APACHE ASTERIXDB

Kaden Fairchild

Project Overview

- Goal: Creating a query without hint function for interval join algorithms in Apache AsterixDB
- Create a union interval range map function for automatic index creation
- Create dynamic query plan for joins without hint provided
- Eliminate the need for manual dataset searches

Union_interval_range

```
INSERT into Staff
([{ "id": 14, "name": "Alex", "office": "A", "employment": interval(date("2003-01-01"), date("2008-01-01")), "vacation":
interval(datetime("2003-01-01T00:00:00.0"), datetime("2008-01-01T00:00:00.0")), "office_hours": interval(time("03:00:00.0+00:00"),
time("08:00:00.0+00:00"))},
{ "id": 13, "name": "Elisabeth", "office": "B", "employment": interval(date("2002-01-01"), date("2010-01-01")), "vacation":
interval(datetime("2002-01-01T00:00:00.0"), datetime("2010-01-01T00:00:00.0")), "office_hours": interval(time("02:00:00.0+00:00"),
time("10:00:00.0+00:00"))},
{ "id": 16, "name": "Franklin", "office": "A", "employment": interval(date("2004-01-01"), date("2009-01-01")), "vacation":
interval(datetime("2004-01-01T00:00:00.0"), datetime("2009-01-01T00:00:00.0")), "office_hours": interval(time("04:00:00.0+00:00"),
time("09:00:00.0+00:00"))},
{ "id": 15, "name": "Henry", "office": "C", "employment": interval(date("2003-01-01"), date("2008-01-01")), "vacation":
interval(datetime("2003-01-01T00:00:00.0"), datetime("2008-01-01T00:00:00.0")), "office_hours": interval(time("03:00:00.0+00:00"),
time("08:00:00.0+00:00"))},
{ "id": 17, "name": "MaryAnn", "office": "B", "employment": interval(date("2006-01-01"), date("2010-01-01")), "vacation":
interval(datetime("2006-01-01T00:00:00.0"), datetime("2010-01-01T00:00:00.0")), "office_hours": interval(time("06:00:00.0+00:00"),
time("10:00:00.0+00:00"))},
{ "id": 11, "name": "Vicky", "office": "D", "employment": interval(date("2001-01-01"), date("2010-01-01")), "vacation":
interval(datetime("2001-01-01T00:00:00.0"), datetime("2010-01-01T00:00:00.0")), "office_hours": interval(time("01:00:00.0+00:00"),
time("10:00:00.0+00:00"))},
{ "id": 12, "name": "Zack", "office": "A", "employment": interval(date("2002-01-01"), date("2003-01-01")), "vacation":
interval(datetime("2002-01-01T00:00:00.0"), datetime("2003-01-01T00:00:00.0")), "office_hours": interval(time("02:00:00.0+00:00"),
time("03:00:00.0+00:00"))},
{ "id": 18, "name": "Jake", "office": "C", "employment": interval(date("2006-01-01"), date("2008-01-01")), "vacation":
interval(datetime("2006-01-01T00:00:00.0"), datetime("2008-01-01T00:00:00.0")), "office_hours": interval(time("06:00:00.0+00:00"),
time("08:00:00.0+00:00"))}]);

select `union_interval_range`(vacation) from Staff;
```

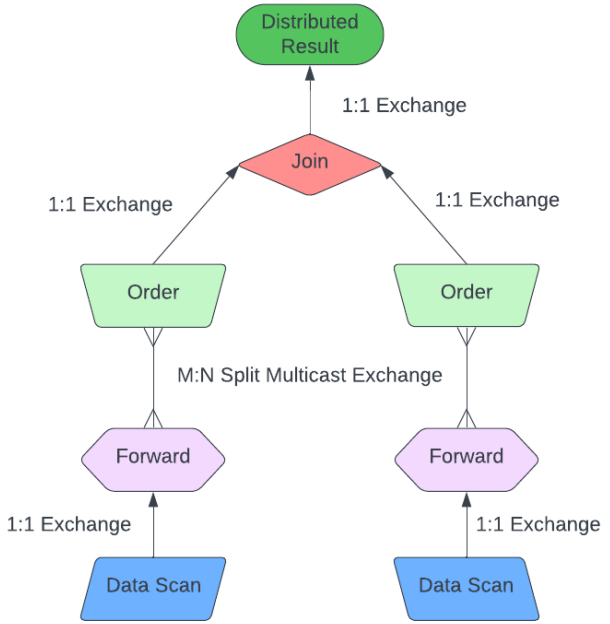
- Returns the interval range of a given attribute in data

$1
{ "interval": { "start": "2001-01-01T00:00:00.000", "end": "2010-01-01T00:00:00.000" } }

Query Plan

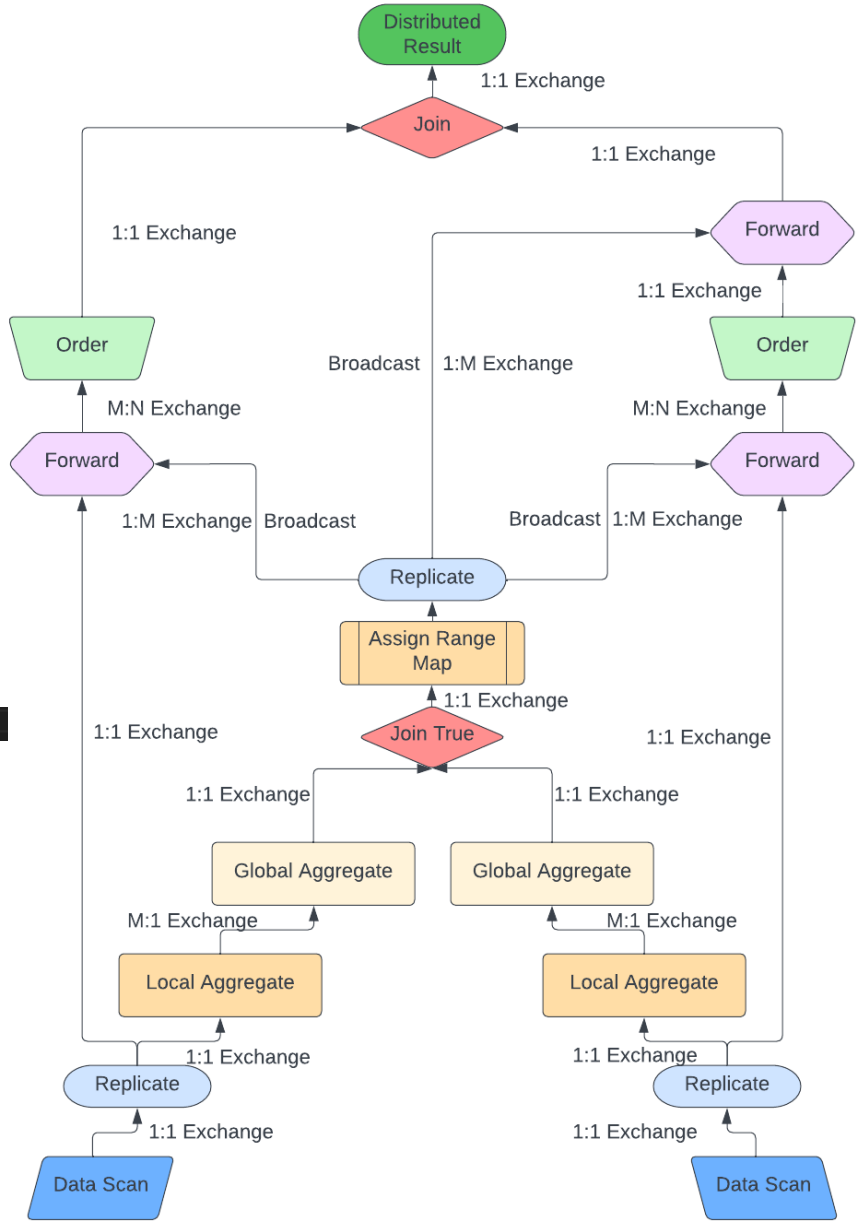
- Dynamic workflow for interval computation
- Range calculators, global range aggregate operators, and forward operators
- Inner join operator for finding interval overlaps
- Range map creation and replication
- Partitioning exchange operators for interval partitioning
- Interval merge join physical operator for efficient join operation

Interval Join With Static Hint




```
WHERE interval-merge-join [-1,0,1753200,3657600,5637600,7887600,9784800]
```

Interval Join With Dynamic Hint



Original Timeline

Fall

- Setup IntelliJ IDEA
- Create Project Workflow
- Review code written in AsterixDB

Winter

- Testing query hint functionality
- Develop query without hint
- Testing

Spring

- Benchmark Query without hint
- Create forward scan interval join algorithm
- Prepare presentation and poster

Customer Meetings

Weekly throughout all quarters

Actual Timeline

Fall

- Setup IntelliJ IDEA
- Create Project Workflow
- Review code written in AsterixDB

Winter

- Testing query hint functionality
- Develop dynamic query plan
- Debugging

Spring

- Debugging
- Prepare presentation and poster
- Write Blogpost

Customer Meetings

Weekly throughout all quarters

Challenges

Undoubtedly, the most challenging aspect of the project revolved around formulating the logical and physical execution plan for the query.
This intricate process entailed the implementation
of exchange, replicate, and forward operators, all
vital components necessary to successfully execute the query plan.

Completed Goals

- Updated old branch to be usable with new version of Apache AsterixDB
- Created union_interval_range function
- Made Tests for union_interval_range function
- Refactored old code and improved naming conventions
- Created dynamic physical and logical query plan for no hint interval joins

Future Work

- Refactor code
- Get it to work with optimization rule enabled
- Go through code review