Observations

The data was downloaded every 15 minutes for about 5 days from <http://mesowest.utah.edu/data/> using bash script. The data was needed to be formatted and cleaned to be useful for our purpose, so we used python and bash scripts to reformat them. Weather data were kept in the TxWeatherData folder and Station data were kept in TxStationData and TxStationData2 folders. TxStationData contained all the original data (all 18 columns), whereas TxStationData2 contained station data with only 11 columns. We wanted to compare insertion time difference between normalized tables vs. de-normalized tables. So, TxStationData folder included data for normalized tables and TxStationData2 included data for de-normalized tables. When we

JDBC- Experimentation

* We tried to use COPY command to dump the data from files into database but this method can’t be used with normalized tables and with relation integrity constraint. The integrity rules caused COPY command to break and threw error exceptions.
* The insertion was done tuple by tuple into the database. The average insertion time was 8 msec per row. We tried turning auto commit off and turning on with a delay to increase insertion time, but didn’t notice any significant advantage.
* We tried to insert the station data in a batch of 1000 rows, and 10000 rows, but again the insertion time was same as inserting row by row (avg. 8 msec/row).

Lessons Learnt:

* Reading files manually took longer time than we anticipated.
* It’s better to deal data files in ETL process than dealing them in the application program.
* Hibernate requires less lines of code because the framework hides lots of detailed implementation.