CSE177/EECS277 – DATABASE SYSTEMS IMPLEMENTATION
Project 4: Advanced Relational Algebra Operators (Join, Distinct, Sum, GroupBy)
Due date: April 5 & 6 (in the lab)

This project requires the implementation of the remaining four relational algebra operators (Join, DuplicateRemoval, Sum, and GroupBy) in the case when data fit in memory. In other words, you have to implement one-pass versions of these operators. With these operators, it is possible to run all the SQL queries supported by our reduced query language, as long as each operator in the plan can be evaluated in memory. Essentially, the output of this phase is an in-memory database server.

Join

Implement the `GetNext` method from the `Join` relational operator. You have to implement the nested-loops algorithm, with one of the relations fitting entirely in memory. There are two phases. In the *build* phase, one of the child relations – preferably the smaller – is read entirely in memory and stored in a list data structure, e.g., `TwoWayList`. In the *probe* phase, the other child relation is read one tuple-at-a-time. For every tuple, an iteration is executed over the stored list and the matching records are produced, one-at-a-time. This method is the most general form of join since it supports any join predicate. In order to find matching records, method `Run` from `CNF` has to be invoked with the two records, one from each relation. Since join records are produced one-at-a-time, you have to be careful how you manage the iteration for every probing record.

DuplicateRemoval

Implement the `GetNext` method from the `DuplicateRemoval` relational operator. You have to use a set-like data structure. Whenever a record is generated by the child operator, check to see if it appears in the set data structure. If not, return it to the caller operator and add it to the set. If it appears, ask for another record from the child operator. Repeat the process until a record can be produced or no more records exist. The most complicated part is to compare two records in order to find if they are identical or not. Class `OrderMaker` already implements this functionality in method `Run`. It is important to remember that the `DuplicateRemoval` operator appears at the top of the tree, above `Project`, and below `WriteOut`.

Sum

Implement the `GetNext` method from the `Sum` relational operator. Apply the `Function` to every record produced by the child operator. Keep a running sum that is continuously updated with the result of `Function`. When all the records are processed, create the result record containing only the sum and pass it to the parent operator. Method `Apply` from `Function` does all the work.

GroupBy

Implement the `GetNext` method from the `GroupBy` relational operator. This is a combination of the `DuplicateRemoval` and `Sum` operators. Replace the set-like data structure in `DuplicatRemoval` with a `Map` having as key the grouping attributes and as value the running sum. `OrderMaker` over the grouping attributes allows you to run comparisons between records. For every record produced by the child operator, check to see if it appears in the map. If yes, compute the function on the aggregate attributes and add the result to the running sum. If no, add the new grouping attributes to the map and initialize the running sum with the result of `Function` applied to the record. Remember that records are produced from `GroupBy` only after all the records in the child operator are processed. The order in which you generate the records is not important. However, remember that a single record is returned at-a-time. The sum aggregate appears in the first position of the created record, followed by the grouping attributes.

Requirements

1. Implement `GetNext` for the operators presented above. There are no modifications required to `main.cc` in order to support the full query language.
2. Execute the queries provided with this stage of the project over the `TPC-H` data you generated and loaded in Phase 3 of the project.
3. For correctness and performance analysis, compare the results you obtain with the results generated by some other database server, e.g., `SQLite`.

Resources

- `http://www.tpc.org/tpch/`