Санкт-Петербургский Национальный Исследовательский Университет ИТМО

МФКТиУ, факультет ПИиКТ

Лабораторная работа №1 по предмету «Низкоуровневое программирование»

Преподаватель: Кореньков Юрий Дмитриевич

Выполнил: Стефан Лабович

Группа: P33102

Вариант: Реляционные таблицы

Санкт-Петербург, 2022

Цель работы:

Разработка модуля реализующий хранение в одном файле данных (выборку, размещение и гранулярное обновление) информации общим объёмом от 10GB соответсвтующего реляционным таблицам.

Описание:

Данные которые хранятся в файле, хранятся в трех разных логических видов блоках.

-Database block (Блок Базы Данных)

-Schema block (Блок Схемы)

-Record block (Блок Записи)

В каждом из блоках хранятся метаданные Блока, которые содержат информацию о типу блока, его идентификатор, идентификатор следующего блока, и смещение в этом блоке.

В блоке Базы Данных хранятся метаданные, содержащие информацию о названию базы данных (название файла в котором данные хранятся), величине блока, количеству блоков в файле, идентификатор первого блока схемы, идентификатор последнего блока схемы и идентификатор первог свободного блока. В одном файле хранится одна база данных.

В блоке Схемы хранятся данные связанные с таблицами, валидность таблицы, название таблицы, идентификатор первого блока записи, идентификатор последнего блока записи, количество аттрибутов таблицы, название, тип и величина каждого аттрибута, количество записей таблицы.

В блоке Записи хранятся элементы данных соответсвующей таблицы.

Публичный интерфейс:

Добавление, удаление и получение информации о таблицам:

```
void insert_table_to_schema(table* tb, database* db, FILE* f);
bool delete_table_from_schema(char* name, database* db, FILE* f);
table* get_table_from_schema(char* name, database* db, FILE* f);
```

Добавление, удаление, изменение и получение информации о элементах данных:

```
void insert_record_to_table(record* r, table* tb, database* db, FILE* f);

void select_records_from_table(uint8_t num_cols,
                               char** view_cols,
                               condition* condition,
                               table* tb,
                               database* db,
                               FILE* f,
                               FILE* output);

void update_records_in_table(column_to_update* col,
                             condition* cond,
                             table* tb,
                             database* db,
                             FILE* f);
```

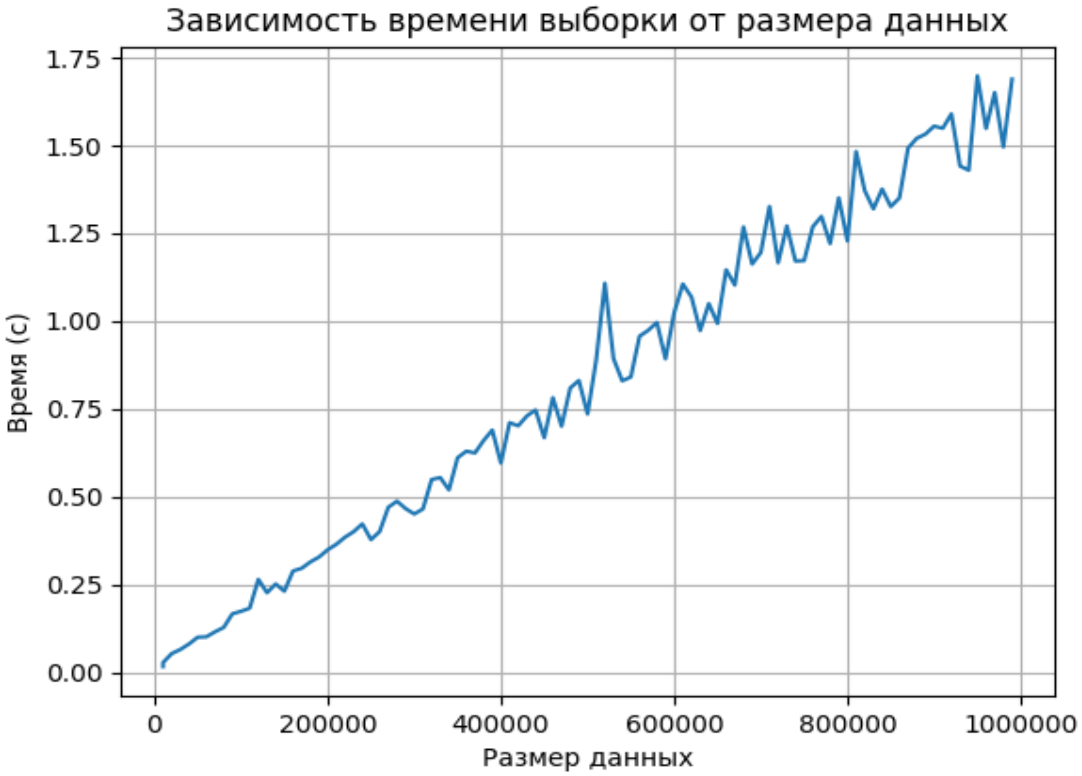
```
void delete_records_from_table(condition* cond,
                               table* tb,
                               database* db,
                               FILE* f);

void select_records_from_table_inner_join(table_to_join* left,
                                          table_to_join* right,
                                          database* db,
                                          FILE* f,
                                          FILE* output);
```

Результаты тестов:

Вставка:
Для теста вставки, мы 100 раз засечем время вставки 10 000 элементов в одну и ту же таблицу.

Зависимость линейная.
Выборка:
Для теста выборки, мы 100 раз засечем время выборки, и после каждой добавим 10 000 элементов в одну и ту же таблицу.



Зависимость линейная.

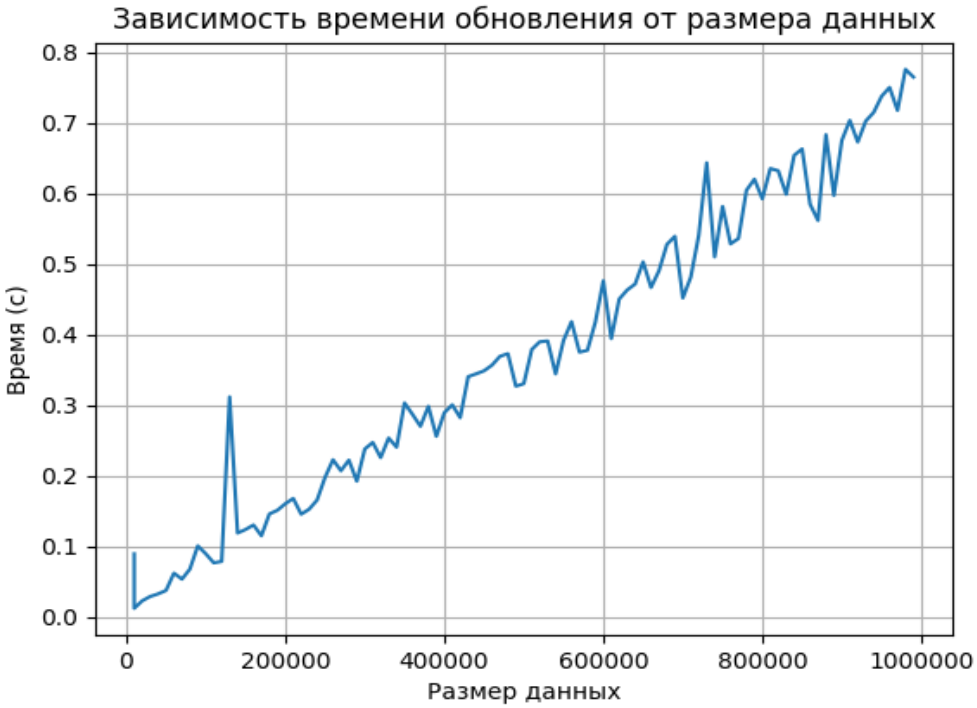
Удаление:

Для теста удаления, мы 100 раз засечем время удаления 5 000 элементов, и после каждого добавим 10 000 элементов в одну и ту же таблицу.

Зависимость линейная.

Обновление:

Для теста обновления, мы 100 раз засечем время обновления 5 000 элементов, и после каждого добавим 10 000 элементов в одну и ту же таблицу.



Зависимость линейная.

Слияние двух таблиц:

Для теста слияния, мы 20 раз засечем время слияния двух таблиц, будем увеличивать их размер за 100 и 50 элементов между измерениями.

Зависимость степенная.
Если построить график зависимости времени от производения количества элементов данных первой таблицы и второй таблицы получаем следующий график.

Отсюда и доказательство что O(n*m), где n размер первой, а m второй таблицы.
Использование файлового пространства:
Переиспользование файлового пространства

Создадим новый файл:
Добавим туда 100 000 записей:
-Величина файла: 4132KB
Удалим оттуда 100 000 записей:
-Величина файла: 4132KB
Добавим туда 100 000 записей:
-Величина файла: 4132KB
Ясно что файловое пространство переиспользуется.

Реальное использование файлового пространства
-Величина файла: 4132KB
-Количество записей в файле: 100 000
-Величина одной записи: 42B
-Величина метаданных блока:16B
-Доступное для размещения элементов данных пространство блока: 4080B
-Количество блоков со злементами данных: 100000*42B/4080KB=1029.36=1030
-1 Блок базы данных
-1 Блок схемы
-1 Первый свободный блок

-Количество всех блоков в файле:1033

-Величина всех блоков в файле в KB = 1033*4096B/1024B= 4132KB

Выводы:

Во время выполнения данной лабораторной работы, мне удалось создать модуль реализующий хранение в одном файле данных (выборку, размещение и гранулярное обновление) соответствующий реляционным таблицам.