



Limbaje formale si translatoare

Translator sql

Tools:Lex and yacc

Name:Bogdan Andrei
Group: 30233
Email: andrei.bogdan01@gmail.com



Contents

1	Introduction	3
1.1	Scurta prezentare a proiectului	3
2	Implementare	4
2.1	Tokens	4
2.2	Yacc	5
3	Results, limits and possible improvements	6

Chapter 1

Introduction

1.1 Scurta prezentare a proiectului

- Scopul proiectului ii de a traduce din limbaj natural in sintaxa Sql. Acest proiect se bazeaza complet pe lex si yacc.
- Proiectul suporta instructiuni de tip SELECT, UPDATE ,DELETE, INSERT cu un vocabular de 27 de cuvinte cheie pentru sintaxa sql.
- link spre github <https://github.com/TheWolfOfFarron/SqlParser>

Chapter 2

Implementare

2.1 Tokens

- Toti tokeni folositi sunt descrisi mai jos si reprezinta tipuri de sintaxe sql.

- SELECT printf("SELECT "); return SELECT;
- selecteaza printf("SELECT "); return SELECT;
- afiseaza printf("SELECT "); return SELECT;
- UPDATE printf("UPDATE "); return UPDATE;
- actualizeaza printf("UPDATE "); return UPDATE;
- DELETE printf("DELETE "); return DELETE;
- sterge printf("DELETE "); return DELETE;
- INSERT printf("INSERT "); return INSERT;
- adauga printf("INSERT "); return INSERT;
- insereaza printf("INSERT "); return INSERT;
- FROM printf("FROM "); return FROM;
- din printf("FROM "); return FROM;
- in printf("FROM "); return FROM;
- WHERE printf("WHERE "); return WHERE;
- unde printf("WHERE "); return WHERE;
- tot printf("* "); yylval.strval = "*"; return STRING;
- INTO printf("INTO "); return INTO;
- in printf("INTO "); return INTO;
- AND printf("AND "); return AND;
- si printf("AND "); return AND;

- OR printf("OR "); return OR;
- sau printf("OR "); return OR;
- SET printf("SET "); return SET;
- seteaza printf("SET "); return SET;
- VALUES printf("VALUES "); return VALUES;
- valori printf("VALUES "); return VALUES;

2.2 Yacc

- Yacc syntax:

```

select-query: SELECT columns FROM table WHERE conditions { $$ = opr3('s',$
conditions: condition { $$ = opr2('C',$1,NULL); }
| condition AND conditions { $$ = opr2('a',$1,$3); }
| condition '>' conditions { $$ = opr2('>',$1,$3); }
| condition '<' conditions { $$ = opr2('<',$1,$3); }
| condition '=' conditions { $$ = opr2('=',$1,$3); }
| condition OR conditions { $$ = opr2('o',$1,$3); }

;

```

- Pentru select am decis ca orice propozitie de tip select incepe cu un token select , coloane tabel si conditi daca exista. De ex "afiseaza tot din anagati unde city = Cluj"
- Pentru conditi am luat in considerare operatiile de baza pentru a putea crea o expresie sql mai complexa.
- Programul nu suporta expresii sql mai complexe precum count(...) sau joins deoarece nu am putut gasi o varianta generala care sa suporte aceste operatii.

Chapter 3

Results, limits and possible improvements

```
interpreter.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
```

Open

test

~/Desktop/lft/interpreter_C

```
1 sterge din employees unde age > '30' AND lol < '32' AND lol < '32'
```

```
2
```

Plain Te

```
● ~/Desktop/lft/interpreter_C$ ./out < test
  sergeParser error: syntax error
● wolf@wolf:~/Desktop/lft/interpreter_C$ ./out < test
  DELETE nameParser error: syntax error
● wolf@wolf:~/Desktop/lft/interpreter_C$ ./out < test
  DELETE FROM employeesWHERE age>'30'AND lol<'32'AND lol<'32'

  ( D ( t employees ) ( > ( C age ) ( a ( C '30' ) ( < ( C lol ) ( a ( C '32' ) ( < ( C lol )
○ /interpreter_C$
```

Figure 3.1: Results

```

SELECT nameFROM employeesWHERE age>'30'AND lol<'32'AND lol<'32'

```

Open

test

~/Desktop/lft/interpreter_C

```

1 afiseaza tot din a
2

```

Plain Te

```

DELETE nameParser error: syntax error
• wolf@wolf:~/Desktop/lft/interpreter_C$ ./out < test
DELETE FROM employeesWHERE age>'30'AND lol<'32'AND lol<'32'

( D ( t employees ) ( > ( C age ) ( a ( C '30' ) ( < ( C lol ) ( a ( C '32' ) ( < ( C lol )
• /interpreter_C$ ./out < test
SELECT * FROM a

( s ( c * ) ( t a ) ) wolf@wolf:~/Desktop/lft/interpreter_C$

```

Figure 3.2: Results

```

command 'mark' from deb mmh (0.4-4)

```

Open

*test

~/Desktop/lft/interpreter_C

```

1 afiseaza name FROM employees WHERE age > '30' AND lol < '32' AND lol < '32'
2

```

Plain Te

```

yacc -d interpreter.y
interpreter.y: warning: 2 shift/reduce conflicts [-Wconflicts-sr]
interpreter.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
gcc -o out lex.yy.c y.tab.c -ly -ll
• wolf@wolf:~/Desktop/lft/interpreter_C$ ./out < test
SELECT nameFROM employeesWHERE age>'30'AND lol<'32'AND lol<'32'

( s ( c name ) ( t employees ) ( > ( C age ) ( a ( C '30' ) ( < ( C lol ) ( a ( C '32' ) ( <
• ~/Desktop/lft/interpreter_C$

```

Figure 3.3: Results

```

command 'mark' from deb mmh (0.4-4)

```

Open

*test

~/Desktop/lft/interpreter_C

```

1 serge name din employees unde age > '30' AND lol < '32' AND lol < '32'
2

```

Plain Te

```

yacc -d interpreter.y
interpreter.y: warning: 2 shift/reduce conflicts [-Wconflicts-sr]
interpreter.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
gcc -o out lex.yy.c y.tab.c -ly -ll
• wolf@wolf:~/Desktop/lft/interpreter_C$ ./out < test
SELECT nameFROM employeesWHERE age>'30'AND lol<'32'AND lol<'32'

( s ( c name ) ( t employees ) ( > ( C age ) ( a ( C '30' ) ( < ( C lol ) ( a ( C '32' ) ( <
• ~/Desktop/lft/interpreter_C$

```

Figure 3.4: Results

```

( s ( c name ) ( t employees ) ( > ( C age ) ( a ( C '30' ) ( < ( C lol ) ( a ( C '32' ) ( <
• Desktop/lft/interpreter_C$ ./out < test
Open test
~/Desktop/lft/interpreter_C
1 afiseaza tot din a
2
Plain Te

( D ( t employees ) ( > ( C age ) ( a ( C '30' ) ( < ( C lol ) ( a ( C '32' ) ( < ( C lol )
• nterpreter_C$ ./out < test
SELECT * FROM a
• wolf@wolf:~/Desktop/lft/interpreter_C$ ./out < test
SELECT * FROM a
○ ( s ( c * ) ( t a ) ) wolf@wolf:~/Desktop/lft/interpreter_C$

```

Figure 3.5: Results

```

interpreter.y:22:22: note: expected 'char *' but argument is of type 'struct _node *'
22 | node *setConst(char* value);
Open test
~/Desktop/lft/interpreter_C
1 actualizeaza employees seteaza salary = 5000 unde department = 'HR'
Plain Te

interpreter.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
gcc -o out lex.yy.c y.tab.c -ly -ll
• wolf@wolf:~/Desktop/lft/interpreter_C$ ./out < test
INSERT INTO productsVALUES ('1001',Parser error: syntax error
• wolf@wolf:~/Desktop/lft/interpreter_C$ ./out < test
UPDATE employeesSET salary=5000WHERE department='HR'
• ( U ( t employees ) ( u ( = salary salary ) ) ( = ( C department ) ( C ( C 'HR' ) ) ) ) wolf@w
UPDATE employeesSET salary=5000WHERE department='HR'
○ ( U ( t employees ) ( u ( = salary salary ) ) ( = ( C department ) ( C ( C 'HR' ) ) ) ) wolf@w

```

Figure 3.6: Results

```

|
interpreter.y:22:22: note: expected 'char *' but argument is of type 'struct _node *'
22 | node *setConst(char* value);
| ~~~~~^~~~~
• wolf@wolf:~/Desktop/lft/interpreter_C$ make
lex interpreter.l
yacc -d interpreter.y
interpreter.y: warning: 3 shift/reduce conflicts [-Wconflicts-sr]
interpreter.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
gcc -o out lex.yy.c y.tab.c -ly -ll
• wolf@wolf:~/Desktop/lft/interpreter_C$ ./out < test
INSERT INTO productsVALUES ('1001',Parser error: syntax error
• wolf@wolf:~/Desktop/lft/interpreter_C$ ./out < test
UPDATE employeesSET salary=5000WHERE department='HR'
○ ( U ( t employees ) ( u ( = salary salary ) ) ( = ( C department ) ( C ( C 'HR' ) ) ) ) wolf@w

```

Figure 3.7: Results