УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерных технологий
Направление подготовки: Информатика и вычислительная техника

Низкоуровневое программирование

Лабораторная работа №2

Вариант 5

Выполнил:

Иевлев К. В.
Группа № P33302

Преподаватель:
Кореньков Ю. Д.

г. 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.

Язык запросов - AQL.

Используемые средства:

Bison – парсинг
Flex – токенизатор

Разработанный модуль:

Flex производит лексический анализ строки и образует токены. Полученные токены передаются в Bison, который проходит по полученным токенам и строит дерево.

Flex:

```
%option noyywrap
%option caseless
%option yylineno

%{
    #include <stdio.h>
    #include <string.h>

    #include "include/ast.h"
    #include "parser.tab.h"
    void print_error(char*);
%}

%option noyywrap nounput noinput debug

alpha   [A-Za-z_0-9 \t\r]
id    [a-zA-Z][a-zA-Z_0-9]*
int   ([-+]?[0-9])+
float   [+-]?([0-9]*[.])?[0-9]+
blank [ \t\r]
word ([a-zA-Z_][a-zA-Z0-9_]*)
quoted_string \"{word}*\"

%%

"for"       {return FOR;}
"insert"    {return INSERT;}
"update"    {return UPDATE;}
"remove"    {return REMOVE;}
"in"        {return IN;}
"filter"    {return FILTER;}
"with"      {return WITH;}
"return"    {return RETURN;}

"create"    {return CREATE;}
"drop"      {return DROP;}

"&&"        {yylval.logic_op_type = 1; return AND;}
"||"        {yylval.logic_op_type = 2; return OR;}
```

```
">"         {yylval.cmp_op_type = 1; return CMP;}
">="        {yylval.cmp_op_type = 2; return CMP;}
"<"         {yylval.cmp_op_type = 3; return CMP;}
"<="        {yylval.cmp_op_type = 4; return CMP;}
"=="        {yylval.cmp_op_type = 5; return CMP;}
"!="        {yylval.cmp_op_type = 6; return CMP;}

"float"     {yylval.type = 0; return TYPE;}
"string"    {yylval.type = 1; return TYPE;}
"int"       {yylval.type = 2; return TYPE;}
"bool"      {yylval.type = 3; return TYPE;}
"true"      {yylval.boolean_value = 1; return BOOL;}
"false"     {yylval.boolean_value = 0; return BOOL;}

"("         {return START_PARENTHESIS;}
")"         {return CLOSE_PARENTHESIS;}
"{"         {return START_BRACKET;}
"}"         {return CLOSE_BRACKET;}
";"         {return SEMICOLON;}
":"         {return COLON;}
"."         {return DOT;}
","         {return COMMA;}
"\""        {return QUOTE;}

{word}     {
    sscanf(yytext, "%s", yylval.string);
    return (STR);
}
{int}   {
    yylval.number_value = atoi(yytext);
    return (INT);
}
{float}     {
    yylval.float_num_value = atof(yytext);
    return (FLOAT);
}

[ \t]   {   }
[\n]    {   }
.           {
    print_error(yytext);
    return (UNEXPECTED_TOKEN);
}

%%

void print_error(char* token) {
    printf("Error in tokenizer token = %s \n", token);
}
```

Bison:

```
terminal:
|   TYPE { $$ = new_type($1); }
|   INT { $$ = new_number($1); }
|   FLOAT { $$ = new_float_number($1); }
|   BOOL { $$ = new_bool($1); }
|   QUOTE string_list QUOTE { $$ = new_string($2, NULL); }
;

query:
```

```
|   select_query
|   insert_query
|   delete_query
|   drop_query
|   update_query
|   create_query
;

select_query:
|   select_simple
|   select_with_filter
|   select_join_simple
|   select_join_with_filter
;
select_simple:
|   FOR row_alias IN STR RETURN STR { $$ = new_select($4, NULL, NULL, NULL,
$2); }
;
select_with_filter:
|   FOR row_alias IN STR FILTER filter_condition RETURN STR { $$ =
new_select($4, $6, NULL, NULL, $2); }
;

select_join_simple:
|   FOR row_alias IN STR FOR row_alias IN STR RETURN STR "," STR { $$ =
new_select($4, NULL, $8, $6, $2); }
;

select_join_with_filter:
|   FOR row_alias IN STR FOR row_alias IN STR FILTER filter_condition RETURN
STR "," STR { $$ = new_select($4, $10, $8, $6, $2); }
;

update_query:
|   update_simple
|   update_with_filter
;
update_simple:
|   FOR row_alias IN STR UPDATE STR WITH "{" values_list "}" IN STR { $$ =
new_update($4, NULL, $9); }
;
update_with_filter:
|   FOR row_alias IN STR FILTER filter_condition UPDATE STR WITH "{"
values_list "}" IN STR { $$ = new_update($4, $6, $11); }
;

delete_query:
|   delete_simple
|   delete_with_filter
;
delete_simple:
|   FOR row_alias IN STR REMOVE STR IN STR { $$ = new_delete($4, NULL, $2); }
;
delete_with_filter:
|   FOR row_alias IN STR FILTER filter_condition REMOVE STR IN STR { $$ =
new_delete($4, $6, $2); }
;

insert_query:
|   INSERT "{" values_list "}" IN STR { $$ = new_insert($6, $3); }
;

create_query:
|   CREATE STR "{" values_list "}" { $$ = new_create($2, $4); }
```

```
;

drop_query:
|   DROP STR { $$ = new_drop($2); }
;

values_list:
|   values_list "," values_pair { $$ = new_list($3, $1); }
|   values_pair { $$ = new_list($1, NULL); }
;

values_pair:
|   STR ":" terminal { $$ = new_pair($1, $3); }
;

filter_condition:
|   filter_condition "&&" filter_condition { $$ = new_where($2, $1, $3); }
|   filter_condition "||" filter_condition { $$ = new_where($2, $1, $3); }
|   "(" filter_condition ")" { $$ = $2; }
|   logic_statement
;

logic_statement:
|   column CMP terminal { $$ = new_compare($2, $1, $3); }
|   terminal CMP column { $$ = new_compare(switch_cmp_mode($2), $1, $3); }
|   column CMP column { $$ = new_compare($2, $1, $3); }
;

column:
|   STR DOT STR { $$ = new_name($1, $3); }
;

row_alias:
|   STR { $$ = new_name(NULL, $1); }
;

string_list:
|   string_list STR { $$ = new_string($1, $2); }
|   STR { $$ = new_string(NULL, $1); }
;
```

На основе результатов токенизации и расинга получается дерево

```
struct ast_node {
    ast_node_type type;
    fields fields_one;
    fields fields_two;
    ast_node* first;
    ast_node* second;
    ast_node* third;
};
```

Каждый узел содержит тип операции/объекта запроса, для отоброжания в текстовый вид, ссылки на поля, относящиеся к запросу.

С помощью show_tree.c происходит вывод структуры в текстовый формат

Примеры вывода:

Выборка с условием: `FOR x IN table FILTER x.a == "abc" || x.b  > 1 && x.c < 5 RETURN x;`

```
FOR {
   KEYS:
      KEY {
         row_alias: x
      }
   TABLE: table
   JOIN: NULL
   FOR: NULL
   FILTER:
      WHERE {
         COMPARE {
            KEY {
           row_alias: x
               column_name: a
            }
            EQUAL
            STRING { abc }
         }
         GREATER
        WHERE {
           COMPARE {
              COLUMN {
                  table: x
                  column_name: b
               }
               LESS
               NUMBER { 1 }
            }
            ||
            COMPARE {
               COLUMN {
                  table: x
                  column_name: c
               }
               LESS_OR_EQUAL
              NUMBER { 5 }
            }
         }
      }
}
```

Выборка с соединением таблиц с условием: `FOR x IN table_x FOR y IN table_y FILTER x.is_true == true && (x.text == "abc" || y.n > 4 || y.f < 1.1) RETURN x, y;`

```
FOR {
   KEYS:
      KEY {
         row_alias: x
      }
   TABLE: table_x
   JOIN: table_y
   FOR:
      KEY {
         row_alias: y
```

```
      }
   FILTER:
      WHERE {
         COMPARE {
            KEY {
               row_alias: x
               column_name: is_true
            }
            EQUAL
            BOOLEAN { TRUE }
         }
         ||
         WHERE {
            WHERE {
               COMPARE {
                  KEY {
                     row_alias: x
                     column_name: text
                  }
                  EQUAL
                  STRING { abc }
               }
               GREATER
               COMPARE {
                  KEY {
                     row_alias: y
                     column_name: n
                  }
                  GREATER_OR_EQUAL
                  NUMBER { 4 }
               }
            }
            GREATER
            COMPARE {
               KEY {
                  row_alias: y
                  column_name: f
               }
               LESS_OR_EQUAL
               FLOAT { 1.100000 }
            }
         }
      }
}
```

Требования:

Flex
Bison (3.8+)
Make

Запуск:

Выбираем makefile
Make all

Вывод:

В процессе выполнения лабораторной работы были использованы инструменты Flex и Bison. Рассмотрен язык запросов AQL, на основе которого создан модуль позволяющий составлять запросы для манипуляции с данными.