CSB 353: Compiler Design

Project Report (Parser for P3SQL)

Submitted By:

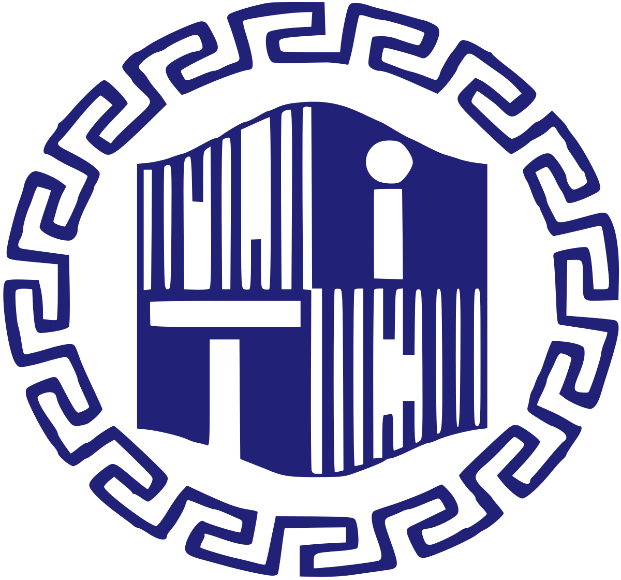
Prashant Borkar (191210036)

Prem Kumar (191210037)

Prince Kumar (191210038)

Submitted To: Dr. Shelly Sachdeva

Department of Computer Science and Engineering



NATIONAL INSTITUTE OF TECHNOLOGY DELHI

2019-2023

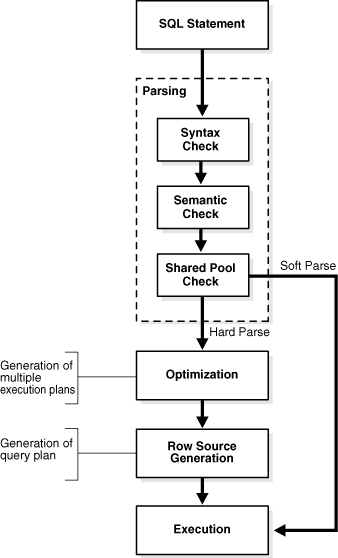
Table of Content

|  |  |  |
| --- | --- | --- |
| S No. | Content | Page No |
| 1 | Introduction | 3 |
| 2 | Lexical Analyzer | 5 |
| 3 | Syntax Analyzer | 7 |
| 4 | Semantic Analyzer | 9 |
| 5 | Execution | 10 |
| 6 | References | 11 |

Introduction

Our project aims to undertake a sequence of experiments to design and implement various phases of a Parser for a Structured Query language which we have named as P3SQL, it is a subset of SQL.

We have developed this in python making the use of lex and yacc and we use pymysql to connect with MySQL Database.



Covered Scope:

* **Type**
  + STRING
  + INTEGER
* **Statement**
  + CREATE DATABASE
  + DROP DATABASE
  + SHOW DATABASES
  + USE DATABASE
  + CREATE TABLE
  + DROP TABLE
  + SHOW TABLES
  + INSERT
  + DELETE
  + UPDATE
  + SELECT

Inputs can be given through the input file or through terminal.

Requirements:

* Python 3
* PLY

PLY is yet another implementation of lex and yacc for Python. Some notable features include the fact that it is implemented entirely in Python and it uses LALR(1) parsing which is efficient and well suited for larger grammars.

* PyMySQL

This package contains a pure-Python MySQL client library.

* MySQL
* VS Code(ide)

Phases:

* Lexical Analyzer:

Identification of Keywords, Identifiers, Operators (Relational, Logical and Arithmetic), Punctuators, Constants (Integer, Character) and String Literals with invalid string error handling. Parenthesis matching with error reporting.

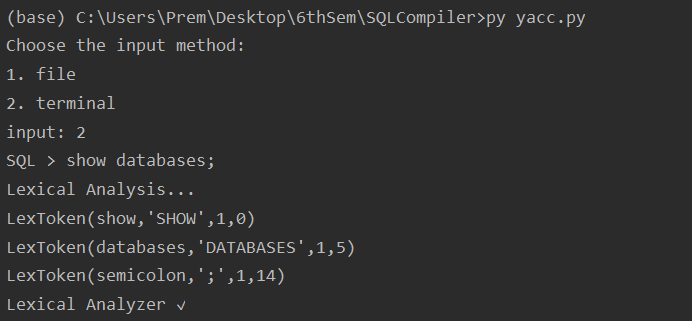
Following are the valid tokens:

tokens = [  
 'create',  
 'use',  
 'show',  
 'insert',  
 'select',  
 'update',  
 'delete',  
 'drop',  
 'exit',  
 'databases',  
 'database',  
 'tables',  
 'table',  
 'into',  
 'values',  
 'from',  
 'all',  
 'where',  
 'set',  
 'compare',  
 'logic',  
 'char',  
 'int',  
 'id',  
 'number',  
 'string',  
 'comma',  
 'semicolon',  
 'left\_paren',  
 'right\_paren'  
]

And tokens are defined as follows using regular expression -

def t\_create(t):  
 *r"""(C|c)(R|r)(E|e)(A|a)(T|t)(E|e)"""* t.value = "CREATE"  
 return t

Output for Lexical Analyzer



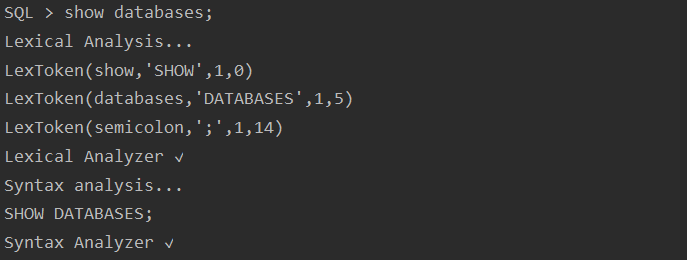
* Syntax Analyzer:

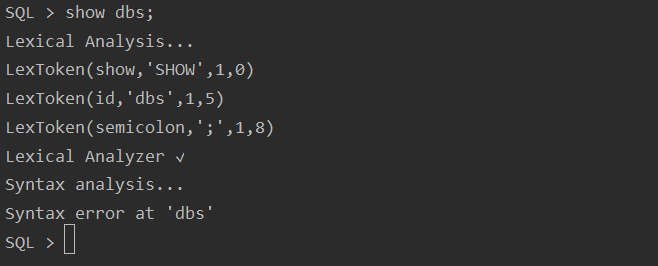
In this phase it receives the inputs, in the form of tokens, from lexical analyzers. Checks if the expressions from these tokens are syntactically correct or not.

Production rules are as follows -

Rule 0 S' -> start  
Rule 1 start -> statement  
Rule 2 statement -> <empty>  
Rule 3 statement -> create\_db statement  
Rule 4 statement -> show\_db statement  
Rule 5 statement -> drop\_db statement  
Rule 6 statement -> use\_db statement  
Rule 7 statement -> create\_tb statement  
Rule 8 statement -> show\_tb statement  
Rule 9 statement -> drop\_tb statement  
Rule 10 statement -> insert\_tb statement  
Rule 11 statement -> delete\_tb statement  
Rule 12 statement -> update\_tb statement  
Rule 13 statement -> select\_tb statement  
Rule 14 statement -> exit\_db statement  
Rule 15 create\_db -> create database id semicolon  
Rule 16 show\_db -> show databases semicolon  
Rule 17 drop\_db -> drop database id semicolon  
Rule 18 use\_db -> use id semicolon  
Rule 19 create\_tb -> create table id left\_paren cols right\_paren semicolon   
Rule 20 cols -> id type col  
Rule 21 type -> int  
Rule 22 type -> char left\_paren number right\_paren  
Rule 23 col -> <empty>  
Rule 24 col -> comma id type col  
Rule 25 show\_tb -> show tables semicolon  
Rule 26 drop\_tb -> drop table id semicolon  
Rule 27 insert\_tb -> insert into tb\_name values left\_paren value\_cols right\_paren semicolon  
Rule 28 tb\_name -> id  
Rule 29 tb\_name -> id left\_paren id\_cols right\_paren  
Rule 30 id\_cols -> id id\_col  
Rule 31 id\_col -> <empty>  
Rule 32 id\_col -> comma id id\_col  
Rule 33 value\_cols -> string value\_col  
Rule 34 value\_cols -> number value\_col  
Rule 35 value\_col -> <empty>  
Rule 36 value\_col -> comma string value\_col  
Rule 37 value\_col -> comma number value\_col  
Rule 38 delete\_tb -> delete from id where conditions semicolon  
Rule 39 conditions -> condition\_col  
Rule 40 conditions -> left\_paren conditions right\_paren condition  
Rule 41 condition\_col -> id compare id  
Rule 42 condition\_col -> id compare number  
Rule 43 condition\_col -> id compare string  
Rule 44 condition\_col -> number compare id  
Rule 45 condition\_col -> number compare string  
Rule 46 condition\_col -> string compare id  
Rule 47 condition\_col -> string compare string  
Rule 48 condition -> <empty>  
Rule 49 condition -> logic left\_paren condition\_col right\_paren condition  
Rule 50 update\_tb -> update id set update\_cols semicolon  
Rule 51 update\_tb -> update id set update\_cols where conditions semicolon  
Rule 52 update\_cols -> id compare number update\_col  
Rule 53 update\_cols -> id compare string update\_col  
Rule 54 update\_col -> <empty>  
Rule 55 update\_col -> comma id compare number update\_col  
Rule 56 update\_col -> comma id compare string update\_col  
Rule 57 select\_tb -> select all from id\_cols semicolon  
Rule 58 select\_tb -> select id\_cols from id\_cols semicolon  
Rule 59 select\_tb -> select all from id\_cols where conditions semicolon  
Rule 60 select\_tb -> select id\_cols from id\_cols where conditions semicolon  
Rule 61 exit\_db -> exit semicolon

Output for Syntax Analyzer





* Semantic Analyzer:

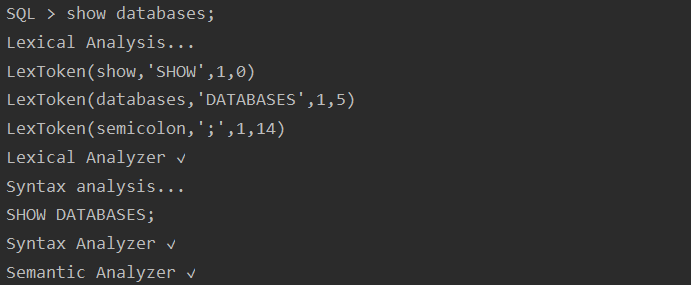
In this phase, we extract necessary semantic information from the P3SQL queries.

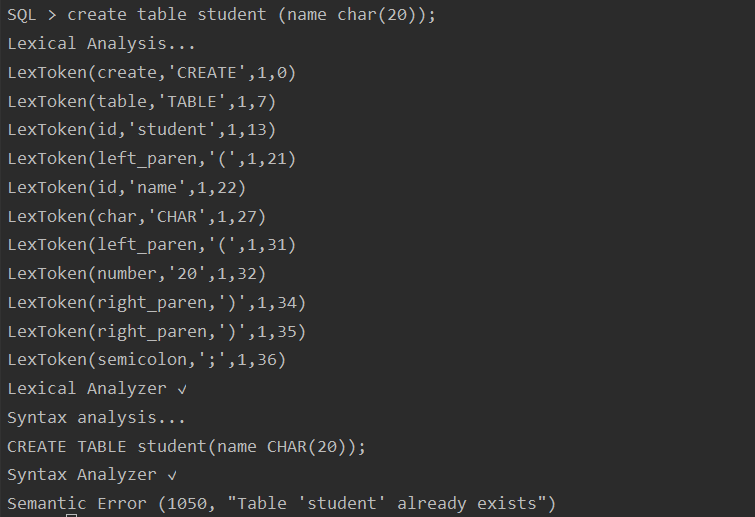
We are validating the following things –

Validation of Databases and Tables if already exists or not.

Validating values for insert and update queries.

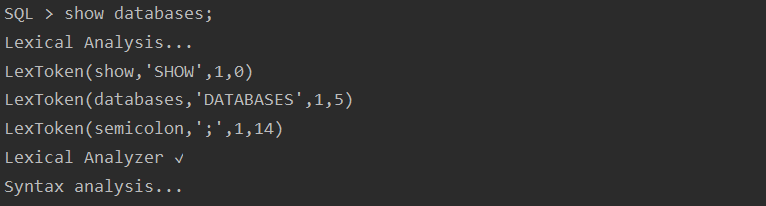
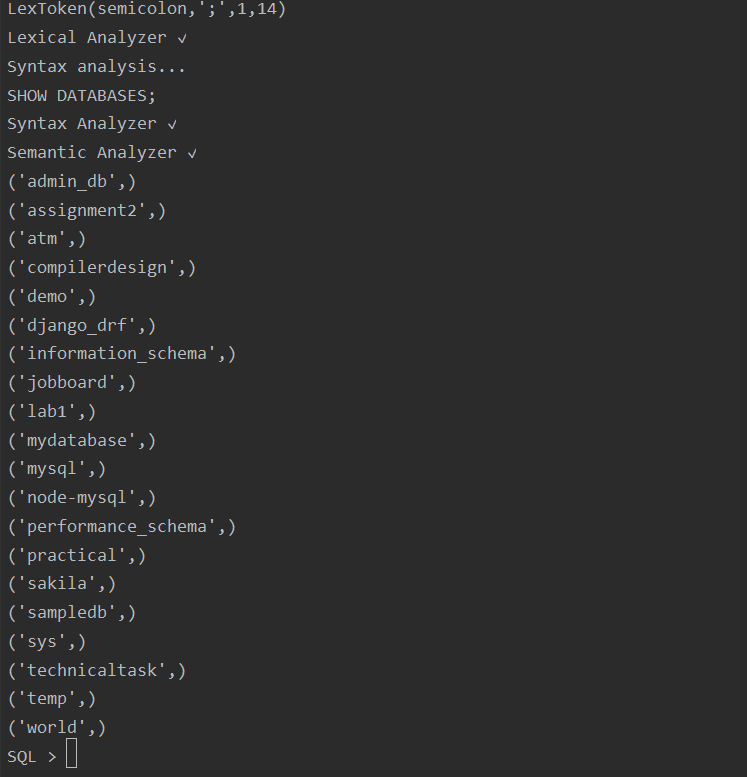
Output for Semantic Analyzer





Execution:

Execution of P3SQL Query:

References

<https://www.dabeaz.com/ply/ply.html>

<https://ply.readthedocs.io/en/latest/>