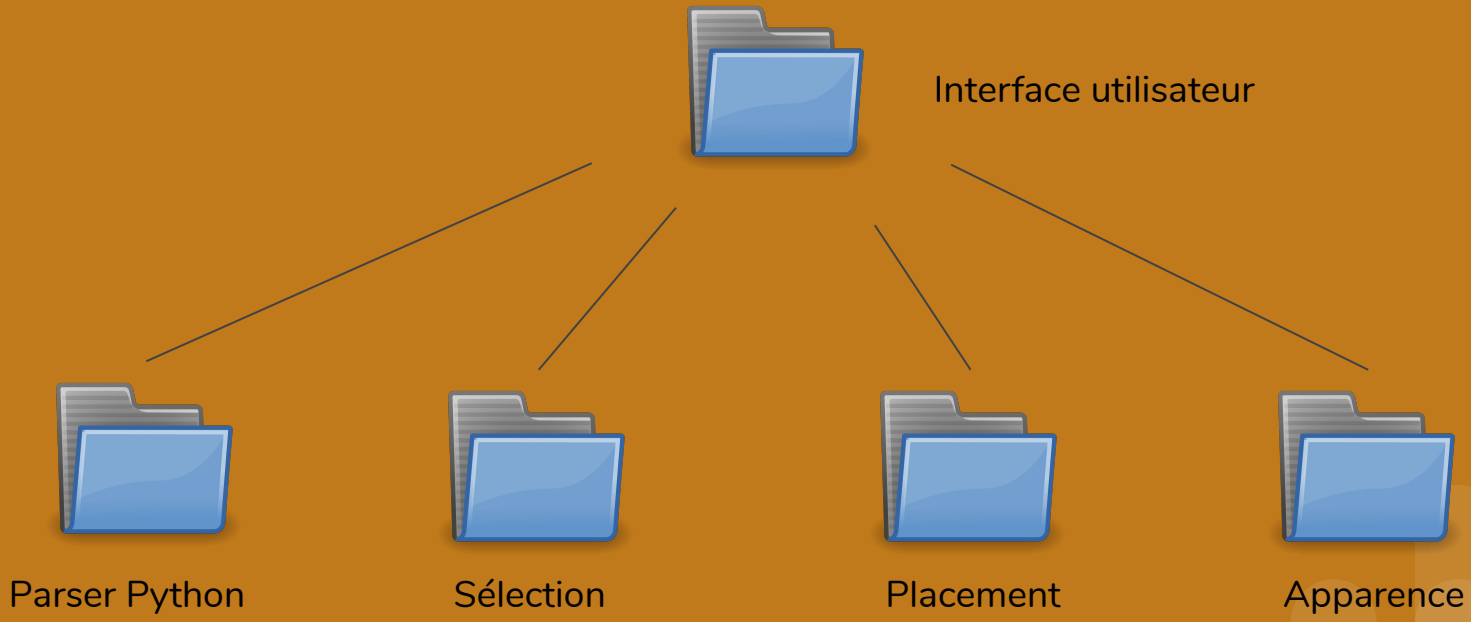


2. Conception

- **Architecture du Projet**
  Vue d'ensemble du logiciel
- **Conception du Projet**
  Différent choix d'implémentations, comment le problème principal a-t-il été décomposé?

Architecture du projet :

Le logiciel dispose d'une architecture hiérarchique.



Leurs rôles ?

Conception du projet :

A quel problème nouveau doit répondre le logiciel ?

Le logiciel doit permettre l'analyse statique d'une base de données.

A quels problèmes important doit répondre le logiciel ?

La génération d'un graphe de dépendance à partir d'un ensemble de requêtes (transactions) sur une base de donnée.

Comment se décomposent ces problèmes ?

- analyse des requêtes
- création d'une interface pour étudier un graphe
- génération d'un graphe :
    - gestion de la configuration initiale du graphe
    - gestion de l'étude du graphe

Conception du Projet:

Le Parser Python :

- prend en entrée une liste de fichier SQL, dont :
  - un fichier de création de la Base de Données nommé : GenDB.sql
  - un ensemble de fichiers contenant les requêtes et les transactions de la Base de Données
- rend en sortie un ensemble de 2 fichiers :
  - un fichier correspondant au graphe de dépendances (avec .graphml en extension)
  - un fichier répertoriant l'ensemble des raisons de dépendances (avec .gogol en extension)

L'Interface utilisateur :

- prend une des deux options ci dessous en entrée :
  - un répertoire contenant un ensemble de fichier .sql comme décrit pour le parseur
  - un fichier avec l'extension .graphml
- propose un graphe de dépendance (modifiable)
- propose un fichier lisible contenant le graphe de dépendance

Conception du Projet:

Le Parser génère deux fichiers, un fichier .graphml et un fichier .gogol .

Dans l'Interface, l'utilisateur à le choix pour générer un graphe de dépendance entre :

- lancer une analyse d'un ensemble de fichiers .sql qui seront envoyés au Parser
- lancer l'affichage d'un fichier .graphml contenant les informations nécessaire sans passer par les calculs du Parser

Conception du Projet:

L'Interface utilisateur du logiciel permet un ensemble de fonctionnalitées dont :

- générer un graphe dont chaque partie est mobile
- analyser le graphe selon plusieurs critères :
    - le type de dépendances que l'on veut mettre en évidence
    - les différentes Transactions que l'on souhaite étudier
- exporter le graphe généré sous divers format :

    pdf, svg, png, gif, jpg, jpeg, xdot, dot, ps, fig, json

Durée de développement ?

Organisation de l'équipe ?

3. Programmation

- **Présentation d'une partie du code**
  Présentation d'une partie intéressante de l'implémentation du projet
- **Objectifs réalisés**
  Présentation des objectifs atteint ainsi que ceux laissés de côté.

Objectifs réalisés :

Les jalons ou objectifs atteint sont:

- l'écriture d'un Parseur analysant des requêtes PL/PGSQL
- l'implémentation d'une Interface Utilisateur
    - génération d'un graphe modifiable et lisible par l'utilisateur
    - mise en évidence de certains types de relations de dépendance
    - recherche et mise en évidence des raisons de dépendance
    - exportation du graphe généré dans divers format

Présentation d'une partie du code :

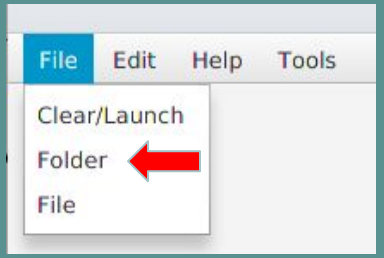
```java
/**
 * applies a loop turn of Fruchterman-Reingold algorithm
 *
 * @param cooling   value to subtract to the temperature at each loop turn
 * @param k ideal size of edge
 */
private void manageFRPlacement(double cooling, double k) {
    /* repulsion of each node between them */
    for (Node node : nodes) {
        node.setDisp(0, 0);
        for (Node node1 : nodes) {
            if (!node.equals(node1)) {
                double dx = node.x - node1.x;
                double dy = node.y - node1.y;
                double delta = Math.sqrt((dx * dx) + (dy * dy));
                if (delta != 0) {
                    double d = repulsiveForce(delta, k) / delta;
                    node.addDisp(dx * d, dy * d);
                }
            }
        }
    }
    /* calculate attractive forces (only between neighbors) */
    /* attraction due to the links between two nodes*/
    for (Edge e : edges) {
        double dx = e.v.x - e.u.x;
        double dy = e.v.y - e.u.y;
        double delta = Math.sqrt(dx * dx + dy * dy);
        if (delta != 0) {
            double d = attractiveForce(delta, k) / delta;
            double ddx = dx * d;
            double ddy = dy * d;
            e.v.addDisp(-ddx, -ddy);
            e.u.addDisp(+ddx, +ddy);
        }
    }
    optimum = true;
    /* node displacement affectation */
    for (Node v : nodes) {
        double dx = v.dx, dy = v.dy;
        double delta = Math.sqrt((dx * dx) + (dy * dy));
        if (delta != 0) {
            double d = Math.min(delta, temperature) / delta;
            double x = v.x + dx * d, y = v.y + dy * d;
            x = Math.min(width, Math.max(0, x)) - width / 2;
            y = Math.min(heigth, Math.max(0, y)) - heigth / 2;
            v.setPos(Math.min(Math.sqrt((width * width / 4) - (y * y)),
                    Math.max(-Math.sqrt((width * width / 4) - (y * y)), x)) + (width / 2),

                    Math.min(Math.sqrt(Math.abs((heigth * heigth / 4) - (x * x))),
                            Math.max(-Math.sqrt(Math.abs((heigth * heigth / 4) - (x * x))),
                                    y))+ (heigth / 2));
        }
    }
    temperature -= cooling;
    iterations++;
}
```

Algorithm 2: Fruchterman-Reingold

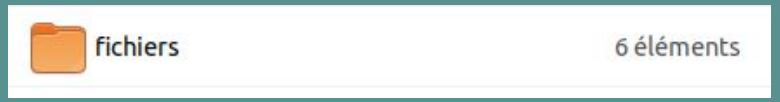
$area \leftarrow W * L$; /* frame: width W and length L */
initialize $G = (V, E)$; /* place vertices at random */
$k \leftarrow \sqrt{area/|V|}$; /* compute optimal pairwise distance */
function $f_r(x) = k^2/x$; /* compute repulsive force */
for $i = 1$ **to** *iterations* **do**
  foreach $v \in V$ **do**
    $v.disp := 0$; ; /* initialize displacement vector */
    for $u \in V$ **do**
      if $(u \neq v)$ **then**
        $\Delta \leftarrow v.pos - u.pos$; /* distance between u and v */
        $v.disp \leftarrow v.disp + (\Delta/|\Delta|) * f_r(|\Delta|)$; /* displacement */
  function $f_a(x) = x^2/k$; /* compute attractive force */
  foreach $e \in E$ **do**
    $\Delta \leftarrow e.v.pos - e.u.pos$; /* e is ordered vertex pair $.v$ and $.u$ */
    $e.v.disp \leftarrow e.v.disp - (\Delta/|\Delta|) * f_a(|\Delta|)$;
    $e.u.disp \leftarrow e.u.disp + (\Delta/|\Delta|) * f_a(|\Delta|)$;
  foreach $v \in V$ **do**
    /* limit max displacement to frame; use temp. t to scale */
    $v.pos \leftarrow v.pos + (v.disp/|v.disp|) * \min(v.disp, t)$;
    $v.pos.x \leftarrow \min(W/2, \max(-W/2, v.pos.x))$;
    $v.pos.y \leftarrow \min(L/2, \max(-L/2, v.pos.y))$;
  $t \leftarrow cool(t)$; /* reduce temperature for next iteration */

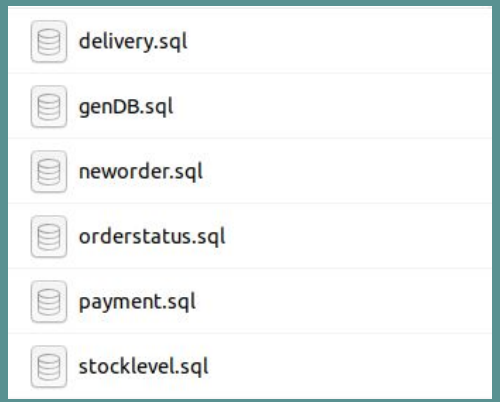
Fonctionnalités réalisées : génération d'un graphe de dépendance

1)

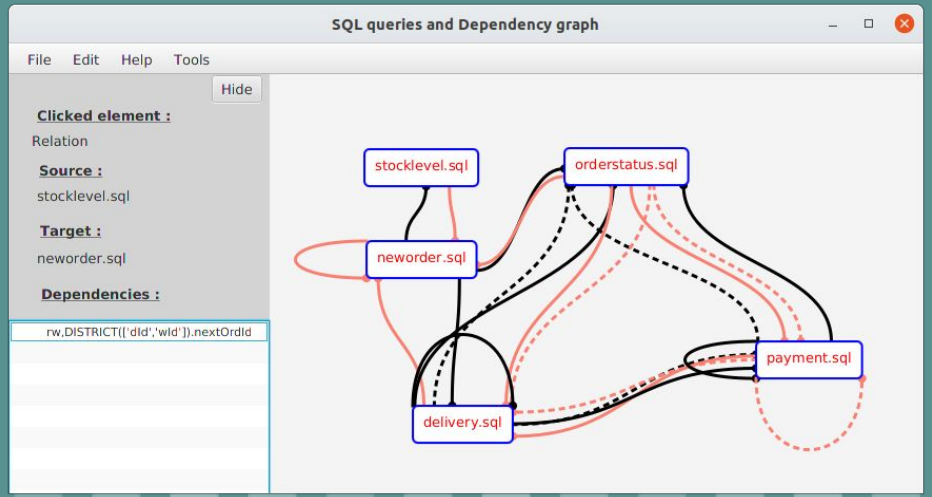


2) Choix du répertoire



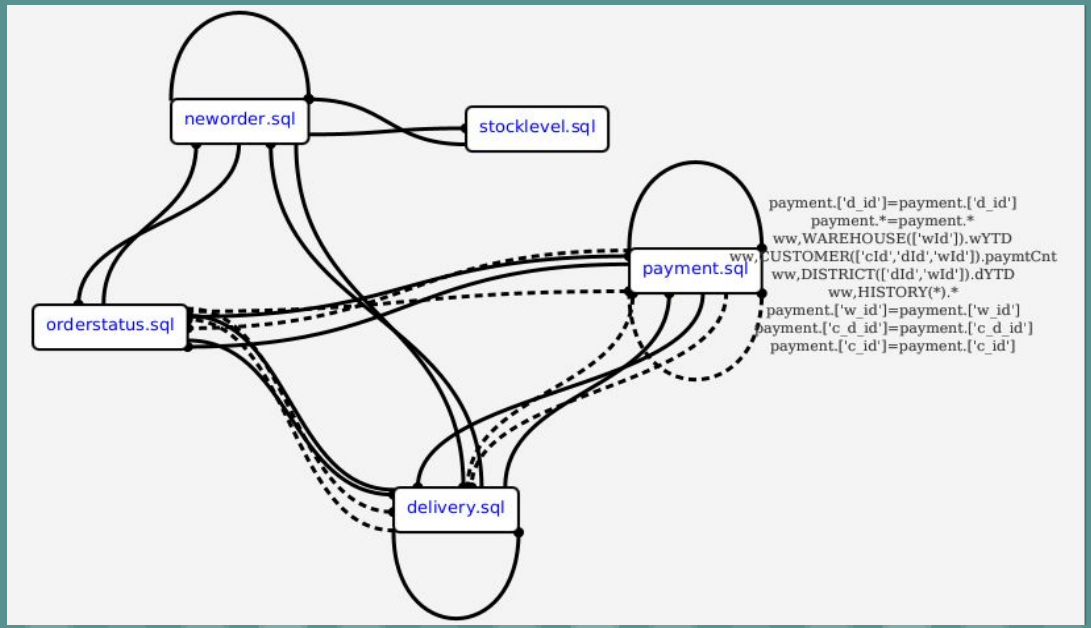


3)



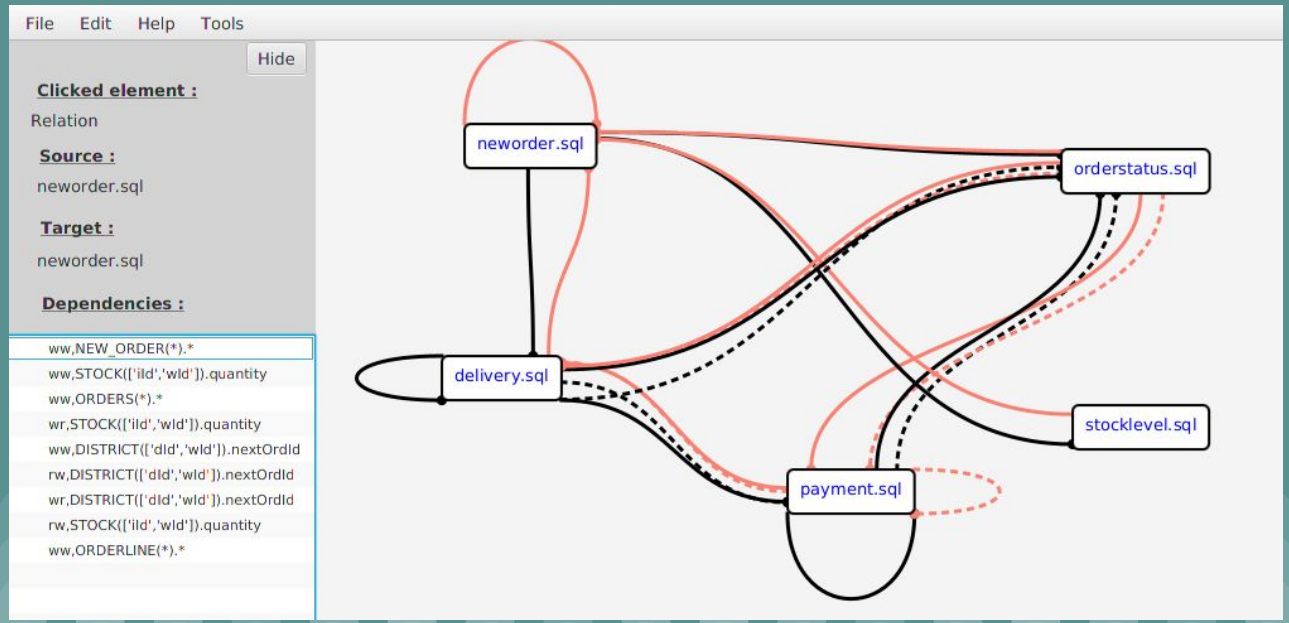
Fonctionnalités réalisées :

1) affichage du texte de relation de dépendance



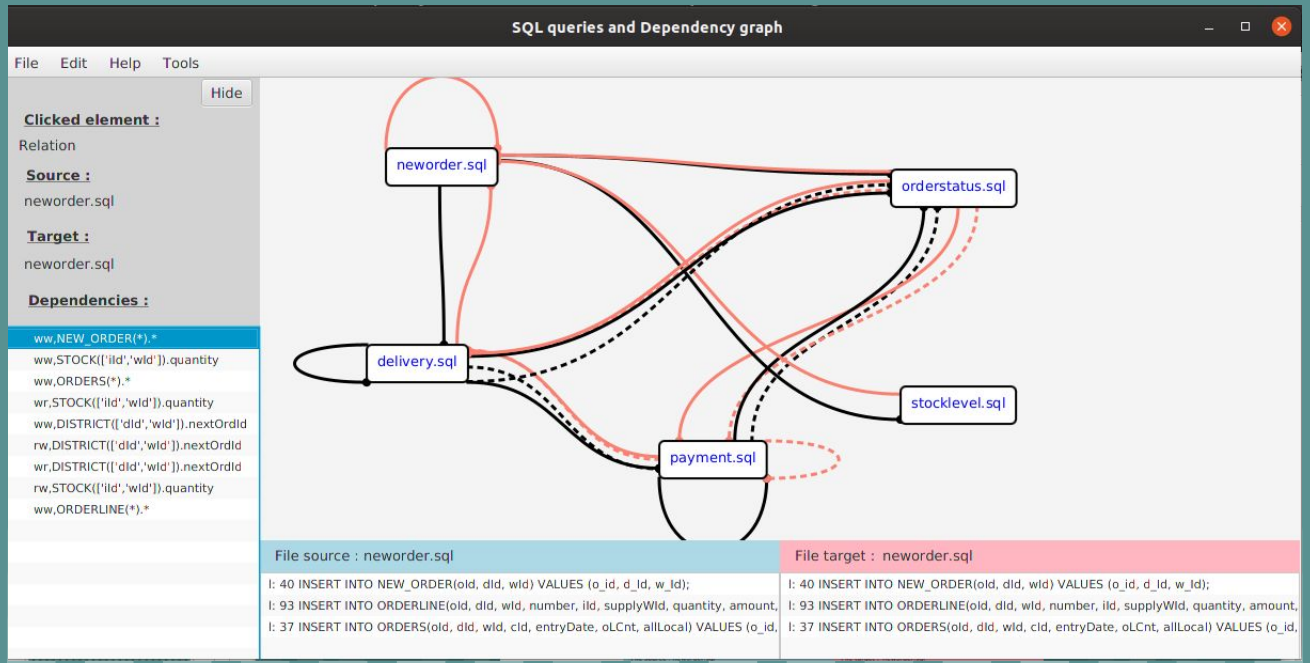
Fonctionnalités réalisées :

1) affichage du texte de relation de dépendance
2) **affichage des dépendances**



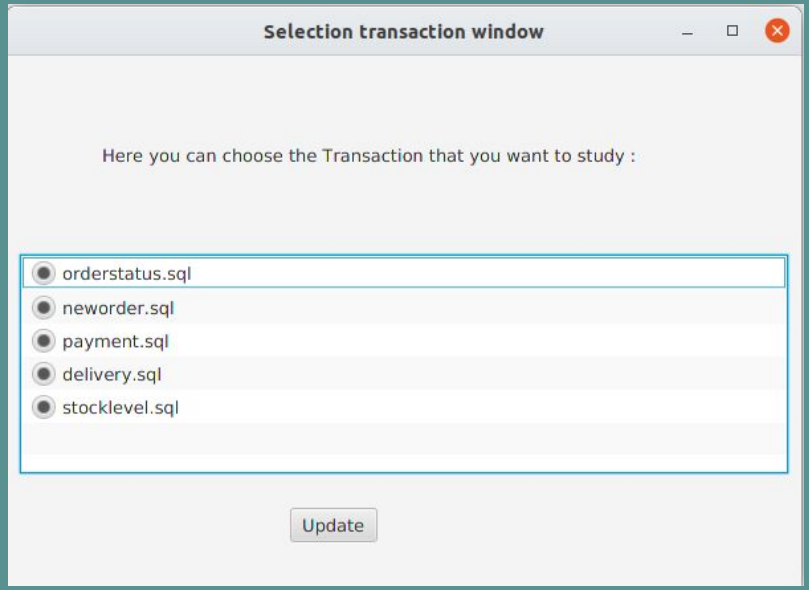
Fonctionnalités réalisées :

1) affichage du texte de relation de dépendance
2) affichage des dépendances
3) **affichage des raisons de dépendances**



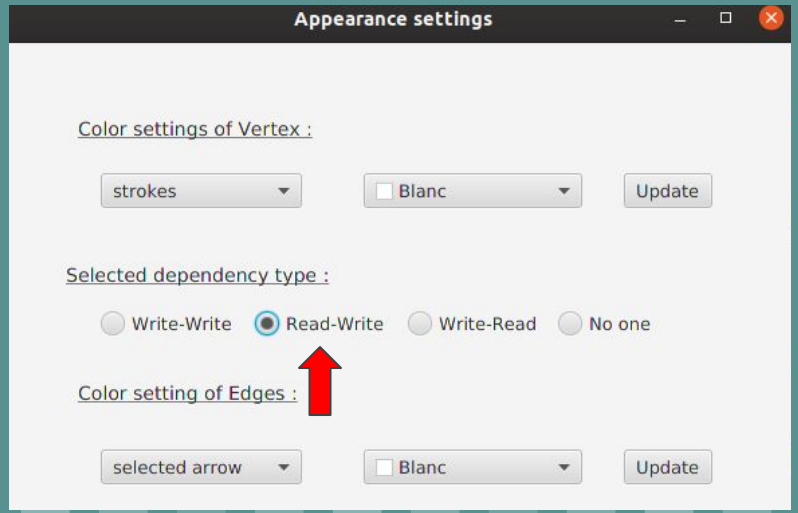
Fonctionnalités réalisées :

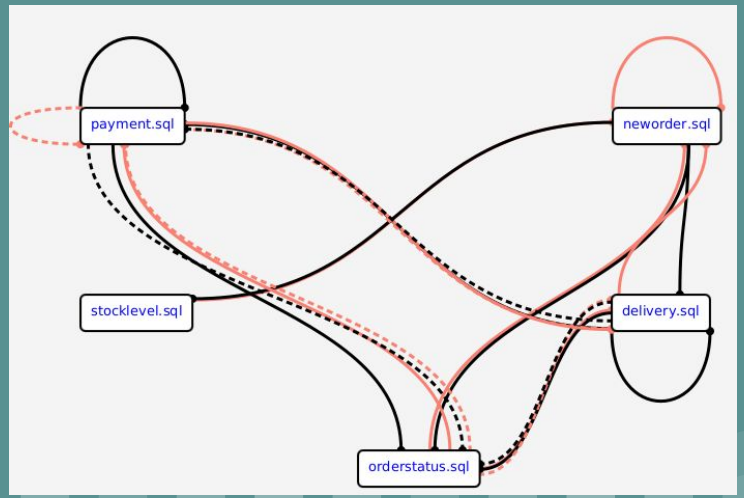
1) affichage du texte de relation de dépendance
2) affichage des dépendances
3) affichage des raisons de dépendances
4) **sélection de certaines Transactions**



Fonctionnalités réalisées :

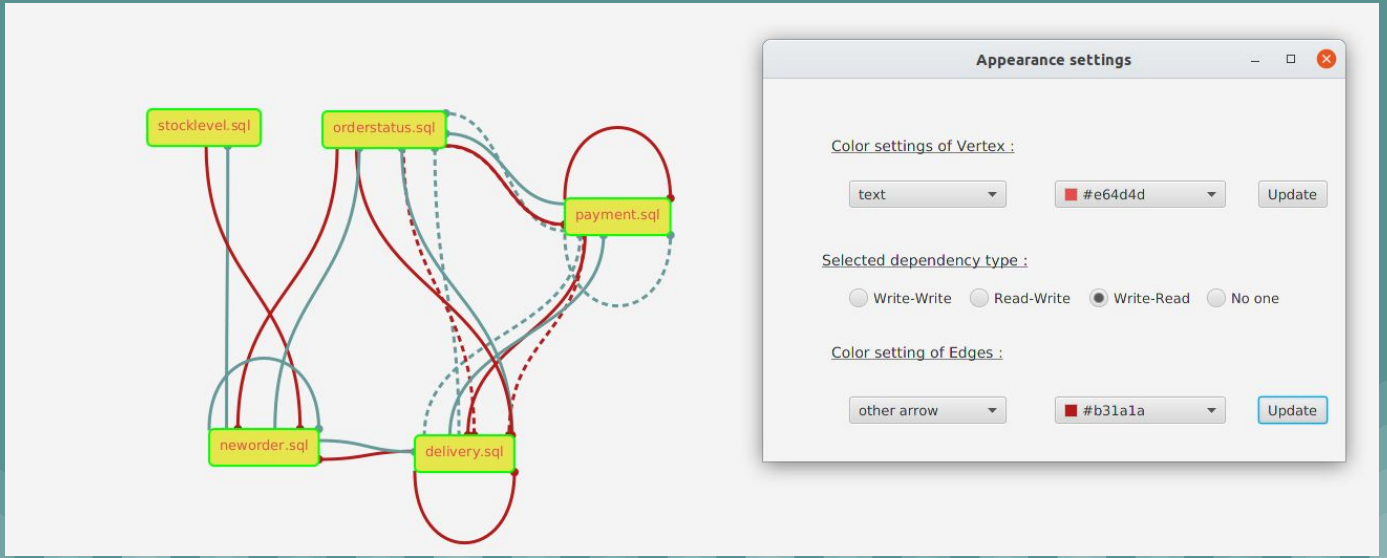
1) affichage du texte de relation de dépendance
2) affichage des dépendances
3) affichage des raisons de dépendances
4) sélection de certaines Transactions
5) **mise en évidence de certains types de dépendance**





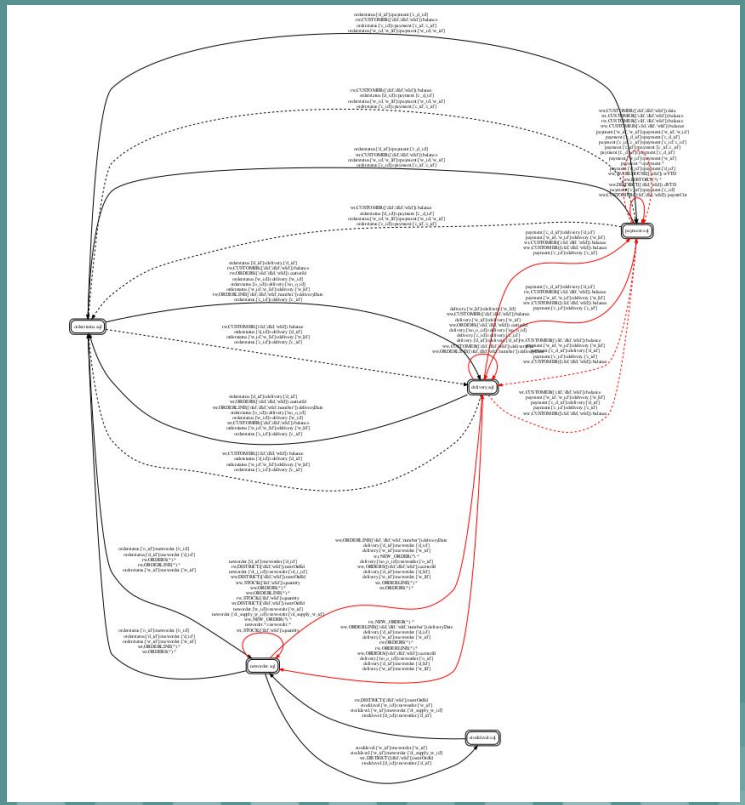
Fonctionnalités réalisées :

1) affichage du texte de relation de dépendance
2) affichage des dépendances
3) affichage des raisons de dépendances
4) sélection de certaines Transactions
5) mise en évidence de certains types de dépendance
6) **gestion du style du graphe**



Fonctionnalités réalisées :

1) affichage du texte de relation de dépendance
2) affichage des dépendances
3) affichage des raisons de dépendances
4) sélection de certaines Transactions
5) mise en évidence de certains types de dépendance
6) gestion du style du graphe
7) **exportation d'un graphe sous divers format**



Fonctionnalités non implémentées

Les jalons ou objectifs non-atteint sont:

- dans l'implémentation d'une Interface Utilisateur
    - génération d'un graphe ayant une légende lors de l'exportation
- la recherche du nombre de cycle critique dans le graphe de dépendance

4. COVID 19

- ➔ Quel impact les mesures contre le COVID19 ont eu sur le déroulement du projet ?

- ➔ Comment le projet aurait-il évolué dans des conditions normales ?

5. Conclusion

- **Ce qu'il faut retenir du Projet Long**
  Impressions personnelles sur l'année de Projet Long et ses enseignements
- **Quelle suite donner au logiciel ?**
  En quoi et comment améliorer et compléter le logiciel ?

Quel serait le but d’une version 2 du logiciel ?

Afin d’envisager une continuité du développement de ce logiciel nous avons remplis une petite liste d’extensions dans le Rapport_Intermediaire.pdf afin de constituer divers jalons pouvant mener à la complétion ou l’amélioration de l’application dans sa globalité .

cf : Rapport intermédiaire.pdf

- Étendre le domaine d’application a d’autre langage
- génération d’un graphe ayant une légende lors de l’exportation
- la recherche du nombre de cycle critique dans le graphe de dépendance

Merci de votre écoute.