Intégration des données – TP1 – Partie 1

M1 – Informatique - Luminy

Gautier Maurice – 27 mars 2020

Structure des sources remisent :

- Le dossier datasources contient des sources de données au format csv.
- Le dossier mediator_config contient des fichiers de configurations nécessaires pour la construction des wrappers
- Le dossier boost_1_72_0 contient les headers boost nécessaires pour la librairie msql-connector-cpp
- Le dossier src contient les sources du TP
- Le dossier include contient les headers du TP
- Le fichier CMakeList.txt est nécessaire pour créer un makefile compatible avec votre environnement.
- Le script sql_init_script.sql créé une base de données 'maurice_gautier_inte_donnees_tp1' et créé un utilisateur local 'tp1-user-gm' avec tout les droits sur cette base de données.
- Le script sql_view_creation.sql créé une vue intéressante pour répondre à l'énoncé du TP.
- Le script sql_close_script.sql détruit la base de données et l'utilisateur.
- Le script local-build-mysqlconnector++.sh clone le dépôt git officiel de my-sql-connector++ dans le dossier mysql-connector-cpp et installe une version compatible avec votre environnement dans le dossier mysql-conn-cpp.

Prérequis :

Le projet a été écrit en c++, pour compiler et installer les librairies requises dans le répertoire du projet, il faut avoir CMake, Git d'installer et disposer des librairies standard c++ ainsi qu'un d'un compilateur c++ (comme g++). CMake et Git doivent être accessible via les commande cmake et git depuis un terminal.

Un environnement MySql opérationnel sur lequel vous pouvez créer des bases de données et des utilisateurs

Librairies utilisées :

-STL C++ https://en.cppreference.com/w/
-MySql 5.7 ou plus
-MySql Connector/C++ https://dev.mysql.com/doc/dev/connector-cpp/8.0/

Comment utiliser les sources :

- Créer les exécutable :
  Lancer le script $ ./local-build-mysqlconnector++.sh.
  Lancer la commande $ cmake .
  Puis $ make
- Initialiser la base de données :
  Lancer le script sql_init_script.sql dans un environnement mysql

$ mysql -u sql-login < sql_init_script.sql

- Charger la base de données grâce aux wrappers :
  $ ./bin/app
- Créer la vue :
  Lancer le script view_creation.sql dans le même environnement mysql
  $ mysql -u sql-login < view_creation.sql

A ce stade, l'utilisateur du TP peut utiliser la vue TouristeCinephile pour répondre à ses besoins.

Le fonctionnement de ./bin/app

Dans le main.cpp, l'application propose une fonction :
void load_wrapper(std::string config_path, std::shared_ptr<sql::Connection> connection)

C'est la fonction principale, elle va :

- Utiliser la wrapper factory pour instancier le bon wrapper à partir du fichier de configuration
- Donner la connexion sql au wrapper
- Demander au wrapper de remplir la base de données.

Le format des fichiers de configurations est, pour l'instant, très simple étant donné que notre application ne supporte que les wrapper CSV → SQL

Futures améliorations :

-Faire en sorte que l'utilisateur de ./bin/app n'ait pas à toucher le code :

Par exemple créer un fichier config.list qui liste les chemins vers les fichiers de configurations de wrapper qu'on souhaite (re)charger. L'application va juste lire ce fichier, et pour chaque ligne instancier le wrapper configuré à ce chemin et le charger.

-Implanter des wrappers XML → SQL, JSON → SQL

-Proposer des factories adéquate.

-Écrire une documentation du format des fichiers de configurations pour les futurs utilisateurs de l'application.