COMPILER DESIGN PROJECT REPORT

- *Parser for create and drop sql commands* -

MCA II Year

Submitted by :
- Bhavna - Deepti - Himanshi -

SQL

SQL is a special purpose programming for relational databases. Rather than manipulating data in memory, it manipulates data in database tables.

1. DROP COMMAND

We use the SQL DROP table command to drop a table from the database. It completely removes the table structure and associated indexes, statistics, permissions and constriants. DROP database command to drop database.

In this project we have created a simple parser for SQL language - drop command with limited cases and tokens supported.

SYNTAX

Using lex/yacc implement a parser for the select SQL statement.
The syntax of the command is

```
drop table table_name;

drop database db_name;
```

If you want to drop multiple tables/databases using a single statement, you can use a comma separated list of tables/database names after the DROP statement.

```
drop table table_1, table_2,... table_n; - drop multiple tables.

drop database db_1, db_2,... db_n; - drop multiple databases.
```

Here drop, table and database are our tokens and table_name & db_name have a usual structure of identifiers(alphanumeric strings beginning with a letter).

COMPILE & RUN

```
flex drop.l
bison -vyd drop.y
gcc lex.yy.c y.tab.c
```

ASSUMPTION

1. SQL is case insensitive.
2. Table & Database name should not start with '_' & 0-9 but can be contained in between.
3. Multiple table and database drop using a comma separated list.
4. Semicolon ' ; ' to end the statement.

RESULTS

1. Test case : case insensitive.

```
C:\Users\hp\Desktop\CD_assignment\COMPILER_PROJECT\drop>a
Enter expression : DROP TABLE emp;
Syntax Correct
```

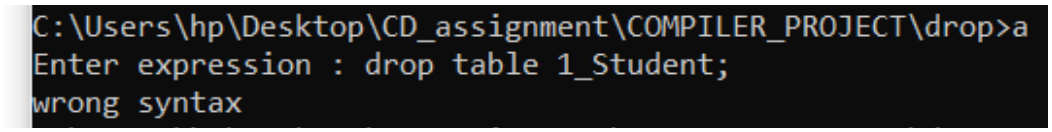
```
C:\Users\hp\Desktop\CD_assignment\COMPILER_PROJECT\drop>a
Enter expression : DROP DATABASE EMP;
Syntax Correct
```

2. Test case : Multiple table/database drop.

```
C:\Users\hp\Desktop\CD_assignment\COMPILER_PROJECT\drop>a
Enter expression : drop table emp1,emp2;
Syntax Correct
```

```
C:\Users\hp\Desktop\CD_assignment\COMPILER_PROJECT\drop>a
Enter expression : drop database db_1,db_2;
Syntax Correct
```

3. Test case : wrong naming conventions will lead to wrong syntax.



CONCLUSION

Drop command parser has been created successfully.

2. CREATE COMMAND

We use the SQL CREATE table/database command to create a table/database.It creates the table with given name and columns along with data types and constraints provided with columns.it also creates a database with given name. In this project we have created a simple parser for SQL language create command with limited cases and tokens supported.

SYNTAX

Using lex/yacc implement a parser for the create SQL statement.
The syntax of the command is

```
create table table_name (col1 data_type constraints,...,coln data_type constraints);

create database db_name;
```

Here create, table and database are our tokens and table_name,column name & db_name have a usual structure of identifiers(alphanumeric strings beginning with a letter).

ASSUMPTIONS

There are few assumptions that we have made while creating tables and databases.these are as follows:

- Table name,Column name,database name can not be "_".
- Rules like "NOT USING KEYWORDS TO NAME TABLES,COLUMNS AND DATABASES" are not assured.
- Few constraints are provided such as unique,primary key and not null only.
- Data types provided are char,varchar,int,smallint,bigint,float,date and bool.
- Primary key Constraint can not be defined separately at the end of create command
- Semicolon is Compulsory.
- More than one constraint is not allowed on a column in a table (exception when one of the constraints is primary key,it accepts more than one constraints).
- Correct range of data types like varchar and char is not assured.

COMPILE & RUN

```
flex drop.l
bison -vyd drop.
gcc lex.yy.c y.tab.c
```

RESULTS

Test Case1: Case insensitive,to show that our parser is not case sensitive regarding sql keywords.

```
CReaTe TaBle CustOmer (Cust_id InT PrimarY KeY,NaMe Char(20) NOT NULL,BirthDate DaTe,SalAry BigInt);
Correct syntax
```

Test Case2: Creating a Database Library

```
create database Library;
Correct syntax
```

Test Case3: Creating a table Customer with columns with their associated data types and constraints.

```
create table customer (cust_id int primary key,name char(30) not null,birthdate date,salary bigint);
Correct syntax
```

Test Case4:Won't accept name of columns,table and database starting with number.

```
create database 1Library;
Syntax Error
```

Test Case5: In Accordance to the assumption made it accepts keywords as names.

```
create table customer (int int primary key,name char(30) not null,birthdate date,salary bigint);
Correct syntax
```

Test Case6:Primary key check is done by parser to make sure only one primary key is allowed per table.

```
create table customer (cust_id int primary key,name char(30) not null primary key,birthdate date,salary bigint);
Only 1 primary key is allowed!
Syntax Error
```

CONCLUSION

That's how we can create an sql parser to parse the create command . Similarly we can create a parser for other sql commands.