Project 1-1: SQL Parsing

General

In this assignment, you will build a simple version of SQL parser that will be used in Project 1-2 and Project 1-3. The SQL Parser is responsible for understanding and analyzing the structure of SQL Statements.

1. Requirements

- Implement a simple SQL parser which
  - Parses SQL statements listed in section 2. If an error occurs, show an error message. Otherwise, in case of success, show a success message.
  - Parses SQL statements in multiple lines, assuming that every query ends with a semi-colon ';', not the line feed'₩n' or carriage return '₩r'
  - Use standard input/output
- In addition, start your parser's command line with "SQL_StudentID>"
  EX) SQL_2007-23593> CREATE TABLE ...

2. SQL

2.1 CREATE TABLE

```
CREATE TABLE table_name
(
        column_name data_type [PRIMARY KEY] [NOT NULL],
        …
);
```

- *data_type*

Type	Constraints	Example
INT		id = 123
CHAR (*length*)	Length > 0	name = 'Sangkeun Lee'
DATE	'YYYY-MM-DD'	reg_date = '2004-08-04'

- *table_name, column_name*
  - Ignore case *(Ex. Apple = APPLE)*
  - Only alphabets and'_'are allowed

- EXAMPLE

```
SQL_2007-23593>
CREATE TABLE dbta_test
(
        id INTEGER PRIMARY KEY NOT NULL,
        name CHAR(10)
);
SQL_2007-23593> “CREATE TABLE” requested!!
```

2.2 DROP TABLE

```
DROP TABLE table_name;
```

- EXAMPLE

```
SQL_2007-23593>
DROP TABLE dbta_test;
SQL_2007-23593> “DROP TABLE” requested!!
```

2.3 DESC

```
DESC table_name;
```

- EXAMPLE

```
SQL_2007-23593>
DESC dbta_test;
SQL_2007-23593> “DESC” requested!!
```

2.4 SHOWTABLES

```
SHOWTABLES;
```

- EXAMPLE

```
SQL_2007-23593>
SHOWTABLES;
SQL_2007-23593> “SHOWTABLES” requested!!
```

2.5 INSERT

```
INSERT INTO table_name (column_list) VALUES (value_list);
```

- EXAMPLE

```
SQL_2007-23593>
INSERT INTO dbta_test (id, name) VALUES (1,  'SangKeun Lee' );
SQL_2007-23593>  "INSERT"  requested!!
SQL_2007-23593>
INSERT INTO dbta_test (id, name) VALUE (2);
SQL_2007-23593>  "INSERT"  error!!
```

2.6 UPDATE

```
UPDATE table_name
SET column_name = value, …
[WHERE column_name operator value [AND …]];
```

- Operators
  - <, >, =, !=, <=, >=
  - IS NULL, IS NOT NULL
- If there is no where clause in the SQL statement, update all records in the table
- 'AND' operation has to be implemented
- Extra credit, if implementing 'OR' operation
- EXAMPLE

```
SQL_2007-23593>
UPDATE dbta_test
SET type =  'ta'
WHERE id = 1 AND name =  'SangKeun Lee' ;
SQL_2007-23593>  "UPDATE"  requested!!
```

2.7 DELETE

```
DELETE FROM table_name
[ WHERE column_name operator value [AND …]];
```

- Operators
  - <, >, =, !=, <=, >=
  - IS NULL, IS NOT NULL
- If there is no where clause in the SQL statement, delete all records in the table
- 'AND' operation has to be implemented
- Extra credit, if implementing 'OR' operation
- EXAMPLE

```
SQL_2007-23593>
DELETE FROM dbta_test
WHERE name IS NULL;
SQL_2007-23593>  "DELETE"  requested!!
```

2.8 SELECT

```
SELECT [table_name.]column_name [AS name], …
FROM table_name [AS name], …
[WHERE [table_name.]column_name operator value|[table_name.]column_name AND …]
```

- Select statements can contain a wildcard character (`*')
- If there is no where clause in the SQL statement, select all records in the table
- Extra credit, if implementing rename operator 'AS'
- EXAMPLE

```
SQL_2007-23593>
SELECT *
FROM dbta_test
SQL_2004-21595> "SELECT" requested!!

SQL_2007-23593>
SELECT id, name, type
FROM dbta_test
WHERE name = 'Sangkeun Lee' AND reg_date = '2006-09-14';
SQL_2007-23593> "SELECT" requested!!

SQL_2007-23593>
SELECT dbta_test.id, name, type
FROM dbta_test, dbta_term
WHERE dbta_test.reg_date = dbta_term.reg_date;
SQL_2007-23593> "SELECT" requested!!
```

3. Development Environment

- 3 Programming languages are allowed (C , C++, Java)
- LINUX or Windows
- API : lex&yacc, JAVACC&JAX (for java), …
  - You can use any kind of API, but it should be specified in your report

4. Submit

Files to submit

- Source files (must have comments), Binary files, Makefile (or bat),
- A Report (this file contains the following)
  a) Development environment

b) Explanation on the major modules and algorithms
c) What you have implemented and what you have not (Specify in detail)
d) Brief explanation of your implementation (less than half a page)
e) Any assumptions you have made
f) How to compile and run
g) Talk about your experience of implementing a simple SQL Parser

- Please submit the files in .zip format with the filename corresponding to your student id.(e.g **PRJ1-1_StudentID.zip**) via email to the teaching assistant (liza183@europa.snu.ac.kr)
  - Email Title : [introDB Project1-1] Your Student ID, Your Name
- Please submit the hard copy of your report to Building# 302 Room# 314-1
- Due Date: March 21st, 2008 23:59(Fri)

5. Late Assignment Policy

Programming assignments are due at 11:59pm on the date specified. A grading penalty will be applied to late assignments. (10% penalty up to the first 24 hours, 20% for 24 to 48 hours, with no credit received after that)

6. Reference

- lex&yacc
  - http://dinosaur.compilertools.net/
  - http://kldp.org/KoreanDoc/html/Lex_Yacc-KLDP/Lex_Yacc-KLDP-1.html
  - http://cui.unige.ch/dbresearch/Enseignement/analyseinfo/AboutBNF.html
  - http://gnosis.cx/publish/programming/regular_expressions.html

- Jax&javacc
  - http://www.engr.mun.ca/~theo/JavaCC-Tutorial/javacc-tutorial.pdf
  - http://linux4u.jinr.ru/usoft/WWW/www_blackdown.org/kbs/jax.html