**CS 6530 Database Systems**

**Project 3: SimpleDB Operators**

**Team 15 - Tanvi Gangadhar(u1205740), Greeshma Mahadeva Prasad (u1141804)**

**DECISION DESIGN**

**Filter and Join**

- The implementation of Filter, Predicate and Join was pretty straight forward after looking at the sample Project and OrderBy Operators' implementation provided.
- For Simple Join, we are using the nested loops logic and simply going over tuples of child 1 and child 2 and merging them on satisfying the given predicate
- For HashEquiJoin, we are using a hashMap to load the tuples of child 1 and going over the map to find matches for tuples in child 2.

**Aggregates**

- For the IntegerAggregator, we are maintaining two HashMaps, one that holds the aggregate values and the other one that holds the count ( needed for COUNT and AVG aggregates)
- For all aggregates, we check if the group by field is already present in the map, if not we load an initial map assigning default values. If the field is present, we compute the aggregate. For example we add the previous value and current value for SUM, we take the MAX or MIN depending on the Op passed.
- For the StringAggregator, we maintain a single HashMap since we are only supporting COUNT for the StringAggregator. The procedure used to compute aggregate is same as above.

**HeapFile Mutability**

- To insert tuples in HeapFile and inturn in HeapPage, we are following the below algorithm:
- Get Db File(HeapFile) from catalog with the given tableId > get empty page ( check for empty slot in each page, return the first one found) > if empty page present, insert tuple ( mark slot as used) > else, create empty page, insert tuple, write heap page to heap file using randomaccessfile > return page > mark page as dirty, add page to buffer pool
- To delete tuples in HeapFile and inturn on the HeapPage, we follow a similar algorithm as the one above, by first locating the tuple on the page and then marking it as null and its slot as free.

**Insertion and deletion**

- After implementing the above HeapFile and HeapPage insert/ delete methods, Insertion and deletion Operators were pretty straightforward to implement.
- Both inserts and deletes are passed through BufferPool
- Extra care was taken to return null when insert/delete was called more than once.

**API CHANGES**

We have not made any changes to the provided API. We did not have to implement any new classes for this project.

**MISSING/INCOMPLETE ELEMENTS**

All the unit and system test cases passed, we believe we have implemented everything needed as per project 3 requirements.

**TIME SPENT**

Similar to proj1 & proj2, we pair programmed most of part of the project and together we spent about 20 hours over a few days. The breakup is as shown below,

- Understanding the provided operators such as Project, OrderBy etc and the general structure of all operators - around 3 hours
- Filter and Join - 2 hours
- Aggregates - 5 hours
- HeapFile Mutability- 5 hours
- Insertion and Deletion - 2 hours
- Query walkthrough & Parser - 3 hours

**DIFFICULTIES ENCOUNTERED**

While implementing the IntegerAggregator, we spent a lot of time figuring out how to compute average since average unlike other SQL aggregates requires all tuples. We ended up implementing it in the iterator instead of while merging the tuples.