Gaurav Singh | UFID: 36398850

COP 6726: Database Systems Implementation
Spring 2018
Weekly Assignment 7 (week after spring break)

13-03-2018:

- Mainly discussed Grokit today
- Grokit takes a long time to start up but process the data a lot faster once that initial startup cost per query is paid for. It can consume billions of records much faster that other engines.
- In fact at times the rate of data consumption is so high, that Grokit has to throw away result sets and keep a track of what sets of were thrown.
- It is better to throw records then cache for optimal speed in case of Grokit.
- RCode to PHP to C++
- PHP Templates with lots of comments are used in grokit
- GroupBys might not work in the traditional sense
- Massive code generation step takes place
- Group by is tricky, will discuss it later on
- Multiplexer is also a special GLA. Its almost a 2nd order function (takes as argument a function and date)
- MetaGLA's – take other GLAs as input
- How to GLA for group by?
    - Hash based grouping
    - Add item would look like look for group by
- Group By Map
- First manually write the code for a specific condition then generate a template using that condition as a starting point
- Group by can have constant states, that ways we can implement iterator
- Segmenter builds medium sized states will continue discussion in next class.

15-03-2018

- Grokit and Generalized Linear Aggregation
- Doing group bys in GLAs in much harder than normal where it is already a hard problem
- Use hash maps along with one of two solutions: global search or segmenter.
- In normal usage, histograms are a good example, each "bin" is separate and represents one "group" of group by.
- So store all the data in the hash map in parts and join them till they resemble the required groups
- We can do so by having individual representation globally which is terrible or by dividing data into segments and combining those
- Do segmented join lazily and it should be fine to fit a few of them in the memory at a time.