Fundamentos de Bases de Datos

Práctica 3

Grosso Christian, Zhan Jiabo

**1. Introducción.**

En esta práctica aprenderemos a gestionar tablas e índices en una base de datos simplificada, almacenando tanto los datos como los índices en archivos binarios. Las operaciones de inserción, búsqueda y visualización se realizarán directamente sobre el disco, permitiendo compartir los datos entre múltiples instancias del programa.

Implementaremos un sistema que almacena registros de longitud variable, con una cabecera para manejar registros borrados, y utilizaremos un árbol binario de búsqueda como índice. Este enfoque práctico refuerza conceptos de estructuras de datos y manejo de archivos, integrándolos en un sistema funcional de base de datos simplificada

**2. Objetivos**

El objetivo de esta práctica es crear un sistema básico para almacenar, buscar y organizar datos de manera eficiente usando archivos binarios. Esto incluye implementar tablas para registros de longitud variable y un índice con un árbol binario de búsqueda para un acceso rápido. Además, busca desarrollar habilidades en el manejo de archivos en disco, estructuras dinámicas y la integración de diferentes partes del sistema, ayudando a entender mejor cómo funcionan las bases de datos.Los datos son almacenados en un archivo .dat y los índices en un archivo de extensión .idx. Tambien se implementa una interfaz con opciones:

USE: para crear o usar tablas en la base de datos

INSERT: para introducir los datos en las tabla

PRINT: para devolver por pantalla el árbol obtenido a partir de la base de datos

EXIT: para salir de la interfaz.

**3. Algoritmos a implementar**

1. **createTable**: Crea un archivo para almacenar los datos, inicializando su cabecera. Si ya existe, simplemente lo abre.

2. **createIndex**: Crea un archivo para el índice asociado a la tabla, inicializando su cabecera. Si ya existe, lo abre.

3. **findKey**: Busca una clave en el índice. Si la encuentra, devuelve su posición en el archivo de datos; si no, señala dónde debería insertarse.

4. **addTableEntry**: Añade un nuevo registro en el archivo de datos y actualiza el índice. Reutiliza registros borrados si es posible.

5. **addIndexEntry**: Inserta un nuevo nodo en el árbol binario de búsqueda del índice, organizándolo correctamente.

6. **printTree**: Muestra el árbol binario del índice en pantalla, indicando claves, nodos y su relación jerárquica.

7. **get\_max\_nodeId**: Este algoritmo actualiza el valor de la variable max\_nodeId si el nodo actual tiene un identificador mayor al almacenado. Sirve para mantener un seguimiento del identificador más alto en el árbol de índices.

8. **check\_dat**: Verifica si un archivo tiene la extensión .dat. Devuelve true si cumple estas condiciones, o false en caso contrario.

9. **replaceExtensionByIdx**: Cambia la extensión de un archivo de .dat a .idx.

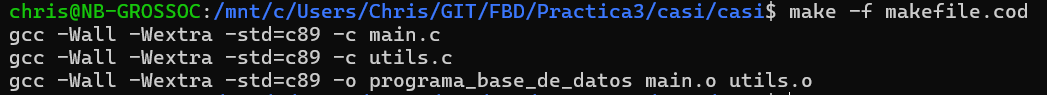
**4. Ficheros de la practica**

En la practica hay los checks de las funciones de 1 a 6 implementadas en la práctica a excepción del check de addTableEntry. Tambien hay un makefile.cod para la compilación del menu.

**5. Compilaciones de Tester y Menu**

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente



**6. Ejecuciones del Tester**

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

Aqui hay todos los tests que pasan con ok y el arbol imprimido correctamente.

También ejecutando el tester con el comando **valgrind ./tester** no se muestran fugas de memoria.

Immagine che contiene testo, schermata, software, Software multimediale

Descrizione generata automaticamente

**7. Ejecuciones del Menu**

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

En el menú mostrado, las funciones ejecutadas correctamente incluyen:

1. la selección de una tabla con la opción Use, que permite al sistema preparar un archivo .dat para operaciones;
2. la inserción de registros con la opción Insert, en la que se agrega información como el ID y el título de un libro a la tabla seleccionada;
3. la opción Print, que imprime el árbol binario asociado a la tabla, mostrando las claves primarias y sus posiciones jerárquicas de manera estructurada.
4. finalmente, la opción Exit cierra el programa de manera segura.

**8. Conclusion**

En esta práctica se han trabajado conceptos básicos pero esenciales sobre cómo funcionan las bases de datos por dentro. Se aprendió a manejar tablas e índices almacenados en archivos binarios, optimizando el acceso a los datos mediante un árbol binario de búsqueda. También se abordó cómo reutilizar espacio con registros borrados y cómo mantener la sincronización de datos entre diferentes ejecuciones del programa. En resumen, esta práctica ayuda a entender mejor el funcionamiento interno de una base de datos sencilla, sirviendo como base para estudiar sistemas más complejos en el futuro.