Санкт-Петербургский Национальный Исследовательский Университет ИТМО

Факультет программной ­­­инженерии и компьтерной техники

Лабораторная работа №1 по LLP

Вариант 5

Юнусов Роман Ильдарович  
Группа P33102

Преподаватель Юрий Кореньков

Г.Санкт-Петербург

2024 г.

Задание

Реализовать модуль хранящий в одном файле множество таблиц, произвольного размера.  
Внутри ячейки таблицы может хранится целочисленный тип, числа с плавающей точкой, строка произвольного размера.

Необходимо уметь создать, удалять таблицы.

Также нужно уметь делать операции insert - O(1), select - O(N), delete - O(N), update - O(N), Join(N+M) по отношению к записям в таблицах

Детали реализации

Поделим нам файл на listы фиксированного размера.

У каждого листа есть в конце элемент со ссылкой на следующий лист, и ссылкой на предыдущий лист.

* Первый лист файла хранит служебную информацию о том, где находят концы цепочек из следующих листов, а также хранит размер базы данных в листах.
* Второй лист файла и следующие по цепочке хранят данные таблицы – номер таблицы, количество значений в одной записи, количество записей, последний и первый list принадлежащий странице.
* Третий лист файла и другие в цепи хранят информацию о свободных листах.
* Четвертый лист файла и другие в цепи хранят строковую информацию

Если лист файла не является частью цепочек, то он хранит записи, принадлежащие, какой-то из таблиц.

Запись представляет собой последовательность ячеек cell, одна ячейка хранит либо число, либо строку длиной до 32 символов.

struct cell {

union {

int int\_data;

double double\_data;

struct string\_ref string\_link;

};

enum cell\_flag flag;

};

Соответственно комбинация ячеек, которая начинается со специальной(которая имеет flag RAW\_NUM) – это запись в таблице.

Для реализации операций используют функции addRaw, select, delete, update, join

Начнём с addRaw за константу.

Логика проста, для каждой таблицы поддерживаем последнюю страницу, которая ей принадлежит, и всегда стараемся добавлять в неё, если не хватает места, то просто запросим новую пустую страницу(она может быть либо освобожденной, либо просто создадим новую увеличив файл).

Для условий на операции заведем структуру

struct queryCondition {

int stolbec\_num;

enum sign\_cond conds; // " >", " <", "!=", "==", ">=", "<="

enum cell\_flag type;

union {

int int\_data;

double double\_data;

char\* string\_data;

};

};

Массив таких структур задаёт условия, по которым можно понять подходит запись условиям.

Сами select и delete выполнены просто.  
Идём по содержимому таблицы начиная с её последней страницы, и записываем все raw в специальную структуру

struct raw\_sequence {

int page\_num, cell\_num;

int len;

struct cell\* raw\_data;

struct raw\_sequence\* next;

};

После записи всех raw из таблицы, мы запускаем чекер условия, который отсеивает ненужные записи.

В случае с delete, мы ещё отдельно удаляем каждую запись, поскольку у нас есть номер list и номер cellа, с которого начинается запись, а также длина самой записи.  
  
Операция JOinа представляет собой два селекта с условиями, с последующим слиянием получившихся данных

Графики.

График времени добавления

График времени удаления

График select