PROJET C - SAMY FARSSI

Ce projet consiste en une application de base de données simple développée en C, qui utilise une structure de table pour stocker des données en mémoire et un arbre binaire pour faciliter les opérations telles que l'insertion, la recherche, l'affichage et la suppression de nœuds. Le projet inclut également une persistance sur disque pour l'arbre binaire et un mode interactif (REPL) pour interagir avec les données.

---

2. Architecture des Fichiers

main.c

- **Responsabilité : Point d'entrée du programme.**
- **Fonctions principales :**
    - **run_tests : Lance une série de tests unitaires.**
    - **repl_loop : Active le mode interactif pour manipuler la base de données.**

database.h et database.c

- **Responsabilité : Gestion de la structure de table.**
- **Fonctions principales :**
    - **insert_row : Insère une nouvelle ligne dans la table.**
    - **find_row_by_id : Recherche une ligne par ID.**
    - **delete_row : Supprime une ligne de la table.**
    - **display_table : Affiche le contenu de la table.**

btree.h et btree.c

- **Responsabilité : Implémentation de l'arbre binaire.**
- **Fonctions principales :**
    - **insert_node : Insère un nœud dans l'arbre.**
    - **display_tree : Affiche l'arbre en ordre croissant.**
    - **free_tree : Libère la mémoire allouée pour l'arbre.**
    - **save_tree et load_tree : Gèrent la persistance sur disque.**

repl.h et repl.c

- **Responsabilité : Interface en ligne de commande (REPL).**
- **Fonctions principales :**
    - **Interprète les commandes utilisateur (insert, select, delete, save, load, exit).**

Makefile

- **Responsabilité : Automatisation de la compilation.**
- **Cibles :**
    - **all : Compile l'exécutable.**
    - **clean : Supprime les fichiers temporaires.**

---

3. Drapeaux du Compilateur

Choix des Drapeaux

- **-g : Ajoute des informations de débogage pour les outils comme Valgrind.**
- **-Wall : Affiche tous les avertissements pour identifier des erreurs potentielles.**
- **-Wextra : Ajoute des avertissements supplémentaires.**
- **-pedantic : Assure une conformité stricte à la norme C.**

Version du Compilateur

- **Utilisé : gcc 10.3.0**

---

4. Tests et Outils Dynamiques

Outils Utilisés

- **Valgrind :**
    - **Détecte les fuites de mémoire.**
    - **Vérifie les erreurs d'accès mémoire.**
- **Sanitizers (si disponibles) :**
    - **AddressSanitizer pour détecter les débordements de tampon.**

Résultats

- **Problème de fuite mémoire identifié et corrigé en ajoutant la libération explicite des nœuds dans l'arbre avec free_tree.**

---

5. Commandes de l'Interface

Commandes Disponibles

- **insert <ID> <Name> : Ajoute un nœud dans l'arbre.**
- **select : Affiche tous les nœuds.**
- **delete <ID> : Supprime un nœud spécifique (non implémenté).**
- **save : Sauvegarde l'arbre dans un fichier.**
- **load : Charge un arbre à partir d'un fichier.**
- **exit : Quitte le programme.**

---

6. Instructions pour Compiler et Exécuter

1. **Compilation :**

bash

make

2. **Exécution :**

bash

./main

3. **Nettoyage :**

bash

make clean

---

7. Contributions Futures

- **Implémentation de la suppression complète dans l'arbre binaire.**
- **Ajout de nouvelles commandes SQL simulées (par exemple, update).**

- **Intégration de tests fuzzers pour renforcer la robustesse.**