МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине

“Низкоуровневое программирование”

Вариант № 2

Реляционные таблицы

Студент:

Чухно Матвей
Романович

Группа P33312

Преподаватель:

Кореньков Юрий Дмитриевич

Санкт-Петербург, 2023

Задание:

Создать модуль, реализующий хранение в одном файле данных (выборку, размещение и гранулярное обновление) информации общим объёмом от 10GB соответствующего варианту вида.

- Спроектировать структуры данных для представления информации в оперативной памяти
    - Для порции данных, состоящий из элементов определённого рода (см форму данных), поддержать тривиальные значения по меньшей мере следующих типов:четырёхбайтовые целые числа и числа с плавающей точкой, текстовые строки произвольной длины, булевские значения
    - Для информации о запросе
- Спроектировать представление данных с учетом схемы для файла данных и реализовать базовые операции для работы с ним:
    - Операции над схемой данных (создание и удаление элементов схемы)
    - Базовые операции над элементами данных в соответствии с текущим состоянием схемы (над узлами или записями заданного вида)
        - Вставка элемента данных
        - Перечисление элементов данных
        - Обновление элемента данных
        - Удаление элемента данных

- Используя в сигнатурах только структуры данных из п.1, реализовать публичный интерфейс со следующими операциями над файлом данных:
    - Добавление, удаление и получение информации о элементах схемы данных, размещаемых в файле данных, на уровне, соответствующем виду узлов или записей
    - Добавление нового элемента данных определённого вида
    - Выборка набора элементов данных с учётом заданных условий и отношений со смежными элементами данных (по свойствам/полями/атрибутам и логическим связям соответственно)
    - Обновление элементов данных, соответствующих заданным условиям
    - Удаление элементов данных, соответствующих заданным условиям
- Реализовать тестовую программу для демонстрации работоспособности решения
    - Параметры для всех операций задаются посредством формирования соответствующих структур данных
    - Показать, что при выполнении операций, результат выполнения которых не отражает отношения между элементами данных, потребление оперативной памяти стремится к O(1) независимо от общего объёма фактического затрагиваемых данных
    - Показать, что операция вставки выполняется за O(1) независимо от размера данных, представленных в файле

    - Показать, что операция выборки без учёта отношений (но с опциональными условиями) выполняется за O(n), где n – количество представленных элементов данных выбираемого вида
    - Показать, что операции обновления и удаления элемента данных выполняются не более чем за O(n*m) → t O(n+m), где n – количество представленных элементов данных обрабатываемого вида, m – количество фактически затронутых элементов данных
    - Показать, что размер файла данных всегда пропорционален количеству фактически размещённых элементов данных
    - Показать работоспособность решения под управлением ОС семейств Windows и *NIX
- Результаты тестирования по п.4 представить в составе отчёта, при этом:
    - В части 3 привести описание структур данных, разработанных в соответствии с п.1
    - В части 4 описать решение, реализованное в соответствии с пп.2-3
    - В часть 5 включить графики на основе тестов, демонстрирующие амортизированные показатели ресурсоёмкости по п. 4

Описание:

- Заголовок страницы

```
struct page_header {
    uint16_t free_bytes;
    uint32_t page_number;
    uint32_t page_free_space_seek;
    char table_name[TABLE_NAME_LENGTH];
    uint32_t table_number_in_meta_page;
    uint32_t next_page_number;
};
```

- Заголовок базы данных

```
struct database_header {
    char database_name[DB_NAME_LENGTH];
    database *db;
    uint32_t table_count;
    uint32_t page_count;
    uint32_t last_page_number;
};
```

- База данных

```
struct database {
    database_header *database_header;
    FILE *storage;
```

```
};
```

- Заголовок таблицы

```
struct table_header {
    char name[MAX_TABLE_HEADER_NAME_LENGTH];
    table *table;
    database *database;
    table_schema table_schema;

    uint32_t number_in_meta_page;
    uint32_t page_count;
    uint32_t first_page_number;
    uint32_t last_page_number;
    bool valid;
};
```

- Заголовок схемы таблицы

```
struct table_schema {
    uint32_t column_count;
    row_length row_length;
    column *columns;
    column *last;
};
```

- Таблица

```
struct table {
    table_header *table_header;
    table_schema *table_schema;
};
```

- Колонка

```
struct column {
    char column_name[MAX_TABLE_HEADER_NAME_LENGTH];
    column_type column_type;
    uint16_t column_size;
    column *next;
};
```

- Запись

```
struct row {
    bool is_valid;
    table *table;
```

```
    void **data;
};
```

- Тип запроса

```
enum query_type {
    SELECT;
    UPDATE;
    DELETE;
};
```

- Запрос

```
enum query_type {
    enum query_type query_type;
    table *table;
    char **column_name;
    void **column_value;
};
```

- Запрос соединения

```
enum query_join {
    table *right_table;
    table *left_table;
    char *right_column_name;
    char *left_column_name;
};
```

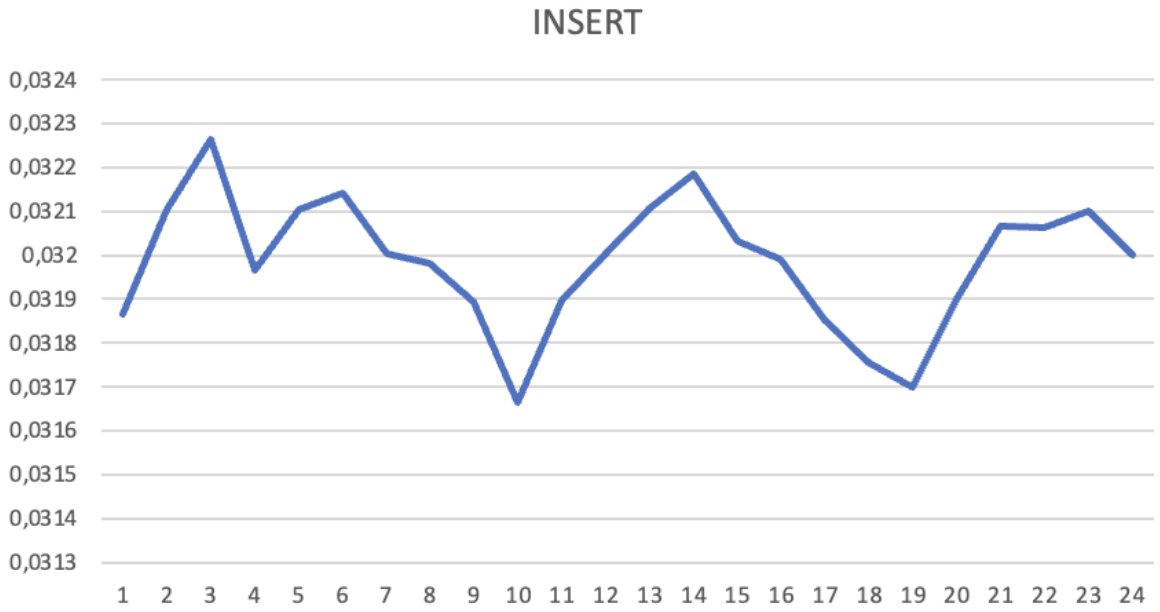
Основные правила размещения данных:

- Самой первой в файле идет мета-страница, которая содержит информацию о базе данных, содержит заголовок страницы и заголовки таблиц, которые находятся в данном файле. Если таблиц будет слишком много и место в этой мета-странице закончится, то создается новая

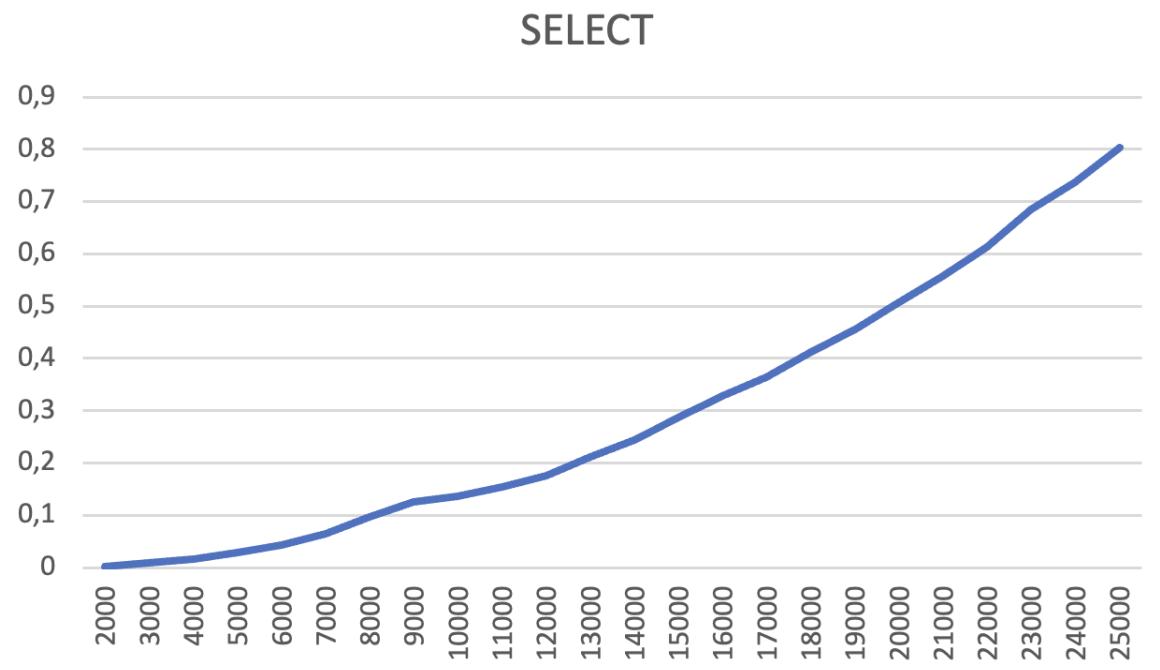
- В заголовке базы данных лежит информация о количестве таблиц, ее название, количество страниц и номер последней из них
- Заголовок страницы содержит информацию о том, какому отношению (таблице) она принадлежит, сколько свободного места осталось в странице, номер самой страницы и следующей, если она есть
- Страница может быть либо служебной, либо принадлежать только одной таблице
- Для создания таблицы прежде всего нужно создать схему для нее, которая будет содержать информацию о полях: тип, размер, название, количество
- Заголовок таблицы содержит всю информацию о ней: название, схему таблицы, количество страниц, на которых она размещается, номер первой и последней странички
- На странице после ее заголовка идет целочисленное значение, которое говорит о количестве колонок, после чего идут записи самих колонок: их названия, размер, смещение, тип. После колонок идут непосредственно сами записи таблицы.
- Для вставки данных в таблицу нам нужно знать ее название, чтобы, опираясь на схему, знать длину записи и поля, отступы для каждого из полей и количество колонок. В случае недостатка места на странице создается новая, для нее дублируется схема таблицы и номер новой страницы записывает в поле “номер следующей страницы” предыдущей страницы
- Для чтения данных таблицы сначала из ее заголовка мы узнаем номер страницы, с которой начинать искать. Далее после чтения заголовка и колонок узнаем размер одной записи, отдельных колонок и их отступ. После чего последовательно итерируемся по записям, проверяя заданное условие

Графики

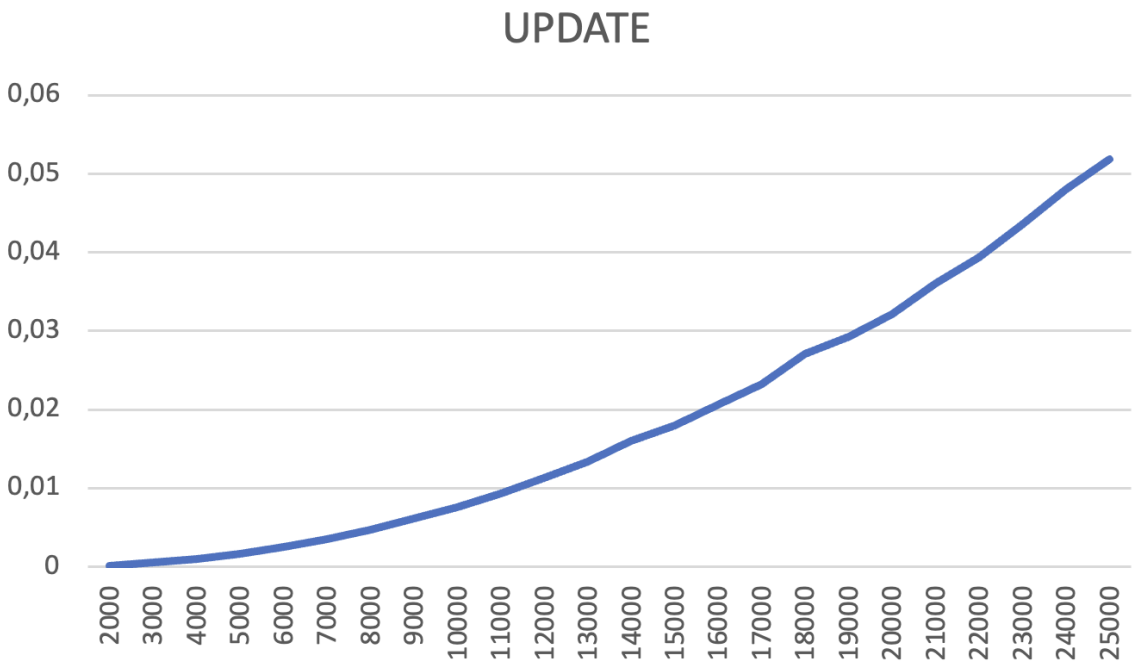
- Операция вставки



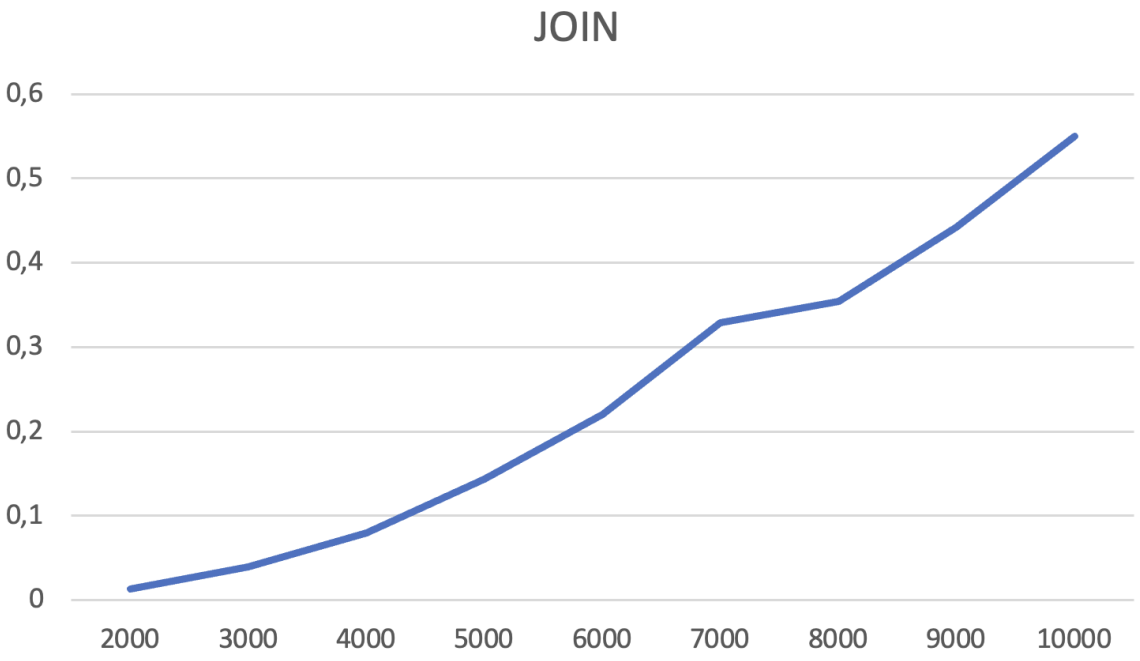
- Операция выборки



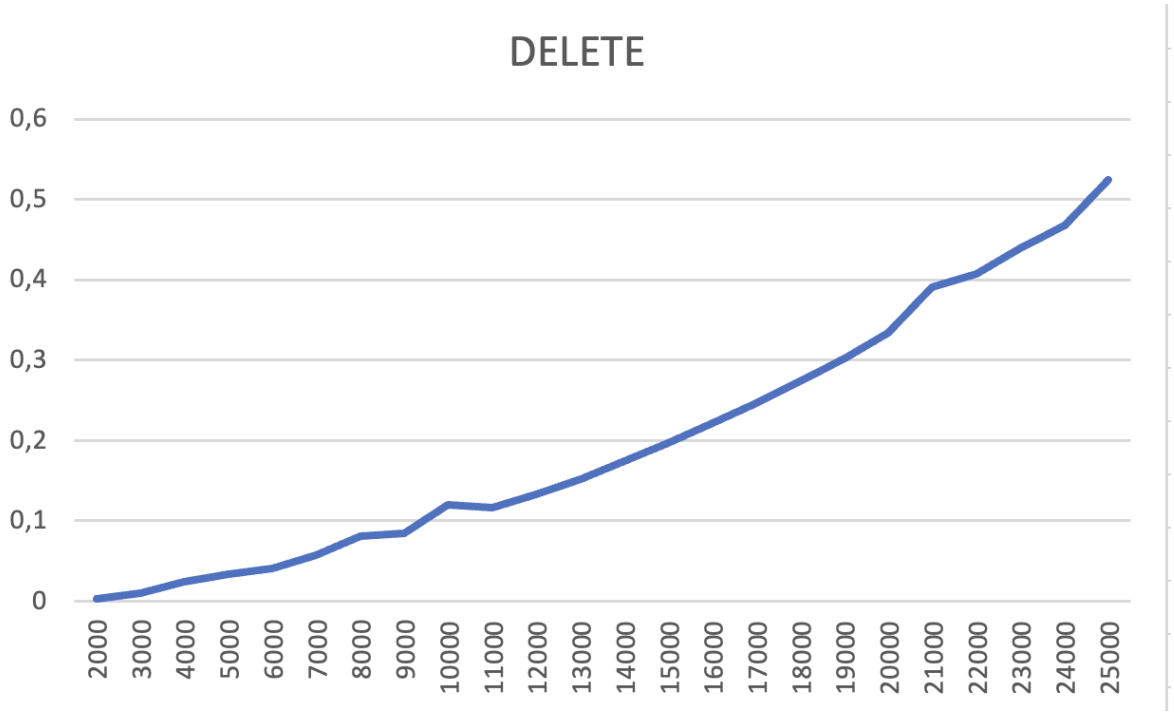
- Операция обновления



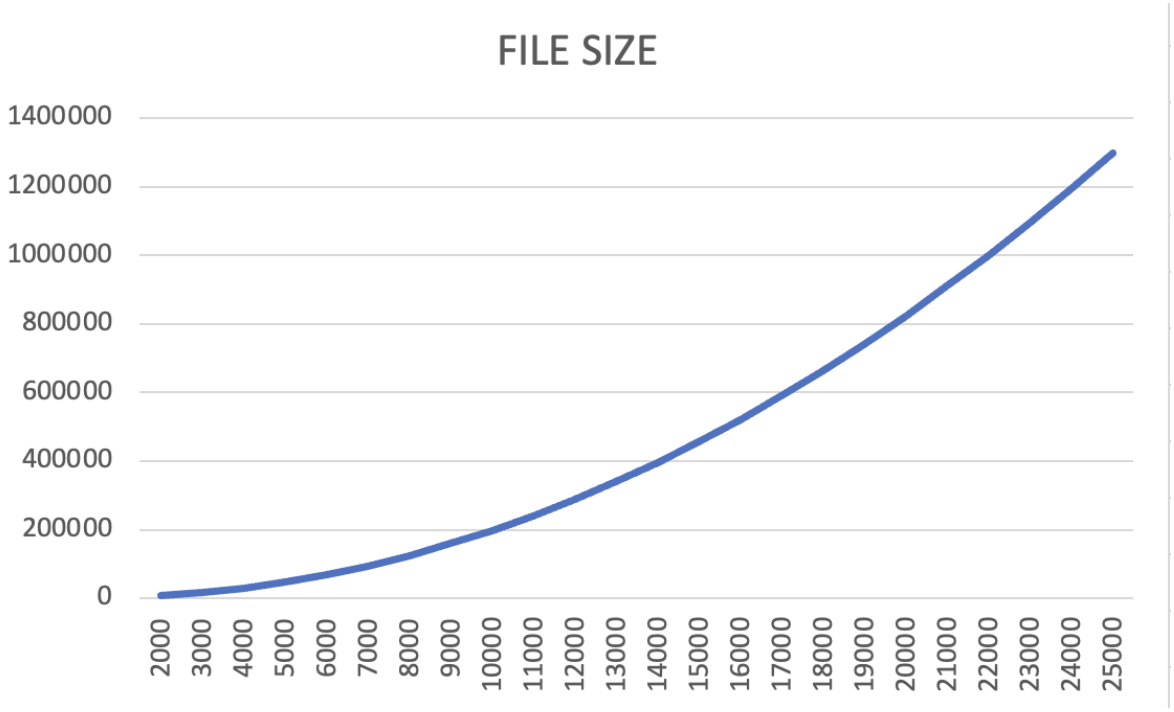
- Соединение таблиц



- Операция удаления



- Зависимость размера файла от количества вставленных записей



Вывод

В ходе данной лабораторной работы был разработан модуль для хранения данных в виде строк в таблицах. Размер данных может достигать 10 ГБ. Были освоены операции работы с файлом, чтением и записи структур. В модуле реализована CRUD

функциональность, а так же возможность соединения таблиц. Модуль может работать под управлением ОС семейств Windows и *NIX.