Университет ИТМО
Факультет программной инженерии и вычислительной техники

Лабораторная работа №1 по дисциплине «Низкоуровневое программирование»

Вариант №3, Граф

Выполнил:
Ерехинский Андрей Владимирович
Студент группы P33312

Преподаватель:
Кореньков Ю.Д.

Санкт-Петербург
2023

Задание: Использовать средство синтаксического анализа по выбору, реализовать модуль для разбора некоторого достаточного подмножества языка запросов по выбору в соответствии с вариантом формы данных. Должна быть обеспечена возможность описания команд создания, выборки, модификации и удаления элементов данных.

Описание:

В токенайзере Flex (lexer.l) предусмотрены токены для обработки базовых запросов. На основе этого модуля создаются структуры узла дерева ast_node, в которых отображены тип узла дерева, а также левый и правый потомки этого узла дерева.

```
typedef struct ast_node {
    type_node type;
    union {
        int32_t int_val;
        double double_val;
        bool bool_val;
        char* str_val;
    };
    ast_node* left;
    ast_node* right;
} ast_node;
```

В парсере Bison (parser.y) вызываются методы для создания узлов. Методы определены в файлах tree.c tree.h. Арифметические операции не поддерживаются. В модуле tree.c метод для вывода (print_node()).

Примеры запросов:

Запрос на один тип узла, но с двумя условиями.

```
select{
    message(filter: or(eq(length, 123.0), le(body_length, topic_length)))
}
type_query: select
query_set:
  query:
    object:
      schema_name: message
      create_values: <undefined>
      filter:
        operation:
          type_operation: or
          operation1:
            type_operation: equal
            key: length
            value:
              type_value: double
              data: 123.000000
          operation2:
            type_operation: less
            first_key: body_length
            second_key: topic_length
    query_set: <undefined>
```

Запрос на два типа узлов, одно условие.

```
select{
    email(filter: eq(id, 1)) {
        message
    }
}
type_query: select
query_set:
  query:
    object:
      schema_name: email
      create_values: <undefined>
      filter:
        operation:
          type_operation: equal
          key: id
          value:
            type_value: integer
            data: 1
    query_set:
      query:
        object:
          schema_name: message
          create_values: <undefined>
          filter: <undefined>
        query_set: <undefined>
```

Запрос на вставку одного типа узла

```
insert{
    email(values: [{id : 111}])
}
type_query: insert
query_set:
  query:
    object:
      schema_name: email
      create_values:
        element_set:
          element:
            key: id
            value:
              type_value: integer
              data: 111
      filter: <undefined>
    query_set: <undefined>
```

Запрос на вставку сразу двух типов узлов

```
insert{
    email(values: [{id : 111}], filter: eq(id, "value")) {
        message(values: [{id : 111}])
    }
}
type_query: insert
query_set:
  query:
    object:
      schema_name: email
      create_values:
        element_set:
          element:
            key: id
            value:
              type_value: integer
              data: 111
      filter:
        operation:
          type_operation: equal
          key: id
          value:
            type_value: string
            data: "value"
    query_set:
      query:
        object:
          schema_name: message
          create_values:
            element_set:
              element:
                key: id
                value:
                  type_value: integer
                  data: 111
          filter: <undefined>
          query_set: <undefined>
```

Запрос на обновление одного типа узла.

```
update{
    email (values: [{name : "Andrey"}])
}
type_query: update
query_set:
  query:
    object:
      schema_name: email
      create_values:
        element_set:
          element:
            key: name
            value:
              type_value: string
              data: "Andrey"
      filter: <undefined>
    query_set: <undefined>
```

Запрос на обновление одного типа узла через условие на связанный узел

```
update{
    email(filter: eq(id, 1)) {
        message(values: [{recipient : "Andrey"}])
    }
}
type_query: update
query_set:
  query:
    object:
      schema_name: email
      create_values: <undefined>
      filter:
        operation:
          type_operation: equal
          key: id
          value:
            type_value: integer
            data: 1
    query_set:
      query:
        object:
          schema_name: message
          create_values:
            element_set:
              element:
                key: recipient
                value:
                  type_value: string
                  data: "Andrey"
          filter: <undefined>
        query_set: <undefined>
```

Удаление двух типов узлов через условие на каждом

```
delete{
    email(filter: eq(id, 1)) {
        message(filter: le(length, 50))
    }
}
type_query: delete
query_set:
  query:
    object:
      schema_name: email
      create_values: <undefined>
      filter:
        operation:
          type_operation: equal
          key: id
          value:
            type_value: integer
            data: 1
    query_set:
      query:
        object:
          schema_name: message
          create_values: <undefined>
          filter:
            operation:
              type_operation: less
              key: length
              value:
                type_value: integer
                data: 50
        query_set: <undefined>
```

Вывод: В ходе выполнения было разобрано абстрактное синтаксическое дерево, инструменты Flex Bison. Разработан модуль распознавания запросов на вариации языка запросов GraphQL.