Database Systems - project 1 report

Lucie Hoffmann

12 April 2021

1 Introduction

Implementations between volcano, operator-at-a-time and column-at-a-time are sometimes very similar. We discuss some peculiarities in the following sections.

2 Project

In operator-at-a-time, we separate the selection vector column from the other columns, we transpose the columns to apply the evaluator, transpose the tuples to get the columns, and finally we add the selection vector column at the end.

Similarly, in column-at-a-time, we apply the evals on all columns after separating them from the selection vector, which we add at the end.

3 Scan

For the RLE scan, we get all tuples once and store them in a class variable. We iterate over all columns of the RLEStore and add every value of every column as much as the value's length in a new set of columns. We then transpose to get the tuples.

For operator-at-a-time and column-at-a-time, we get all columns from the scannable and add a selection vector with every value set to true.

4 Filter

In operator-at-a-time and column-at-a-time, for every tuple, we set the selection vector to the predicate value if the previous value was true, or let it to false otherwise.

5 Join

In volcano Join, we group the left and right inputs by the joining keys, then for each grouping key in the left input, we check whether it exists in the right

input's grouping keys and form a new tuple to add to the resulting join if so. We have tried implementing this with hash join, storing only the right input hashes and comparing them to the left inputs hashes but the running time was worst. Same observation for the RLEJoin.

For the RLEJoin, we store only the right inputs and compare them to the left inputs as they come via the next function.

For operator-at-a-time and column-at-a-time, we optimized using hash join, storing both input hashes, comparing them to each other and getting the corresponding values if matching.

6 Aggregate

For operator-at-a-time and column-at-a-time, we process only tuples with a selection vector value set to true. All outputs of these operators have a all selection vector values set to true.

7 Sort

The implementation is the same over all volcano, operator-at-a-time and column-at-a-time. We optimized by dropping some tuples in intermediate results using offset and fetch, but the speedup was not much higher after this.

Every time we sort on a field:

- We check whether this field at offset-1 is different than that at offset. If so, we drop the offset.
- We check whether this field at 0 is different than that at offset-1. If so, we drop the first N rows with a different field and set the offset to offset-N.
- We check whether this field at offset+fetch-1 is different than that at offset+fetch. If so, we drop elements after (included) offset+fetch.
- We check whether this field at last is different than at offset+fetch-1. If so, we drop the last M rows with a different field.

8 RLE rules, Decode and Reconstruct

It was hard to find the right syntax for the rules implementation, but once found for one then the others are similar and straightforward.

In Decode, we keep a class variable of the lasting number of repetitions in the current RLE entry based on this entry's length, and change to the next entry when this variable becomes 0.

In Reconstruct, we keep class variable for the current left and right entries. We set the endVID of the current resulting entry to the minimum of the left and right entries endVID, and the startVID to the maximum of the left and

right entries startVID. We update the current left/right entry when there is no intersection or when one of them has the minimum endVID.

9 Notes

Repeating similar but not exact same code between volcano, operator-at-a-time and column-at-a-time may be the reason behind the fact that some optimization work in some cases and not in others.