**Projet C : Base de Données - Git Bash**

**Introduction**

Le projet consistait à créer une base de données en C avec une interface REPL (Read-Eval-Print Loop) fonctionnelle, intégrant un arbre binaire pour la gestion des données. Plusieurs défis techniques ont été rencontrés au cours du développement, allant de la structure de l’arbre binaire à l’implémentation des commandes de REPL et à leur intégration dans Git Bash.

**Liste des Problèmes Rencontrés**

**1. Problèmes d’implémentation de l’arbre binaire**

* **Problème** : Difficulté initiale à implémenter les fonctionnalités de base (insertion, recherche, suppression) pour un arbre binaire.
* **Solution** :
  + Implémentation d’un nœud contenant les informations d’une ligne de la base (ID et nom).
  + Création des fonctions : insert, search, delete, et inorder.
  + Ajout d’un mécanisme pour gérer les doublons d’ID (ignorer l’insertion si l’ID existe déjà).

**2. Test des fonctions de l’arbre binaire**

* **Problème** : Les tests des fonctions insert et delete n’étaient pas intégrés dans un environnement interactif.
* **Solution** : Création de la fonction btree\_demo dans btree.c, permettant de tester les principales fonctionnalités avec des exemples codés en dur.

**3. Problème d’inclusion circulaire**

* **Problème** : Inclusion directe de btree.c dans plusieurs fichiers, entraînant des erreurs de redéfinition lors de la compilation.
* **Solution** : Création d’un fichier d’en-tête btree.h pour déclarer les fonctions et structures de l’arbre binaire. L’inclusion directe de btree.c a été remplacée par #include "../include/btree.h".

**4. Problèmes dans la boucle REPL**

* **Problème** :
  + Erreurs lors de la reconnaissance des commandes.
  + Commandes non fonctionnelles pour insérer, sélectionner ou supprimer des données.
* **Solution** :
  + Ajout d’une structure Statement pour représenter les commandes (insert, select, delete).
  + Création des fonctions prepare\_statement pour interpréter les entrées utilisateur et execute\_statement pour exécuter les commandes.
  + Résolution des erreurs de segmentation causées par des appels incorrects à des fonctions non définies ou mal déclarées.

**5. Problème de gestion des doublons**

* **Problème** : Les commandes insert ne géraient pas correctement les doublons d’ID, provoquant des erreurs.
* **Solution** : Modification de la fonction insert pour vérifier si l’ID existait déjà dans l’arbre avant d’ajouter une nouvelle ligne. Un message est affiché si l’insertion est ignorée.

**6. Suppression d’un ID dans l’arbre binaire**

* **Problème** : La commande delete n’était pas correctement intégrée à la boucle REPL.
* **Solution** :
  + Ajout de la reconnaissance de la commande delete dans prepare\_statement.
  + Ajout de l’exécution de la commande dans execute\_statement, avec la suppression du nœud correspondant à l’ID donné.
  + Gestion des cas où l’ID n’existe pas.

**7. Sélection d’un ID spécifique**

* **Problème** : La commande select ne permettait que d’afficher tout l’arbre binaire.
* **Solution** :
  + Ajout d’une vérification dans execute\_select pour permettre la sélection d’un ID spécifique : select <id>.
  + Recherche de l’ID dans l’arbre avec search.
  + Affichage des données correspondantes ou d’un message si l’ID n’existe pas.

**8. Problèmes de compilation et d’exécution**

* **Problème** :
  + Erreurs de compilation fréquentes, telles que "too many arguments" ou "undefined function".
  + L’exécutable se fermait immédiatement après l’exécution de btree\_demo.
* **Solution** :
  + Vérification des signatures des fonctions pour assurer la correspondance avec leur utilisation.
  + Modification de main.c pour basculer entre les modes démonstration et interactif, en fonction des arguments de ligne de commande.
  + Ajout d’une boucle d’attente (getchar()) après btree\_demo pour garder la console ouverte.

**Conclusion**

Ce projet a permis de mettre en pratique plusieurs concepts avancés en C, notamment la gestion des structures de données, l’interactivité avec l’utilisateur et la détection des erreurs. Malgré les défis rencontrés, les solutions implémentées ont abouti à une base de données fonctionnelle et interactive via Git Bash.

DUPONT Lélian 3SI3