Project4 Report

1. Catalog information about Index
- Show your catalog information about an index (tables, columns).
Our Index name is in the format of "Index_" + table name + "_" + attribute name. For example, the index file of table "left"'s attribute "B" is named as "Index_left_B."

Like tables, we put an index file's table id, table name, and file name in the Tables table.

We also put the information of its attribute which serves as a key in the Columns table.

2. Block Nested Loop Join (If you have implemented this feature)
- Describe how your block nested loop join works (especially, how you manage the given buffers.)
Implementation of the BNLJoin can be divided into several steps:

a) allocate a block of memory of size equal to "numPages * pageSize" (buffers).

b) read tuples from the left table into the block page by page until it is full. At the same time construct a separate in-memory hash table to record the pair <key, pointer to the tuple's location in the buffers>.

c) scan all tuples of the right table for the block. If a tuple in the right table matches a key in the hash table, we use the pointer in the hash table to locate the tuple in the left table and return the joined result.

d) When we finish this round of right table scan, we clear the buffers, load remaining tuples from the left table, initiate another round of right table scan, and repeat the process until all tuples in the left table have been read into the buffers and checked against tuples in the right table.

3. Index Nested Loop Join (If you have implemented this feature)
Implementation of the INLJoin can be divided into several steps:

a) setup the left iterator, the right iterator, and the condition in the initialization (the right iterator is the IndexScan class)

b) read next tuple from left iterator by getNextTuple(), if left iterator returns EOF, the INLJoin is finished, the program returns “QE_EOF”, else, continue to the step c)

c) read an attribute value of the left iterator based on the condition, named “attrVal”

d) set the right iterator based on the attrVal obtained from step c), so the IndexScan condition will be set as “key == attrVal”

e) get tuples of the right iterator, note that the right iterator matches the predicate “key == attrVal”, if the getNextTuple() of the right iterator returns 0, go to step f), else if it returns EOF, go to step b)

f) concatenate the left tuple and the right tuple, then return 0, note that the data format will be
[leftTupleNullsIndicator][rightTupleNullsIndicator][leftTupleData][rightTupleData]

4. Grace Hash Join (If you have implemented this feature)
- Describe how your grace hash join works (especially, in-memory structure).
Grace hash join is not implemented.

5. Aggregation
- Describe how your aggregation (basic, group-based hash) works.
We use Projection to select the attribute to be aggregated. And then we use the methods below to calculate the aggregated value for each aggregation operation:

MIN: Set the first tuple's value to be the aggregated value. If a subsequent tuple has a smaller value, we replace the old value with the new value.

MAX: Set the first tuple's value to be the aggregated value. If a subsequent tuple has a larger value, we replace the old value with the new value.

COUNT: Initialize the aggregated value to be zero. For each tuple read in, we increment the value by 1.

SUM: Initialize the aggregated value to be zero. For each tuple read in, we increment the value by the tuple's value.

AVG: It is the division of SUM by COUNT.

6. Implementation Detail
- Have you added your own source file (.cc or .h)?
No.

- Have you implemented any optional features? Then, describe them here.
No.

- Other implementation details:
Filter
It can be divided into several steps:

a) set the iterator, the attributes, the condition in the initialization

b) get next tuple from iterator.getNextTuple(), if the method returns 0, go to step c), else return QE_EOF

c) read an attribute value based on the condition and compare the value with the value of the condition, if the condition is satisfied, go to step d), else go to step b)

d) return 0 (since the tuple from previous step matches the condition)

Project
It can be divided into several steps:

a) set the iterator and the attributesNames in the initialization

b) get next tuple from iterator.getNextTuple(), if the method returns 0, go to step c), else return QE_EOF

c) construct a new data based on the “attributeNames”, the data format will be [ProjectedAttributesNullsIndicator][ProjectedAttributeData], and return 0