README DE MON PROJET: REQUETEUR UNIVERSEL

Principe de fonctionnement: MVC

1- La connexion: Lorsqu'un utilisateur demande à se connecter en entrant spécifiant les informations nécessaire(driver, database, nom d'utilisateur et le mot de passe) on observe 2 cas de figures:

- Tout s'est bien passé, alors il y a envoi de 2 requêtes au serveur, une pour récupérer la liste des tables et des colonnes et l'autre pour récupérer les clés étrangères et leur clé primaire correspondant. et les stocks dans des structures.

Dans un deuxième temps ils chargement de toutes ses tables sont enregistré dans une collection dans le Manager, et un renvoie des noms de tables vers la vue.

- L'identification s'est mal passé il reçoit alors un message d'erreur.

Remarque: Selon le driver choisi par l'utilisateur le Manager spécifiera le nom de la classe à charger et construira la chaîne de connection correspondante.

2- Construction d'une requête: L'utilisateur sélectionne juste des tables et des colonnes dont il a besoin et tout se fait automatique (remplissage de la zone Select, From).
Par défaut les jointures entre les tables sont générées automatique si la case à cochée est sélectionnée.
Il peut désactiver les jointures à tout moment en désélectionnant cette case à cocher.

3- **Operations disponibles:**

- Il peut **executer** sa requête en un clic sur le bouton executer si celle s'est bien passé il obtient le résultat de la requête dans une table, sinon il obtient un message d'erreur.
- Il peut **vider** toutes les zones par un clic sur le bouton Raz.
- Il peut aussi **exporter** le résultat dans un fichier Excel ou CSV.

Difficultés:

- Recuperation des jointures dans la base de données et stockage dans une structure adéquate:
Stockage dans un HashMap<String, Vector<String>> où le premier paramètre représenté le nom de la table référencée par le clé étrangère.

- Aussi nous ne récupérons que les clés primaires des tables référencées par une clé étrangère et ceci se faisant en parallèle pour des raisons d'ordonnancement (là pour des clés composées (étrangères et primaires) on pourra les stocker dans un ordre dans deux collections de string). Ainsi facilitera la recuperation de celles l'index des champs dans les collections coïnciderons normalement.

Traitement:

1- Nous avons une méthode qui prend en paramètre 2 tables et vérifie s'il existe des jointures entre elles.
cette méthode renvoie un tableau de 2 entiers (0 ou 1).
Si la première table reference la 2ème, l'entier d'index 0 reçoit 1 sinon 0.
Si la première table est référencée par la deuxième table l'entier d'index 1 reçoit 1 sinon 0.

Récapitulatif: si notre méthode est existJoin(Table t1, Table t2).
si existJoin(t1, t2) = {1, 0} alors t1 reference t2.
si existJoin(t1, t2) = {0, 1} alors t2 reference t1.
si existJoin(t1, t2) = {0, 0} alors il n'existe pas de jointures entre t1 et t2.

le cas si existJoin(t1, t2) = {1, 1} ne peut presque pas être obtenu en supposant qu'on a un MCD bien conçu (Dans ce cas les 2 tables sont regroupées en une table).

2- Nous avons une deuxième méthode qui fabrique les jointures entre plusieurs tables et les stocke dans une collection et retourne celle-ci.

Sauvegarde de la connexion en utilisant un attribut

L'idée de sauvegarde des connexions utilisé est de les sérialiser dans un fichier et de les desérialiser à chaque lancement de l'application afin de les récupérer et les envoyer à la vue.

Mais j’ai pas le temps de finir cette options additionnelle.

Autres difficultés: la propagation des événements vu que tous sont crées sur ma fenêtre principale afin de pouvoir utiliser un même événements sur plusieurs objets.