AsterixDB Interval Join

Dynamic Hint Deduplication

Edward Coberly

WWU Department of Computer Science

Walla Walla University
SEVENTH-DAY ADVENTIST HIGHER EDUCATION

ABSTRACT

Apache AsterixDB is a database system maintained by the Apache Software Foundation (ASF). Users write queries in SQL++ to create, manage, and analyze data. AsterixDB provides its users with functions that ease the data analysis process.

This project focused particularly on interval data. Among the interval data analysis tools are interval joins, which examine overlaps between datasets that use dates, times, or date-time mixes. With a static hint, users can specify a range map (i.e., a set of partitions) over which they are looking for overlaps. With a dynamic hint, the program decides how to partition the data.

INTRODUCTION

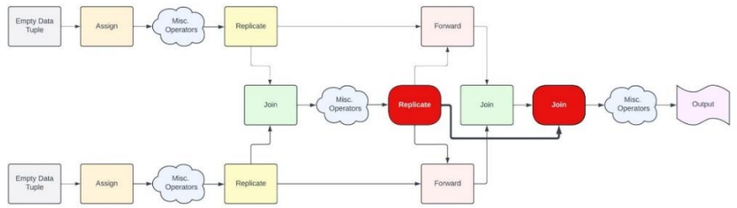
The `interval_overlapping()` function, which comes from the suite of interval join functions, had some problems, and my project was to fix this function.

The purpose of the `interval_overlapping()` function is to return a JSON-format result of all the data points that overlap over a particular interval of times, dates, or both (e.g., employment of teachers and enrollment of students).

The problem with this function happened with a dynamic hint. The function returned duplicate data with the dynamic hint because each partition is checked individually, so if two intervals overlap in two or more partitions, those data points would be returned more than once.

DATA AND ANALYSIS

The data flow starts with the two intervals, which are supplied by the user from their dataset of choice. The data must be processed and converted to a byte array and go through a series of operators that prepare it to be joined. When the user opts for a dynamic hint, the data flow generates its own range map to be compared with each interval, but when the intervals are joined at the end, the previous Join operator would include the duplicate data.



The map shown above describes my change to the data flow. Although the data is processed correctly, my code inserts a stopping point in the data flow that looks over the processed data and chooses each data point once.

Using the results of the current interval join, I implemented one more Join operator that compares the joined data with the machine-generated range map and only scans for data points not already put into a return array. This ensures that when the data is returned, every pair of overlapping data points shows up only once.

This range map must be converted from a byte array to an object, so I also implemented a deserialization function and a test that confirms that the function converts it properly.

CONCLUSION

Building on existing code was much more difficult than I thought it would be. I learned how to properly use many of the existing data types and functions in the AsterixDB framework, as well as how to create my own.

This project pushed me farther than any class in my undergraduate experience, requiring me to adapt and learn new skills, like learning how to code in Java and use an IDE specialized for Java development. While this project was challenging and even frustrating at times, I felt immense satisfaction when the query came back successfully with no duplicate data and all of my tests passed.

SUMMARY

Originally, the `interval_overlapping()` function only fully supported static hints. Now, with my code, the function fully supports dynamic hints and returns the correct data. By modifying the data flow with an additional Join operator, I was able to deduplicate the data resulting from an interval join.

REFERENCES

https://asterixdb.apache.org/dev-setup.html
https://asterixdb.apache.org/pushing.html
https://asterixdb.apache.org/docs/0.9.9/sqlpp/builtins.html
https://asterixdb.apache.org/docs/0.9.9/interval_join.html

Department of Computer Science, Walla Walla University