CS111 SQL Database Project
Milestone 2

Rutgers University New Brunswick

Due: 04/22/2015 11:59pm
Spring 2015

For this milestone you will implement functional SELECT and INSERT queries. In addition you are also required to implement an interface that continuously accepts queries and exits on a specified keyword.

Your implementation must utilize the Query module provided to you. This module can be downloaded here `http://sql.pawel.pw/Query.java`. The functionality of this module is similar to the IO.java module that you are all already familiar with. The documentation for the Query module can be found here `http://sql.pawel.pw/Query.html`. Please make sure to READ THE TOP SECTION of the Query documentation. It specifically outlines the differences in correct query vs. incorrect query. Parsing languages programatically is no easy task hence limitations had to be specified.

Throughout the document the 'employees' relation in Figure 1 below will be used for examples and explanations.

ID	Last	First	City	Age	Salary
1	Smith	Tom	Seattle	45	$280
2	Doe	Lisa	Rochester	21	$350
3	Jones	Andrew	Austin	22	$210
4	Howard	Richard	San Francisco	67	$180
5	Cameron	Adam	Tampa	32	$150

Figure 1: Database table/relation of employees

Your fields should be CASE SENSITIVE. Hence 'last' and 'LaSt' should not be thought of as the same field

SELECT Query Specifications

When executing a SELECT query, the corresponding column/field names should be printed every time along with the data. Your implementation of SELECT queries should be able to execute the following:

1. `SELECT * FROM employees;`

   This should present all of the entries in your relation along with the 'ID' field. The output should look just like in Figure 1.

2. `SELECT Last,First FROM employees;`

   This should present only the specified fields for all of the entries as shown in Figure 2 below.

Last	First
Smith	Tom
Doe	Lisa
Jones	Andrew
Howard	Richard
Cameron	Adam

Figure 2: Select query that filters fields

3. `SELECT * FROM employees WHERE ID=1;`

   This should present only the first row as shown in Figure 3 below.

ID	Last	First	City	Age	Salary
1	Smith	Tom	Seattle	45	$280

Figure 3: Select query that selects one entry

4. `SELECT Last,First FROM employees WHERE ID=1;`

   This is essentially a combination of query 2 and query 3 above.

Keep in mind that the queries show above have been specifically written for the 'employees' relation, hence the fields should correspond to your specific relation.

Use the following function header for the implementation of the SELECT query:

```
public static boolean select(Query selectQuery,[your database]) {
        // your code here
}
```

The first parameter should be acquired through the help of the Query module. The second parameter passed to the function labeled as [your database] should be the data structure that holds your database. The function should return 'true' on a successful query execution, 'false' otherwise.

INSERT Query Specifications

When executing an INSERT query, the provided values should be inserted into your database. A sample INSERT query for the 'employees' relation looks as follows

```
INSERT INTO employees VALUES(Hanks,Tom,Bumbletown,50,$500);
```

Make sure that the number of values passed matches the number of values that your database supports. You can be extra fancy and attempt to cast the values provided to you to specific type and reject certain inserts. This additional type check does not have to be implemented for full credit.

Remember that you need to auto generate the 'ID' field for every INSERT query.

Use the following function header for the implementation of the SELECT query:

```
public static boolean insert(Query insertQuery,[your database]) {
        // your code here
}
```

The first parameter should be acquired through the help of the Query module. The second parameter passed to the function labeled as [your database] should be the data structure that holds your database. The function should return 'true' on a successful query execution, 'false' otherwise.

Interface & Implementation

All of the specified functions should be implemented in 'SQLDatabase.java' file. The main function should implement an interface that allows a user to continuously type queries (use the Query module to help with this). There should be a status printed after each query is executed to indicate whether the query was a success or a failure. You are welcome to have other supporting files but the two required functions along with a main method should live in 'SQLDatabase.java'

Upon starting your program, you should present to the user the name of your table/relation along with the available fields. We understand that printing and aligning everything may prove to be a bit challenging so try to align to the best of your ability. A sample output as shown below would be acceptable.

```
_____________________
| ID | First | Last |
_____________________
| 1  | Smith | Tom  |
_____________________
```

Warning

Just as with any other coding submission make sure that your code compiles! If your code does not compile for whatever reason you will not receive any credit. NO EXCEPTIONS! So make sure to remove any package statements.

Submission

Submit 'SQLDatabase.java' file only, you do not need to attach 'Query.java' module. If you happen to have additional files on top of 'SQLDatabase.java' please zip (archive) your submission into a file called MS2_SQL.zip. Once again submit in the same place as the regular MS2, but just indicate in the submission box that you are doing the alternate project. Good luck!