Projet LP25 A2021

SQL

Membres du groupe :
ALLAIRE Céliane
MACULLO Quentin
SHNEIDERLIN Pierre
FERLIN Jules

utbm
université de technologie
Belfort-Montbéliard

Table des matières

Sujet 3
Sujet 3
Organisation 3
Les différentes parties du projet et leurs tâches complexes 3
Parse ou interprétation 3
Check ou vérification de cohérence 4
Expand ou enrichissement 4
Exécute ou exécution 4
Les autres fichiers 5
Conclusion et Améliorations 6

Sujet

Le but principal est de gérer une base de données à l'aide du langage C. C'est-à-dire que l'utilisateur entre des fonctions SQL et le programme les interprètes, vérifie leurs cohérences, ajoute les données si nécessaire et enfin les exécute. Le programme peut gérer plusieurs tables dans une base de données.

Organisation

Pour l'organisation du projet nous avons opté pour une partie chacun. En effet, le sujet étant divisé en quatre parties cela rendait la division du travail plus simple. Bien sûr cette division du travail ne nous a pas empêché de participer sur les autres parties et nous entraider notamment pour la partie d'exécution.

Les différentes parties du projet et leurs tâches complexes

Pour répondre au sujet, il est nécessaire d'implémenter 4 grandes parties. Chacune de ces parties gère une tâche précise :

Parse ou interprétation

Descriptif

Cette partie correspond à la première partie, elle lit ce que l'utilisateur a entré au clavier, et la traduit en remplissant les structures de données. Cette partie ne s'occupe pas de savoir si la requête entrée par l'utilisateur est cohérente mais seulement si la syntaxe est judicieuse.

Tâches complexes

Les fonctions les plus complexes dans cette partie ont été les fonctions parse_create_fields_list et parse_fields_or_values_list. Ces fonctions ont demandé une grande réflexion car il fallait envisager tous les cas de figures de syntaxes différentes. Il a donc fallu faire appel à différentes fonctions et utiliser différentes combinaisons de "if...then" pour essayer de parcourir tous les cas différents. Pour mieux visualiser ce les différentes conditions qu'il fallait envisager nous avons passé du temps à rédiger différentes requêtes que pouvait entrer l'utilisateur qui ne respectaient pas la même syntaxe (par exemple : en plaçant un espace entre le dernier caractère de la dernière valeur et une parenthèse fermante, ou non.).

Check ou vérification de cohérence

Descriptif

Cette partie est la deuxième partie. Elle reprend les structures de données créées et remplies par l'étape précédente et vérifie leur cohérence quant à la base de données ouverte et aux tables déjà dans cette base ou non. Cette étape est très importante car c'est elle qui détermine si la requête peut être exécuté ou non. Les valeurs entrées, notamment pour la requête « insert », sont aussi traduites et vérifiées.

Tâches complexes

Dans cette partie, les deux tâches les plus complexes ont été check_query_insert et check_query_update. Non seulement elles demandaient l'implémentation d'autres fonctions mais en plus il fallait être minutieux pour ne pas se perdre dans ses grandes fonctions. Au final cette partie n'a pas été la plus compliquée a implémenter une fois avoir compris les attendus.

Expand ou enrichissement

Descriptif

Cette partie n'est valable que pour deux requetés : « select » et « insert ». En effet, elle s'occupe de remplir le nom des colonnes lorsque l'utilisateur a entré « * » pour signifier tous les champs.

Tâches complexes

L'aspect le plus complexe de cette partie fut la lecture des différents noms de champs et leurs types dans la base de données. En effet, il fallait avoir une bonne visualisation de la manière dont étaient stockés les différentes données.

Exécute ou exécution

Descriptif

Cette partie est la dernière. Le but de cette partie est l'écriture et la lecture des différents fichiers gérant les bases de données. En fonction des requêtes et des différentes données on fait appel à une fonction adaptée qui va faire appel aux différentes fonctions en fonctions des actions à effectuer. Grâce au travail effectué précédemment on fait appel aux données stockés dans les différentes structures pour avoir les informations sur la requête mais aussi sur les différentes colonnes et types de données stockés dans la base de données étudié.

Tâches complexes

Par manque d'organisation, les fonctions implémentées dans cette parties sont rares, seul les fonctions de création/suppression de table et suppression de base de données on était implémenté. Nous comptons cependant les finir avant le deuxième rendu du code pour avoir un projet bien complet.

Les autres fichiers

Table.c/h

Ce fichier s'occupe de toutes les fonctions agissant sur les tables. Elle permet notamment l'ouverture des différents fichiers comme le fichier index et le fichier de définition. Ce fichier contient aussi une fonction pour vérifier l'existence d'une table, une fonction pour créer une table, une fonction pour supprimer une table, une fonction pour lire la définition d'une table, et autres. Grace aux fonctions dans ce fichier la manipulation des tables et des différents fichiers qui lui sont associés est rendu plus simple. Le .h contient une déclaration de structure pour la manipulation du fichier index.

Database.c/h

Ce fichier contient deux fonctions qui s'occupent de la gestion des bases de données. C'est-à-dire la création d'un répertoire et la suppression d'un répertoire (ainsi que tout son contenu, c'est-à-dire ses différents fichiers). Ces fonctions sont essentielles lors de la manipulation de nos bases de données.

Utils.c/h

Ce fichier contiens trois fonctions différentes. La première d'entre elle « make_full_path » fait en sorte que le chemin entré par l'utilisateur est sous la bonne forme. C'est-à-dire que cette fonction adapte l'entrée de l'utilisateur pour respecter la syntaxe demandé par linux. La fonction suivante est « directory_exists » qui vérifie l'existence du chemin jusqu'au répertoire au lieu indiqué. Ensuite nous avons la fonction « _mkdir » qui crée un répertoire demandé. Toutes ces fonctions permettent la gestion du chemin entré par l'utilisateur. Le fichier .h quant à lui possède différentes structures essentiels pour le projet.

Record_list.c/h

Ce dernier fichier contient des fonctions utiles pour gérer les listes dans le projet. On y retrouve notamment un fonction « clear_list » pour vider une liste et la fonction « add_record » pour ajouter un élément à la liste. On trouve aussi dedans une fonction pour l'affichage du résultat d'une requête et une fonction pour calculer la longueur d'un enregistrement. Le fichier .h possède aussi deux structures en rapport avec les listes.

Conclusion et Améliorations

Le premier rendu du code est assez incomplet malheureusement. Une mauvaise gestion de notre temps et des difficultés à comprendre certaines parties du sujet nous ont beaucoup ralentis. Nous espérons cependant compléter au mieux notre code avant le deuxième rendu en janvier. Grâce au temps supplémentaire notre code devrais avoir un meilleur fonctionnement et une meilleure gestion des différentes erreurs possibles.

Ce projet fut assez différent des autres projets que nous avons codés jusqu'à maintenant. Ce fut intéressant de tenter de programmer des fonctions déjà existantes, notamment avec des types de structure déjà crée. Nous avons trouvé ça d'ailleurs assez compliqué car au début du projet nous ne connaissions pas les différentes structures et leurs utilités. Cependant, apprendre à gérer un projet déjà structuré et codé en partie fut très enrichissant et sera probablement utile dans le futur. Un autre point positif du projet fut l'apprentissage du langage SQL. Certains membres de notre groupe avaient déjà rencontré ce langage lors des cours d'IF3A ce fut donc l'occasion d'améliorer nos connaissances, tandis que pour d'autres ce fut l'occasion parfaite pour s'initier au langage SQL et son fonctionnement.

Ainsi, malgré certaines difficultés à s'adapter à un sujet d'une ampleur assez grande mais aussi déjà complété en partie nous avons fait de notre mieux pour respecter les consignes et la syntaxe attendue dans notre code. Le premier rendu de notre code contient de nombreuses fonctions essentielles mais malgré cela toutes les fonctions n'étant pas complète le code ne peut seulement effectuer une partie des choses qu'il devrait pouvoir effectuer.