COL362 Assignment 1 Report

Utkarsh Singh, 2015ME10686 **Harsh Malara, 2015EE30649**

Objective

To understand the performance of various table populating methods using the following approaches:

1. Bulk Load
2. Insert statement for each tuple
3. Insert using JDBC

Procedure

1. First, we wrote a C++ program to generate our test data. It can be found in the 'code.zip' file provided alongwith this report.
2. For Bulk Loading we generated a CSV file, for Multiple Individual Inserts we generated an SQL file and for JDBC method we generated a TXT file.
3. After that, each method of table population was executed and the timing was noted down (in seconds).

We have included all relevant files for our code in a zip file 'code.zip' which also has a README.md file explaining all the code. Also, any (explicit) assumption taken in any code was provided as comments in its source file.

Observations

We observe the following running times for populating the table *registers* with 500,000 tuples (all in seconds):

1. Bulk Load : **3.361**
2. Insert statement for each tuple : **6725.450**
3. Insert using JDBC : **38.587**

So, from our experiment, we observe that **Bulk Loading is the fastest** while **Multiple Individual Insert statements** is the slowest method for populating a table.

Bulk Loading method is arguably the fastest method of populating a table (our observation supports the same) because postgres directly take the bulk data from the CSV file and copy it to the table. It is to be noted that the only overhead here is the one associated with copying a huge chunk of data from memory, which isn't much since it's done only once.

JDBC method comes second in our observation. It doesn't fall very far off from Bulk Loading in terms of performance. JDBC seems to be a good option of populating a table as well. It seems that as the JDBC driver is in a binary file it is already complied and so it's faster to execute the inserts.

Multiple Individual Inserts performs very badly (taking about 112 minutes!) in our analysis. This is because unlike the case of JDBC, SQL shell compiles each command sequentially and thus adds to the time overheads. Also, in this case there is a return value (on shell) after each inserts which adds additional latency. Suppressing the return values using "-q" (quiet command) in command line doesn't seem to help much.

Conclusion

Our observations conclude that **Bulk Loading** is the most desirable method of populating table. **JDBC method** can be conveniently used if we need some other functionalities of Java to work with our data. **Multiple Individual Inserts** seems to be a very slow way of populating tables.