

Facultatea de Automatica si Calculatoare

Catedra de Calculatoare

Parser Sql

Limbaje formale si translatoare

Student: Onetiu Alexandru

Grupa: 30238

Indrumator: Prof. Vanessa Mercea

1. Consideratii teoretice

Un **translator** reprezinta o unealta software utilizata pentru a face posibila convertirea unei bucati de cod scrisa intr-un anumit limbaj de programare intr-o bucata de cod care are aceeasi semnificatie, fiind insa implementata intr-un alt limbaj de programare.

Pentru realizarea acestei operatii, au fost utilizate Lex si Yacc. In continuare vom vorbi pe scurt despre cele 2.

* **Lex**

Lex este un program ce genereaza un analizator lexical sau scanner.

Lex foloseste un model care potriveste un string dat la intrare si il converteste din string in tokeni.

In figura de mai jos avem o prezentare generala a Lex:

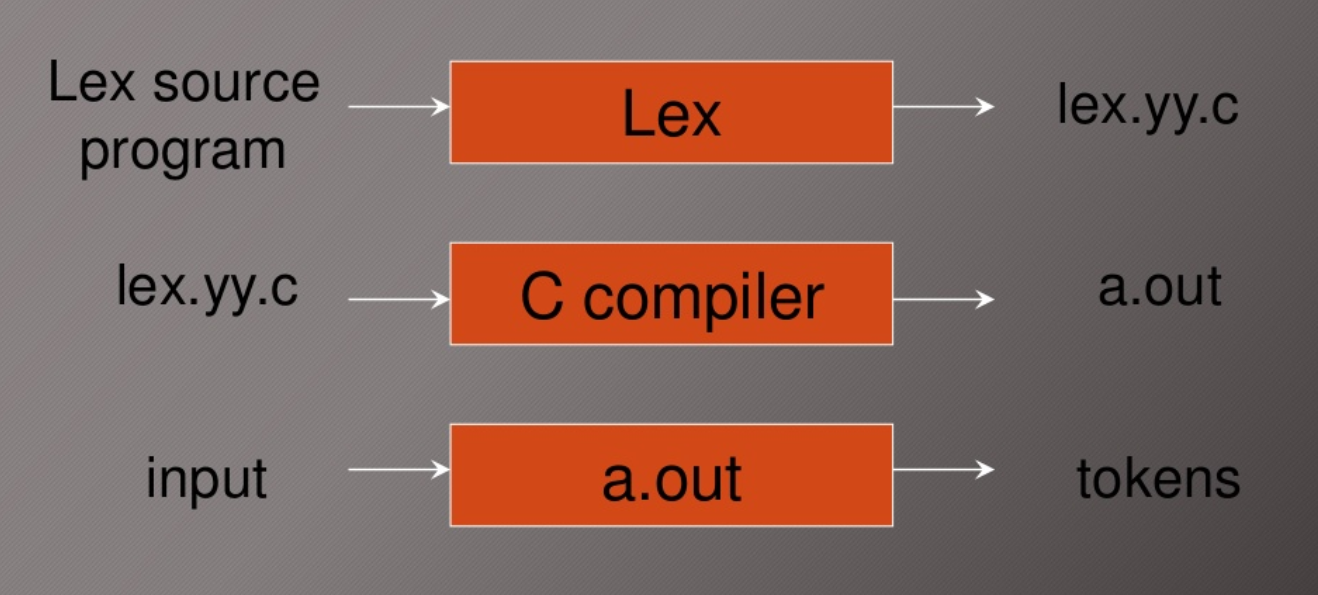


Figura 1. Lex Overview

* **Yacc**

Yacc genereza cod C pentru analizatorul de sintaxa sau parser.

Yacc foloseste reguli gramaticale care ii permit sa analizeze tokeni din Lex sis a formeze un arbore de sintaxa.

In figura de mai jos avem o prezentare generala a Yacc:

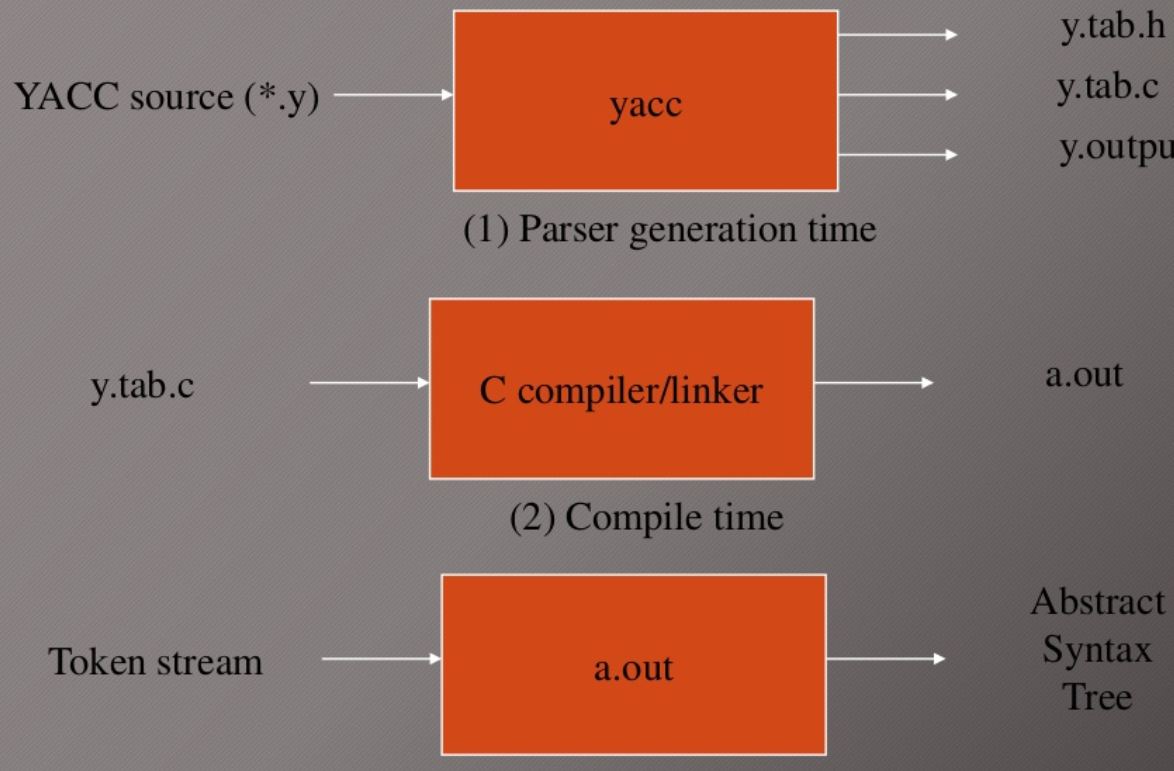


Figura 1. Yacc Overview

1. Parser Sql

Pentru inceput mi-am propus sa implementez urmatoarele operatii de baza:

* CREATE;
* SELECT;
* INSERT;
* DELETE;

1. CREATE – va fi operatia prin care se va putea creea un tabel.

Syntaxa: CREATE TABLE <NAME>(<ARG1> TYPE, <ARG2> TYPE, … <ARGn> TYPE);

TYPE reprezinta tipul si va putea fi int si varchar.

1. SELECT – va fi operatia prin care se va putea afisa continutul unui tabel.

Syntaxa: Vom avea 3 cazuri:

* SELECT \* FROM <TABLE\_NAME>;
* SELECT <COL1>, <COL2>, … FROM <TABLE\_NAME>;
* SELECT SELECT <COL1>, <COL2>, … FROM <TABLE\_NAME> WHERE <CONDITION>;

1. INSERT – va fi operatia prin care se vor putea adauga date noi in tabela.

Syntaxa: INSERT INTO <TABLE\_NAME> (COL1, COL2 … COLn) VALUES (<VAL1>, <VAL2> … <VALn>);

1. DELETE – va fi operatia de stergere a unei informatii din tabela.

Syntaxa: DELETE FROM <TABLE\_NAME> WHERE <CONDITION>.

In functie de evolutia in timp a proiectului ai mai vrea sa adaug operatia de UPDATE, ORDER BY etc.

Implementare si executie

Programul a fost implementat in c lex si yacc.

Pentru partea de memorare a datelor s-au folosit liste simplu inlantuite specifice:

typedef struct row\_node

{

char \*row\_name;

struct row\_node \*row\_next;

} row\_node;

typedef struct columns\_list

{

char \*columns\_name;

char \*columns\_data\_type;

bool primary\_key;

row\_node \*row\_data\_next;

struct columns\_list \*columns\_next;

} columns\_list;

typedef struct columns\_map

{

char \*map\_name;

columns\_list \*next\_column;

struct columns\_map \*columns\_map\_next;

} columns\_map;

typedef struct table\_node

{

char \*table\_name;

columns\_map \*columns\_data\_next;

struct table\_node \*table\_next;

} table\_node;

typedef struct database

{

table\_node \*table\_data\_next;

struct database \*db\_next;

} database;

Aplicatia se ruleaza folosind urmatoarele comenzi:

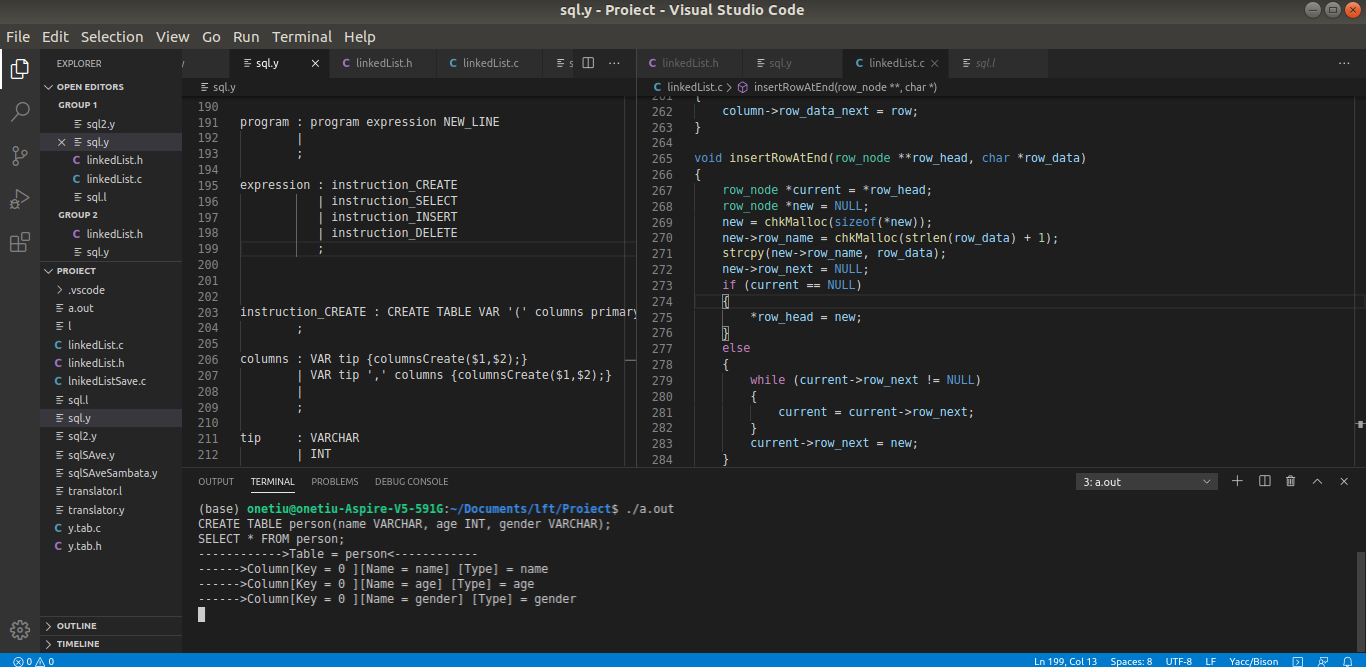
lex sql.l -l

yacc -d sql.y

cc lex.yy.c y.tab.c linkedList.c -ll -ly

./a.out

Reultate experimentale:



Proiectul poate fi dezvolatat pentru a suporta toate instructiunile mySql.

Bibliografie:

-materiale din laboratorul de LFT;

-Wikipedia;

-materiale din laboratorul de SDA;

-stackoverflow;