МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

**Отчет по лабораторной работе**

по дисциплине «Теория языков программирования и методы трансляции»

на тему: «Автоматизированное построение трансляторов с использованием генератора YACC»

Студент Бажин М.И. Шифр:160582

Институт приборостроения, автоматизации и информационных технологий

Специальность 09.03.01 «Информатика и вычислительная техника»

Группа 61-ИВТ

Руководитель Гордиенко А.П.

Орел 2018

Для создания генератора транслятора необходимо сперва настроить распознаватель лексем Lex. Поскольку результатом трансляции должно быть строковое значение, а в семантических определениях будут проведены манипуляции над строками, то в качестве типа атрибутной переменной терминалов будет использован тип myString.

Использование доменного строкового типа, а не стандартного, мотивировано тем, что объявленный тип поля в записях с вариантов должен иметь конечное значение.

Таким образом, файл Lex будет выглядеть следующим образом:

D [0-9]

L [a-zA-Z]

%%

{D}+ begin

yylval.yymyString:=yytext;

returni(num);

end;

"select" begin

yylval.yymyString:=yytext;

returni(select\_t);

end;

"from" begin

yylval.yymyString:=yytext;

returni(from\_t);

end;

"where" begin

yylval.yymyString:=yytext;

returni(where\_t);

end;

"or" begin

yylval.yymyString:=yytext;

returni(or\_t);

end;

"and" begin

yylval.yymyString:=yytext;

returni(and\_t);

end;

{L}({L}|{D})\* begin

yylval.yymyString:=yytext;

returni(id);

end;

\< returnc('<');

\> returnc('>');

\= returnc('=');

\, returnc(',');

" " ;

%%

end.

Для генерации транслятора будем использовать следующую грамматику SELECT-запроса:

SQL→ SELECT FROM WHERE

SQL→ SELECT FROM

SELECT→ select\_t FIELD\_LIST

FROM →from\_t id

WHERE→where\_t CONDITION\_LIST

FIELD\_LIST→FIELD\_LIST , id

FIELD\_LIST →id

CONDITION\_LIST→CONDITION\_LIST or\_t BOOLTERM

CONDITION\_LIST→ BOOLTERM

BOOLTERM → BOOLTERM and CONDITION

BOOLTERM → CONDITION

CONDITION → VALUE EOPR VALUE

EOPR→ <

EOPR→ >

EOPR → =

VALUE→ id

VALUE → num

Семантические правила такой грамматики, согласно заданию лабораторной работы, должны:

1) Проверить существования таблицы, которая указана в разделе FROM;

2) Проверить наличие указанных в запросе той таблицы, которая указана в разделе FROM;

3) Перевести запрос, указанный на языке SQL, к виду операций реляционной алгебры;

Дополнив исходную грамматику семантическими определениями, а также определив вспомогательные переменный для выполнения алгоритма трансляции, получим следящий файл Yacc:

%{ unit yacc;

interface

uses lexlib, yacclib, Classes, SysUtils, FileUtil, Forms, Controls, Graphics, Dialogs;

type

myString=string[255];

var

Table,Attributes: TStringList;

tableErr,attrErr,attrFoundFlag: boolean;

part,table\_str: string;

f:TextFile; i,j:integer;

%}

%token <myString> num

%token <myString> select\_t

%token <myString> from\_t

%token <myString> where\_t

%token <myString> or\_t

%token <myString> and\_t

%token <myString> id

%type <myString> sql

%type <myString> select

%type <myString> from

%type <myString> where

%type <myString> field\_list

%type <myString> condition\_list

%type <myString> boolterm

%type <myString> condition

%type <myString> value

%type <myString> eopr

%%

goal : sql {

attrErr:=false;

for i:=0 to Attributes.Count-1 do

begin

attrFoundFlag:=false;

for j:=1 to Table.Count-1 do

if Attributes[i] = Table[j] then

begin

attrFoundFlag:=true;

break;

end;

if not attrFoundFlag then

begin

attrErr:=true;

break;

end;

end;

if (not tableErr) and (not attrErr)

then showmessage('Успех! Операция реляционной алгебры: '+$1);

if tableErr then showmessage('Ошибка! Не найдена таблица в базе данных!');

if attrErr then showmessage('Ошибка! Не найдены указанные атрибуты таблицы!');

Table.Destroy;

Attributes.Destroy;

}

sql : select from where {$$:=$1+'('+$3+'('+$2+'))';}

| select from {$$:=$1+'('+$2+')';}

select : select\_t field\_list {$$:='π '+$2;}

from : from\_t id {

tableErr:=false;

$$:=$2;

Table:=TStringList.Create;

AssignFile(f,'D:\users\61-вт \db.txt');

Reset(f);

while not eof(f) do

begin

Readln(f,table\_str);

for i:=1 to Length(table\_str) do

if table\_str[i] = ' ' then

begin

Table.Add(part);

part:='';

end

else if i = Length(table\_str) then

begin

part:=part+table\_str[i];

Table.Add(part);

part:='';

end

else part:=part+table\_str[i];

if Table[0] = $2

then break

else Table.clear;

end;

CloseFile(f);

if Table.Count = 0 then tableErr:=true;

}

where : where\_t condition\_list {$$:='σ '+$2;}

field\_list : field\_list ',' id {

$$:=$1+','+$3;

Attributes.Add($3);

}

| id {

$$:=$1;

Attributes:=TStringList.Create;

Attributes.Add($1);

}

condition\_list : condition\_list or\_t boolterm {$$:=$1+' '+'∨'+' '+$3;}

| boolterm

boolterm : boolterm and\_t condition {$$:=$1+' '+'∧'+' '+$3;}

| condition

condition : value eopr value {

$$:=$1+' '+$2+' '+$3;

Attributes.Add($1);

Attributes.Add($3);

}

eopr : '<' {$$:='<';}

| '>' {$$:='>';}

| '=' {$$:='=';}

value : id

| num

%%

(\*$I SELECT\_L.pas\*)

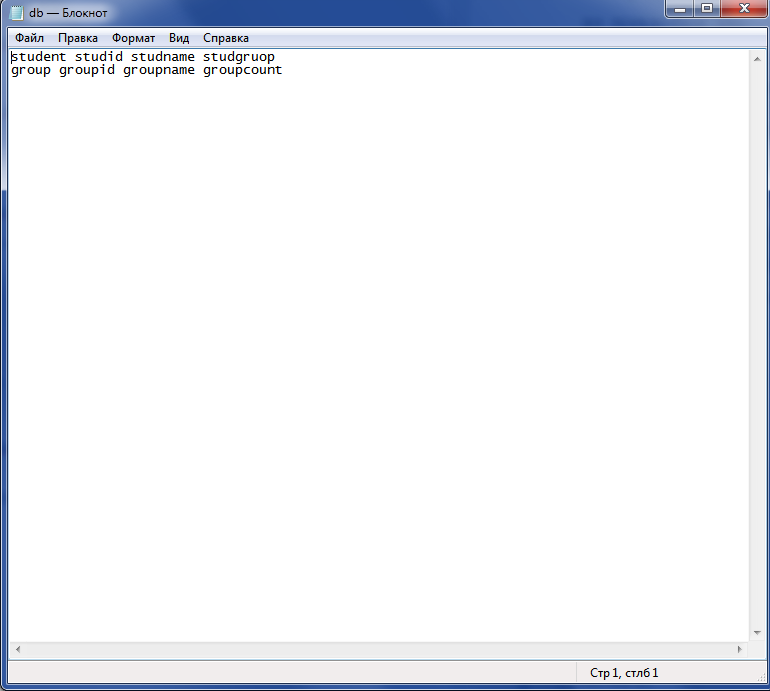
end.

В семантический дополнениях используется динамический массив строк TStringList, предназначенный для хранения тех атрибутов, которые были распознаны в ходе синтаксического и лексического анализатора. Сама же псевдо-база данных представлена в виде текстового файла, в котором информация о таблице указана в одной строке со следующей структурой:

<Таблица1> <Атрибут1> … <АтрибутN>

…

<ТаблицаN> <Атрибут1> … <АтрибутN>



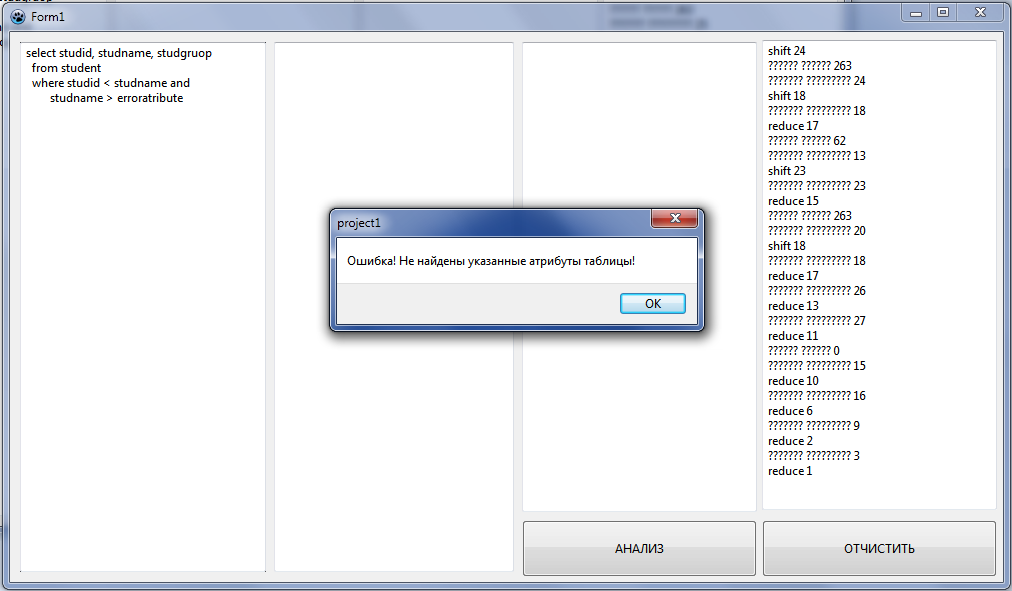
Изображение 1 – Файл базы данных

Имена таблиц и атрибутов разделены пробелами, чтобы можно было в экземпляр массива TStringList занести эти значения, и таким образом нулевой элемент будет соответствовать имени таблицы, а последующие элементы – именам атрибутов этой таблицы.

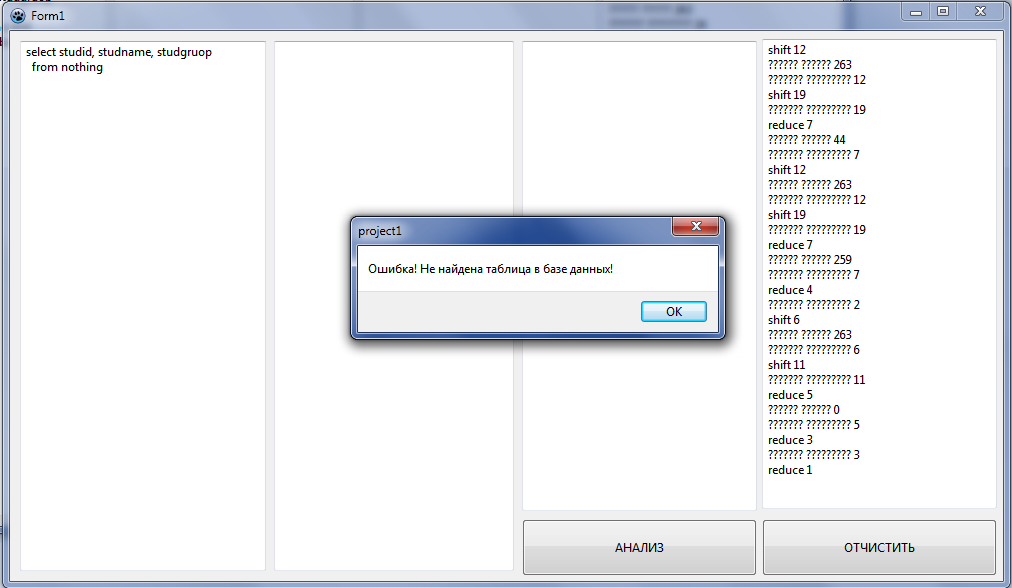
При свёртке в продукциях, где определяются атрибуты таблицы, используемые в запросе, в динамический массив атрибутов будут заноситься соответствующие значения. Таким образом, если все элементы этого массива будут соответствовать 1..N элементам массива таблицы, имя которой должно соответствует атрибуту при свёртке нетерминала FROM, то программа будет завершена успешно и выведет на экран соответствующую операцию реляционной алгебры.

Сама же операция формируется при свёртках нетерминалов SELECT и WHERE, где при SELECT в результат собирается оператор проекции π (Projection), а при WHERE в результат собирается оператор выборки σ (Selection).

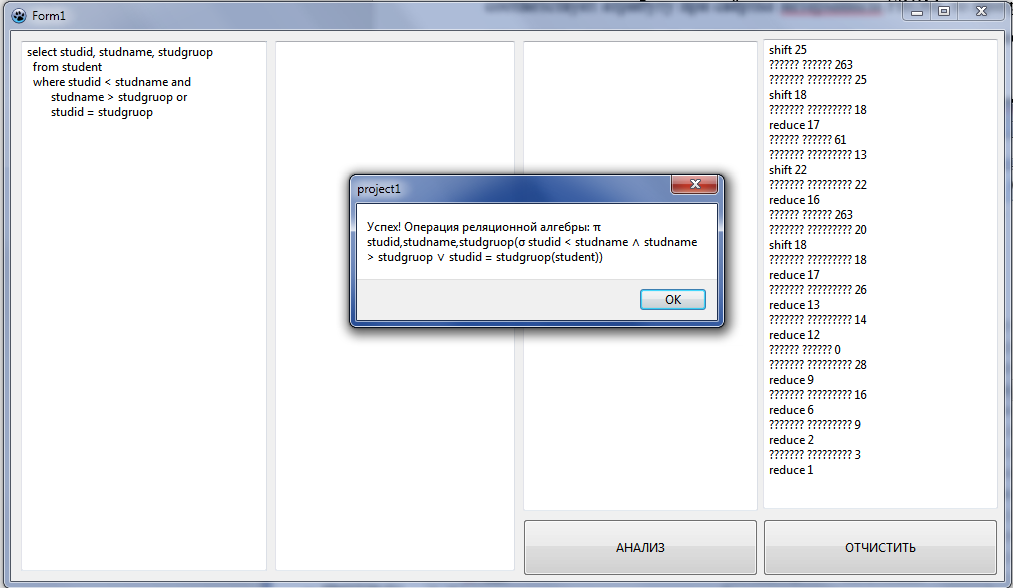
Следовательно, если в запросе указана неверная информация о таблице или её атрибутах, то транслятор выдаст соответствующие сообщение. Если же синтаксическая структура не нарушена, и все совпадения были найдены, то транслятор приведёт SQL-запрос к операции реляционной алгебры:



Изображение 2 – Несоответствующий атрибут таблицы



Изображение 3 – Несоответствующая таблица базы данных



Изображение 4 – Корректное завершение