# CSE 562 Database Systems Project

# SQL Query Evaluator Implementation

(Not finalized yet)

## Introduction

By June 6, 2017, you are all expected to be familiar with how to write SQL queries. In this project, you are required to implement a SQL query evaluator. This system will take a text file that include SQL queries as input, and produce query results from a group of comma separated files that includes data.

There are two different query types you need to support: (1) CREATE TABLE, and (2) SELECT statements.

Your code will be evaluated based on its correctness; which means the query result your queries return will be only indication that your code runs correctly. However, there is a catch: there will be a memory limit. If your program cannot run a script with 512 MB or less memory, you won't receive any points from that query.

The teams will consist of 3 people. If you cannot find a team, put an ad on the piazza group. If you want to have a team of 2 people, get my approval.

## Step 1: Parsing SQL queries

The parser converts the given SQL query string into a structured statement that you are going to evaluate. You can use JSQLParser, an open-source SQL parser, or you can implement your own (not recommended). You can download the parser from

http://maven.mimirdb.info/info/mimirdb/jsqlparser/1.0.0/jsqlparser-1.0.0.jar

And you can find the documentation of the parser from

http://doc.odin.cse.buffalo.edu/jsqlparser/

JSQLParser couples with an Expression library which helps to evaluate the expressions within the query. If you decide to implement your own parser, keep in mind that you cannot use the expression library. You can download the Expression library from

http://maven.mimirdb.info/info/mimirdb/evallib/1.0/evallib-1.0.jar

JSQLParser takes a stream object (java.io.Reader or java.io.InputStream) as the input and from which the file data to be parsed (java.io.FileReader). Let's assume you've created one already (of either type) and called it inputFile. Here is an example of how to start using the parser:

```
CCJSqlParser parser = new CCJSqlParser(inputFile);
```

```
Statement statement;
while((statement = parser.Statement()) != null){
  if(statement instanceof Select) {
      Select selectStatement = (Select)statement;
      // handle the select
      } else if(statement instanceof CreateTable) {
      // and so forth
   }
}
// End-of-file.  Exit!
```

Please see Oliver Kennedy's video on how to user JSQLParser:
https://youtu.be/U4TyaHTJ3Zg

To use the Expression library, specifically the Eval class, you will need to define a method for dereferencing Column objects. For example, if we have a HashMap that contains the schema of the tuple currently being evaluated, we can use that schema to identify which column we need to use from the tuple with the following method (you may need to change this according to your documentation):

```
public LeafValue eval(Column x){
  int colID = tupleSchema.get(x.getName());
  return tuple.get(colID);
}
```

### Step 2: CreateTable statements

You will be provided with an example set of query files, and the corresponding data files. The data files will exactly match the CreateTable statements. For example, if you receive a CreateTable statement as follows:

```
CREATE TABLE R(A string, B int, C date);
```

You are guaranteed to be given a file named R.dat with three columns; first column will be a string, second column will be an integer, and the third column will be a date object. The query file and the data file are guaranteed to be in the same folder.

Each line of text corresponds to one row of data. Each record is delimited by a vertical pipe '|' character. Dates are stored in YYYY-MM-DD form, where YYYY is the 4-digit year, MM is the 2-digit month number, and DD is the 2-digit date.

### Step 3: Select statements

Unfortunately, even if you successfully read and evaluate CreateTable statements, you won't be

graded from them. You will be graded from the correctness of select query results. We will provide you with the SELECT queries that you will need to evaluate. For every SELECT query, you will be expected to output seven different attribute types, and the query results. These attribute types are (1) projections, (2) tables used, (3) selections, (5) joins, (6) group-by, and (7) order-by. For example:

```
SELECT R.*
 FROM R, S
WHERE R.A = "something"
 AND R.B > 5
AND R.C = "2013-02-20"
 AND R.A = S.D
 GROUP BY R.A
ORDER BY R.C;
```

The first part of your output should be as follows:

```
PROJECTION: R.A, R.B, R.C
FROM: R, S
SELECTION: R.A, R.B, R.C
JOIN: R.A, S.D
GROUP-BY: R.A
ORDER-BY: R.C
```

If one of the attribute types doesn't exist in the query, you should put `NULL` in the output for the corresponding attribute type. After that, you should print the query results.

For every SELECT query, you will get 4 points for correctly outputting the attributes, and 6 points for correctly outputting the query results. There is no partial grading, and this is non-negotiable.

### Grading

You will be given an example query file and the corresponding data files. However, we will use a different set of files which will reflect the same characteristics with the files given to you to grade your projects.

This project is 25% of your final grade, and the total points you can get from this project is 100. There will be 10 select queries, and each select query is worth 10 points. There will also be a bonus select query which will include `LIMIT` and an aggregation, and it will be worth extra 10 points.

Between July 07, 2017 and July 14, 2017, your group need to schedule a meeting with us, and answer questions about your submission. Each member of the group will have to answer these

questions. Failing to answer these questions can result in getting a lower grade than the other members of the group. Hence, we recommend you to contribute equally, and have an idea about what your peers are working on.

### Submission

You will email me ( gokhanku@buffalo.edu ) your submissions as zip files. Your submissions should not contain the data files, but it can contain the jar files to make it easier to compile when we uncompress your submissions. If we cannot compile your code, you will receive a tentative zero (0) from the submission. When we are in the group meeting, you will be given a chance to use the same zip file to run on your machine. No 7-zip or rar files.

Only one member of your team should send the email. The email header should be `[CSE562]ProjectSubmission-`*`groupname`* and it should be CC'ed to the other members of your group.

### Deadline: **7/7/2017 (Friday) 11:59am**

The deadline is firm; if the email's timestamp is 12pm, it is a late submission. Note that the deadline is AM, not PM. We will start grading in the afternoon.

### Notes

- Please do not try to evaluate your queries directly using the SQL syntax. Try to form relational algebra trees using the query.