CS-422: Database Systems

Sophia Artioli

Project 1

Task 1: Volcano-style tuple-at-a-time engine

The implementation of the six operators is quite straightforward once the input format is understood. I used hash join to implement the join as it was making some tests timeout in its original form.

Task 2: Late Materialization

The implementation of the operators in late materialization were similar to the ones in task 1. The major difference is the vid attribute that needs to be taken care of especially in the join.

Task 3: Query Optimization Rules

The hardest task due to the lack of explanation. The instructions were very hard to interpret. I only managed to implement LazyFetchRule and LazyProjectRule by basically just reading the documentation and doing pattern matching with the types needed for the LazyFetch.create() function.

Task 4: Execution Models

Once I visualized how column-at-a-time worked after having implemented tuple-at-a-time, the implementation was pretty straightforward as it was similar after having transformed the input. Once the columns are transposed, I was able to do like-tuple operations and once all the transformations were done, I simply transformed them back to columns.
Once operator-at-a-time was implemented, column-at-a-time was very similar, the only difference was to get how the input worked and to transform the Columns to Homogenous Columns.

Feedback

I found this project very difficult to get started with. It was hard to get a grasp of the input form and where it came from/what it looked like. I felt like I did not have the whole context to actually understand why I was doing what I was doing. Perhaps, some examples and more context would have been helpful. On the other hand, the implementation was pretty straight forward but very long. Coming back to scala after python was difficult. I spent most of my time understanding how functions and types worked rather than thinking about efficient algorithms to implement the operators.