Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет Программной инженерии и компьютерной техники

Низкоуровневое программирование

Лабораторная работа №2

Вариант №4

Группа: P33302

Выполнил: Варюхин И.А.

Проверил: Кореньков Ю.Д.

Санкт-Петербург

2023

1. Задача

Основная цель лабораторной работы - 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.

- Спроектировать архитектуру модуля
- Провести изучение технологий flex, bison
- Изучить грамматику языка GraphQL
- Разработать собственную грамматику
- Написать тест-кейсы запросов и прогнать через приложение изучив полученный результат

2. Представление AST

Структура для хранения узла AST дерева. Узел может хранить примитивные значения и указатель на строку, содержащую идентификатор или значение объекта. Также он может иметь неопределённое количество потомков.

```
struct ast_node {
    union {
        char *string_value;
        int int_value;
        float float_value;
        bool bool_value;
    } value;
    enum ast_node_type type;
    size_t children_count;
    struct ast_node **children;
};
```

Реализация лексического анализатора

lexer.l

```
%{
    #include <string.h>
    #include <stdlib.h>
    #include <stdio.h>
    #include <stdbool.h>
    #include "parser.tab.h"
%}
```

```
id [a-zA-Z]+
float [-+]?[0-9]*\.?[0-9]+([eE][-+]?[0-9]+)?
integer [-+]?[0-9]+
boolean true|false
string_w_quotes \"[^\"]*\"

%%

"{" { return L_BRACE; }
"}" { return R_BRACE; }
"(" { return L_BRACKET; }
")" { return R_BRACKET; }
";" { return SEMICOLON; }
":" { return COLON; }
"=" { return OP_EQUALS; }
"<" { return OP_LESS; }
">" { return OP_GREATER; }

select { return SELECT; }
insert { return INSERT; }
update { return UPDATE; }
delete { return DELETE; }

{boolean} {
    yylval.bval = (strcmp(yytext, "true") == 0);
    return VAL_BOOL;
}

{id} {
    yylval.sval = strdup(yytext);
    return ID;
}

{integer} {
    yylval.ival = atoi(yytext);
    return VAL_INTEGER;
}

{float} {
    yylval.fval = atof(yytext);
    return VAL_FLOAT;
}

{string_w_quotes} {
    yylval.sval = strdup(yytext);
    return VAL_STRING;
}

[ \t\n] ;
. ;

%%

int yywrap() { return 1; }
```

Реализация парсера

```
input:
	SELECT select_query SEMICOLON { set_root_ast_node($2); $$ = $2; }
|	INSERT insert_query SEMICOLON { set_root_ast_node($2); $$ = $2; }
|	UPDATE update_query SEMICOLON { set_root_ast_node($2); $$ = $2; }
|	DELETE delete_query SEMICOLON { set_root_ast_node($2); $$ = $2; }
;

select_query:
	L_BRACE condition_body repr_body R_BRACE	{
		$$ = create_ast_node(AST_NODE_QUERY_SELECT, 2, $2, $3);
		}
;

update_query:
	L_BRACE condition_body entity_body repr_body R_BRACE	{
		$$ = create_ast_node(AST_NODE_QUERY_UPDATE, 3, $2, $3, $4);
		  }
;

delete_query:
	L_BRACE condition_body repr_body R_BRACE	{
		$$ = create_ast_node(AST_NODE_QUERY_DELETE, 2, $2, $3);
		}
;

insert_query:
	L_BRACE entity_body repr_body R_BRACE	{
		$$ = create_ast_node(AST_NODE_QUERY_INSERT, 2, $2, $3);
		}
;

repr_body:
	L_BRACE R_BRACE			{ $$ =
create_ast_node(AST_NODE_REPR_BODY, 0); }
|	L_BRACE repr_list R_BRACE	{ $$ = create_ast_node(AST_NODE_REPR_BODY,
1, $2); }
;

repr_list:
	repr field		{
		$$ = create_ast_node(AST_NODE_REPR_LIST, 1, $1);
	}
|	repr_list repr_field	{
		add ast node($1, $2);
	}
;

repr_field:
	ID COLON repr_list	{
		$$ = create_ast_node(AST_NODE_REPR_FIELD, 2,
create_identifier_ast_node($1), $3);
		}
|	ID			{
		$$ = create_ast_node(AST_NODE_REPR_FIELD, 1,
create_identifier_ast_node($1));
		}
;

condition_body:
|	ID L_BRACKET condition R_BRACKET	{
		$$ = create_ast_node(AST_NODE_CONDITION_BODY, 2,
```

```
create_identifier_ast_node($1), $3);
              }
;

condition:
       L_BRACE property_list R_BRACE  { $$ = create_ast_node(AST_NODE_CONDITION,
1, $2); }
;

entity_body:
       ID L_BRACKET L_BRACE entity_list R_BRACE R_BRACKET {
              $$ = create_ast_node(AST_NODE_ENTITY_BODY, 2,
create_identifier_ast_node($1), $4);
              }
;

property_list:
       property                       { $$ =
create_ast_node(AST_NODE_PROPERTY_LIST, 1, $1); }
|      property_list property         { add_ast_node($1, $2); }
;

entity_list:
       entity               { $$ = create_ast_node(AST_NODE_ENTITY_LIST, 1, $1);
}
|      entity_list entity     { add_ast_node($1, $2); }
;

property:
       ID OP_EQUALS VAL_STRING               {
              $$ = create_ast_node(AST_NODE_OP_EQ, 2,
create_identifier_ast_node($1), create_string_literal_ast_node($3));
              }
|      ID OP_LESS VAL_INTEGER        {
              $$ = create_ast_node(AST_NODE_OP_LESS, 2,
create_identifier_ast_node($1), create_int_literal_ast_node($3));
              }
|      ID OP_GREATER VAL_INTEGER             {
              $$ = create_ast_node(AST_NODE_OP_GREATER, 2,
create_identifier_ast_node($1), create_int_literal_ast_node($3));
              }
|      ID OP_EQUALS VAL_INTEGER       {
              $$ = create_ast_node(AST_NODE_OP_EQ, 2,
create_identifier_ast_node($1), create_int_literal_ast_node($3));
              }
|      ID OP_LESS VAL_FLOAT          {
              $$ = create ast node(AST NODE OP LESS, 2,
create_identifier_ast_node($1), create_float_literal_ast_node($3));
              }
|      ID OP_GREATER VAL_FLOAT               {
              $$ = create ast node(AST NODE OP GREATER, 2,
create_identifier_ast_node($1), create_float_literal_ast_node($3));
              }
|      ID OP_EQUALS VAL_FLOAT        {
              $$ = create_ast_node(AST_NODE_OP_EQ, 2,
create identifier ast node($1), create float literal ast node($3));
              }
|      ID OP_LESS VAL_BOOL           {
              $$ = create_ast_node(AST_NODE_OP_LESS, 2,
create identifier ast node($1), create bool literal ast node($3));
              }
|      ID OP_GREATER VAL_BOOL        {
              $$ = create_ast_node(AST_NODE_OP_GREATER, 2,
create_identifier_ast_node($1), create_bool_literal_ast_node($3));
```

```
                }
|       ID OP_EQUALS VAL_BOOL          {
                $$ = create_ast_node(AST_NODE_OP_EQ, 2,
create_identifier_ast_node($1), create_bool_literal_ast_node($3));
                }
;

entity:
        ID COLON L_BRACE field_list R_BRACE     {
                $$ = create_ast_node(AST_NODE_ENTITY, 2,
create identifier ast node($1), $4);
                }
|       field_list                              { $$ =
create_ast_node(AST_NODE_ENTITY, 1, $1); }
;

field_list:
        field                 { $$ = create_ast_node(AST_NODE_FIELD_LIST, 1, $1); }
|       field_list field        { add_ast_node($1, $2); }
;

field: ID COLON VAL_BOOL        {
                $$ = create_ast_node(AST_NODE_FIELD, 2,
create identifier ast node($1), create bool literal ast node($3));
                }
|       ID COLON VAL_INTEGER    {
                $$ = create_ast_node(AST_NODE_FIELD, 2,
create_identifier_ast_node($1), create_int_literal_ast_node($3));
                }
|       ID COLON VAL_FLOAT      {
                $$ = create_ast_node(AST_NODE_FIELD, 2,
create_identifier_ast_node($1), create_float_literal_ast_node($3));
                }
|       ID COLON VAL_STRING     {
                $$ = create_ast_node(AST_NODE_FIELD, 2,
create_identifier_ast_node($1), create_string_literal_ast_node($3));
                }
;
```

3. Результаты

Select запрос

```
select {
    person ({
        name = "vova"
    }) {
        name
        height
    }
};
```

AST

```
AST_NODE_QUERY_SELECT
|       AST_NODE_CONDITION_BODY
|       |       person
|       |       AST_NODE_CONDITION
|       |       |       AST_NODE_PROPERTY_LIST
|       |       |       |       AST_NODE_OP_EQ
|       |       |       |       |       name
|       |       |       |       |       '"vova"'
|       AST_NODE_REPR_BODY
|       |       AST_NODE_REPR_LIST
|       |       |       AST_NODE_REPR_FIELD
|       |       |       |       name
|       |       |       AST_NODE_REPR_FIELD
|       |       |       |       height
```

Insert запрос

```
insert {
    person ({
        name: "vova"
        age: 20
        height: 170.5
    }) {}
};
```

AST

```
AST_NODE_QUERY_INSERT
|       AST_NODE_ENTITY_BODY
|       |       person
|       |       AST_NODE_ENTITY_LIST
|       |       |       AST_NODE_ENTITY
|       |       |       |       AST_NODE_FIELD_LIST
|       |       |       |       |       AST_NODE_FIELD
|       |       |       |       |       |       name
|       |       |       |       |       |       '"vova"'
|       |       |       |       |       AST_NODE_FIELD
|       |       |       |       |       |       age
|       |       |       |       |       |       20
|       |       |       |       |       AST_NODE_FIELD
|       |       |       |       |       |       height
|       |       |       |       |       |       170.500000
|       AST_NODE_REPR_BODY
```

Update запрос

```
update {
    person ({
        name = "vova"
    })
    person ({
        name: "alexandr"
        height: 171
        married: true
    })
    {
        name
        married
    }
};
```

AST

```
AST_NODE_QUERY_UPDATE
|       AST_NODE_CONDITION_BODY
|       |       person
|       |       AST_NODE_CONDITION
|       |       |       AST_NODE_PROPERTY_LIST
|       |       |       |       AST_NODE_OP_EQ
|       |       |       |       |       name
|       |       |       |       |       '"vova"'
|       AST_NODE_ENTITY_BODY
|       |       person
|       |       AST_NODE_ENTITY_LIST
|       |       |       AST_NODE_ENTITY
|       |       |       |       AST_NODE_FIELD_LIST
|       |       |       |       |       AST_NODE_FIELD
|       |       |       |       |       |       name
|       |       |       |       |       |       '"alexandr"'
|       |       |       |       |       AST_NODE_FIELD
|       |       |       |       |       |       height
|       |       |       |       |       |       171
|       |       |       |       |       AST_NODE_FIELD
|       |       |       |       |       |       married
|       |       |       |       |       |       true
|       AST_NODE_REPR_BODY
|       |       AST_NODE_REPR_LIST
|       |       |       AST_NODE_REPR_FIELD
|       |       |       |       name
|       |       |       AST_NODE_REPR_FIELD
|       |       |       |       married
```

Delete запрос

```
delete {
    person ({
        name = "vova"
        height > 140.5
        age < 30
    }) {}
};
```

AST

```
AST_NODE_QUERY_DELETE
|       AST_NODE_CONDITION_BODY
|       |       person
|       |       AST_NODE_CONDITION
|       |       |       AST_NODE_PROPERTY_LIST
|       |       |       |       AST_NODE_OP_EQ
|       |       |       |       |       name
|       |       |       |       |       '"vova"'
|       |       |       |       AST_NODE_OP_GREATER
|       |       |       |       |       height
|       |       |       |       |       140.500000
|       |       |       |       AST_NODE_OP_LESS
|       |       |       |       |       age
|       |       |       |       |       30
|       AST_NODE_REPR_BODY
```

4. Выводы

Инструменты flex и bison действительно гибкие, с их помощью можно описать любую грамматику и обработать результат каким угодно образом.