Санкт-Петербургский Национальный Исследовательский Университет ИТМО

МФКТиУ, факультет ПИиКТ

Лабораторная работа №2 по предмету «Низкоуровневое программирование»

Преподаватель: Кореньков Юрий Дмитриевич

Выполнил: Стефан Лабович

Группа: P33102

Вариант: AQL

Санкт-Петербург, 2022

Цель работы:

Реализовать модуль для разбора некоторого достаточного подмножества языка запросов AQL в соответствии с реляционным таблицам, используя средство синтаксического анализа по выбору. Должна быть обеспечена возможность описания команд создания, выборки, модификации и удаления элементов данных.

Запросы отвечающие за операции над элементамы схемы:

Создание таблицы:

```
CREATE TABLE <table_name> {
      <column_name>: <type>,
                ...,
      <column_name>: <type>
};
```

Типы колонок:

```
INTEGER
BOOLEAN
FLOAT
VARCHAR(<size>)
```

Удаление таблицы:

```
REMOVE TABLE <table_name>;
```

Запросы отвечающие за операции над элементамы данных:

Добавление одного элемента данных:

```
INSERT
      {<col_name>: <value>, ..., <col_name>: <value>}
IN <table_name>;
```

Добавление больше элементов данных:

```
FOR <row_name> IN
      [
            {<col_name>: <value>, ..., <col_name>: <value>},
                                    ...,
            {<col_name>: <value>, ..., <col_name>: <value>}
      ]
INSERT <row_name> IN <table_name>;
```

Обновление всех элементов данных:

```
UPDATE
      WITH {<col_name>: <value>, ..., <col_name>: <value>}
IN <table_name>;
```

Обновление элементов данных которые соответствуют условиям:

```
FOR <row_name> IN <table_name>
      FILTER <col_name> == <value>
      UPDATE <row_name>
            WITH {<col_name>: <value>, ..., <col_name>: <value>}
      IN <table_name>;
```

Удаление всех элементов данных:

```
REMOVE IN <table_name>;
```

Удаление элементов данных которые соответствуют условиям:

```
FOR <row_name> IN <table_name>
      FILTER <col_name> == <value>
      REMOVE <row_name> IN <table_name>;
```

Выборка всех поль всех элементов данных:

```
FOR <row_name> IN <table_name>
      RETURN <row_name>;
```

Выборка желаемых поль всех элементов данных:

```
FOR <row_name> IN <table_name>
      RETURN
      {
            <field_name>,
                  ...,
            <field_name>
      };
```

Выборка всех поль всех элементов данных которые соответствуют условиям:

```
FOR <row_name> IN <table_name>
      FILTER <col_name> == <value>
      RETURN <row_name>;
```

Выборка желаемых поль всех элементов данных которые соответствуют условиям:

```
FOR <row_name> IN <table_name>
      FILTER <col_name> == <value>
      RETURN
      {
            <field_name>,
                  ...,
            <field_name>
      };
```

Выборка всех поль всех элементов данных из двух соединенных таблиц (INNER JOIN):

```
FOR <left_row_name> IN <left_table_name>
      FOR <right_row_name> IN <right_table_name>
```

```
            FILTER <left_row_name>.<field> == <right_row_name>.<field>
            RETURN <left_row_name>, <right_row_name>;
```

Выборка желаемых поль всех элементов данных из двух соединенных таблиц (INNER JOIN):

```
FOR <left_row_name> IN <left_table_name>
      FOR <right_row_name> IN <right_table_name>
            FILTER <left_row_name>.<field> == <right_row_name>.<field>
            RETURN
            {
                  <row_name>.<field_name>,
                        ...,
                  <row_name>.<field_name>
            };
```

Структуры которые используются для того чтобы построить AST:
Структура которая представляет самый запрос:

```
struct query {
  enum query_type type;
  create_table* ct;
  remove_table* rt;
  insert_records* ir;
  update_records* ur;
  remove_records* rem_r;
  return_records* ret_r;
};
```

Структуры соответсвтующие разным видам запросов:

```
struct create_table {
  char name[MAX_NAME_LENGTH];
  column* column;
};

struct remove_table {
    char name[MAX_NAME_LENGTH];
};

struct insert_records {
    enum insert_type type;
    char row_name[MAX_NAME_LENGTH];
    char table_name[MAX_NAME_LENGTH];
    record_fields* rfs;
};

struct update_records {
    enum update_type type;
    loop* loop;
    char row_name[MAX_NAME_LENGTH];
    char table_name[MAX_NAME_LENGTH];
    record_field* rf;
    filter* filter;
```

```
};

struct remove_records {
    enum remove_type type;
    loop* loop;
    char row_name[MAX_NAME_LENGTH];
    char table_name[MAX_NAME_LENGTH];
    filter* filter;
};

struct return_records {
    enum return_type type;
    loop* left_loop;
    loop* right_loop;

    comparison* join_condition;

    filter* filter;
    char left_return_name[MAX_NAME_LENGTH];
    char right_return_name[MAX_NAME_LENGTH];
    field* f;
};
```

Дополнительные структуры которые используются запросами:

```
struct loop {
    char row_name[MAX_NAME_LENGTH];
    char table_name[MAX_NAME_LENGTH];
};

struct column {
  char name[MAX_NAME_LENGTH];
  enum data_type type;
  uint8_t size;

  column* next;
};

struct record_fields {
    record_field* rf;
    record_fields* next;
};

struct record_field {
    char* name;
    literal* value;

    record_field* next;
};

struct literal {
    enum data_type type;
    union literal_value* value;
};

union literal_value {
```

```
    int32_t i;
    float f;
    bool b;
    char* s;
};

struct field {
    char* row_name;
    char* name;
    field* next;
};

struct filter {
    comparison* comparison;
    filter* next;
};

struct comparison {
    enum comparison_type type;

    literal* left_operand;
    literal* right_operand;
    enum relation relation;

    enum comparison_relation comparison_relation;

    comparison* next;

};
```

Примеры запросов и соответствующих AST:

Создание новой таблицы:

```
CREATE TABLE studs
{id: INTEGER, name:VARCHAR(32), age: INTEGER, active: BOOLEAN};
```

```
|- Query: 'type': 'Create Table'
  `|- Table: 'name': 'studs'
     `|- Column: 'name': 'id', 'type': 0, 'size': 4
      |- Column: 'name': 'name', 'type': 3, 'size': 32
      |- Column: 'name': 'age', 'type': 0, 'size': 4
      |- Column: 'name': 'active', 'type': 2, 'size': 1
```

Удаление таблицы:

```
REMOVE TABLE tb1;
```

```
|- Query: 'type': 'Remove Table'
  `|- Table: 'name': 'tb1'
```

Добавление новых элементов данных:

```
INSERT {id:5, name:'Stefan', age:22, active: TRUE} IN studs;
```

```
|- Query: 'type': 'Insert Records'
  `|- Table: 'name' : 'studs'
```

```
     `|- Record field: 'name' : 'id', 'value' : 5
      |- Record field: 'name' : 'name', 'value' : Stefan
      |- Record field: 'name' : 'age', 'value' : 22
      |- Record field: 'name' : 'active', 'value' : 0
```

Удаление элементов данных с фильтрами:

```
FOR stud in studs
   FILTER stud.id > 0 AND stud.id < 10
   FILTER stud.age < 20
   REMOVE stud in studs;
```

```
|- Query: 'type': 'Remove Records'
  `|- Table: 'name': 'studs'
     `|- Read Row: 'name' : 'stud'
        `|- Filter:
           `|- 'LeftOperand':'stud.id', 'RightOperand':0 'Relation':'>'
            |- AND
            |- 'LeftOperand':'stud.id', 'RightOperand':10 'Relation':'<'
         |- Filter:
           `|- 'LeftOperand':'stud.age', 'RightOperand':20 'Relation':'<'
```

Обновление элементов данных с фильтрами:

```
FOR stud IN studs
    FILTER stud.id > 0 AND stud.id < 10
    FILTER 15 > 0
    UPDATE stud
        WITH {id:1, name:'Stefan', age:22}
    IN studs;
```

```
|- Query: 'type': 'Update Records'
  `|- Table: 'name' : 'studs'
     `|- Row: 'name' : 'stud'
        `|- Record field: 'name' : 'id', 'value' : 1
         |- Record field: 'name' : 'name', 'value' : Stefan
         |- Record field: 'name' : 'age', 'value' : 22
         |- Filter:
           `|- 'LeftOperand':'stud.id', 'RightOperand':0 'Relation':'>'
            |- AND
            |- 'LeftOperand':'stud.id', 'RightOperand':10 'Relation':'<'
         |- Filter:
           `|- 'LeftOperand':15, 'RightOperand':0 'Relation':'>'
```

Выборка элементов данных с фильтрами:

```
FOR stud in studs
   FILTER stud.id > 0 AND stud.id < 10
   FILTER stud.age < 20
   RETURN stud;
```

```
|- Query: 'type': 'Remove Records'
  `|- Table: 'name': 'studs'
     `|- Read Row: 'name' : 'stud'
        `|- Filter:
           `|- 'LeftOperand':'stud.id', 'RightOperand':0 'Relation':'>'
            |- AND
            |- 'LeftOperand':'stud.id', 'RightOperand':10 'Relation':'<'
```

```
         |- Filter:
           `|- 'LeftOperand':'stud.age', 'RightOperand':20 'Relation':'<'
```

Выборка элементов данных с отношениями и с фильтрами:

```
FOR stud IN studs
    FOR prof IN profs
        FILTER stud.id = prof.id
        FILTER stud.id > 0 AND stud.id < 10
        FILTER prof.age > 40
        RETURN stud, prof;
```

```
|- Query: 'type': 'Return Records'
  `|- Table: 'name': 'studs'
     `|- Read Row: 'name' : 'stud'
      |- Table: 'name': 'profs'
        `|- Read Row: 'name' : 'prof'
         |-'LeftOperand':'stud.id','RightOperand':'prof.id' 'Relation':'=='
         |- Filter:
           `|- 'LeftOperand':'stud.id', 'RightOperand':0 'Relation':'>'
            |- AND
            |- 'LeftOperand':'stud.id', 'RightOperand':10 'Relation':'<'
         |- Filter:
           `|- 'LeftOperand':'prof.age', 'RightOperand':40 'Relation':'>'
         |- Return Row: 'name': 'stud', Return Row: 'name': 'prof'
```

Выводы:

Во время выполнения данной лабораторной работы, мне удалось создать модуль для разбора некоторого достаточного подмножества языка запросов AQL в соответствии с реляционным таблицам, используя Flex и Bison.