Alibaba Cloud | Worldwide Cloud Services Partner

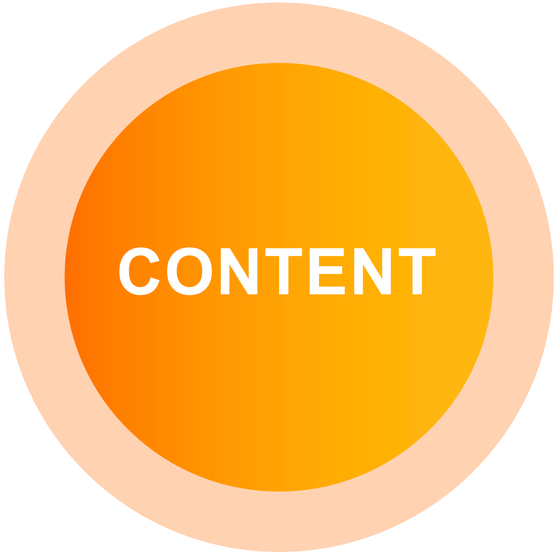
A Deep Dive into Flink SQL

Jark Wu

Software Engineer at Alibaba

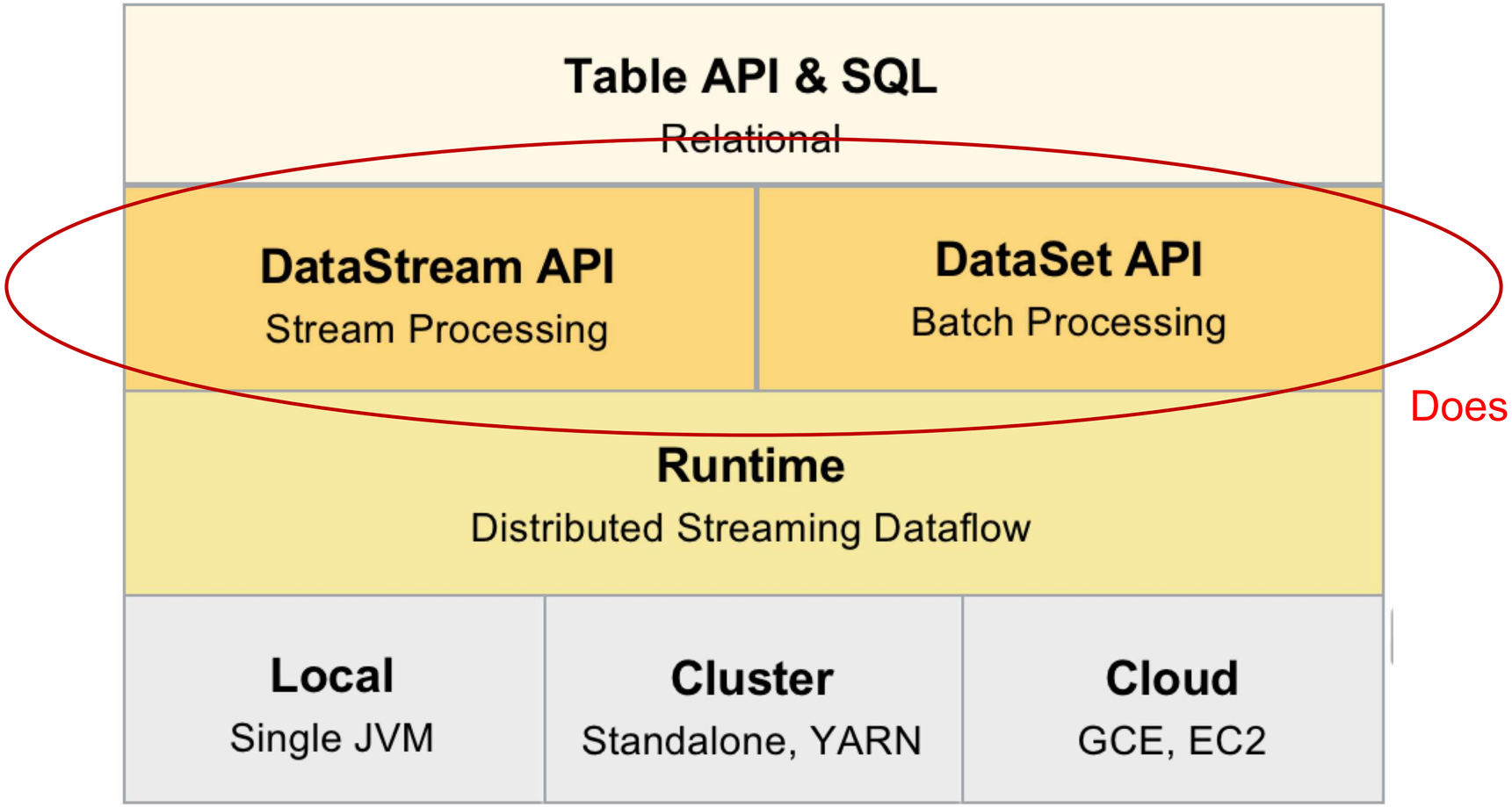
Apache Flink Committer & PMC member

Content



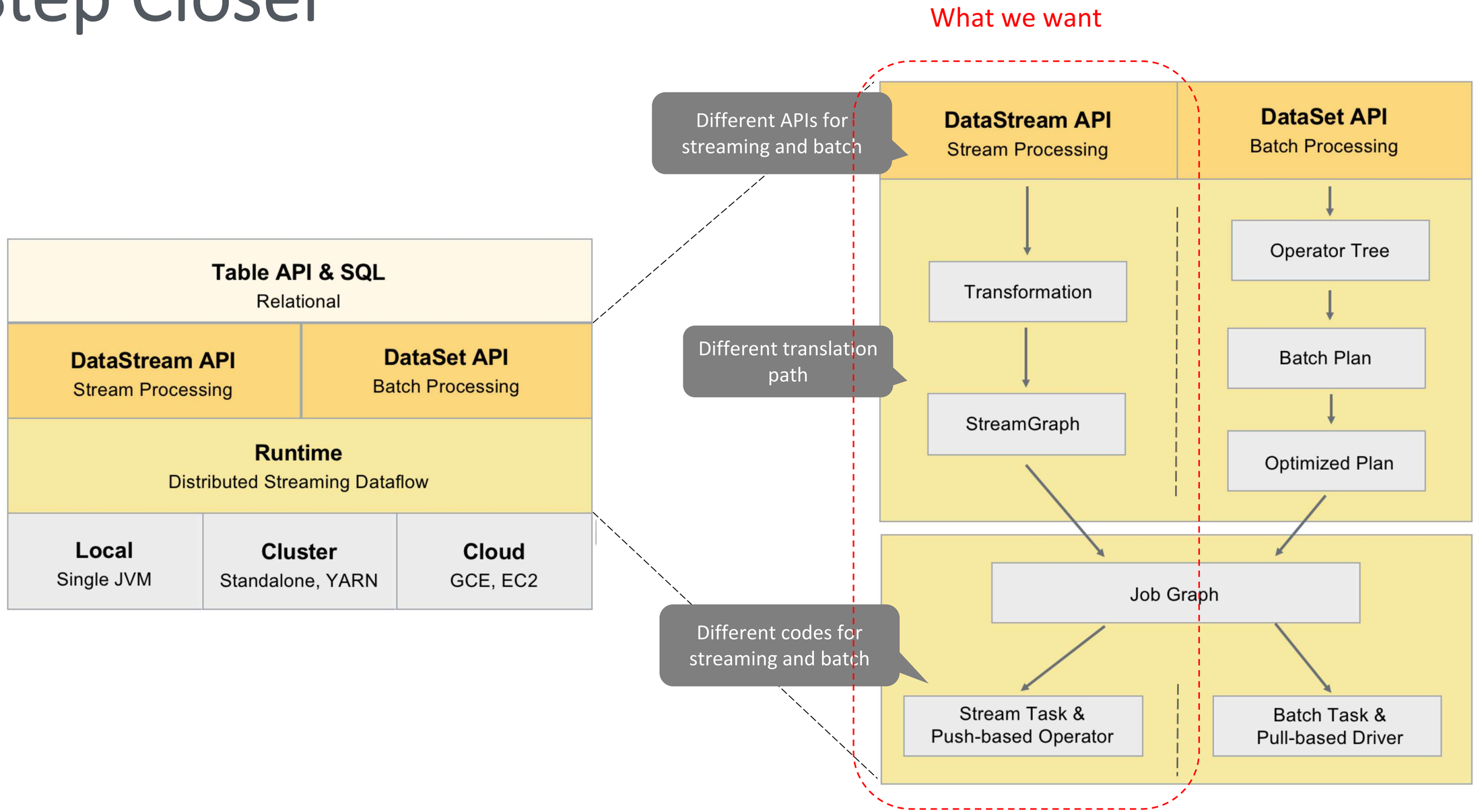
This is joint efforts with entire Apache Flink community!

Architecture before Flink 1.9

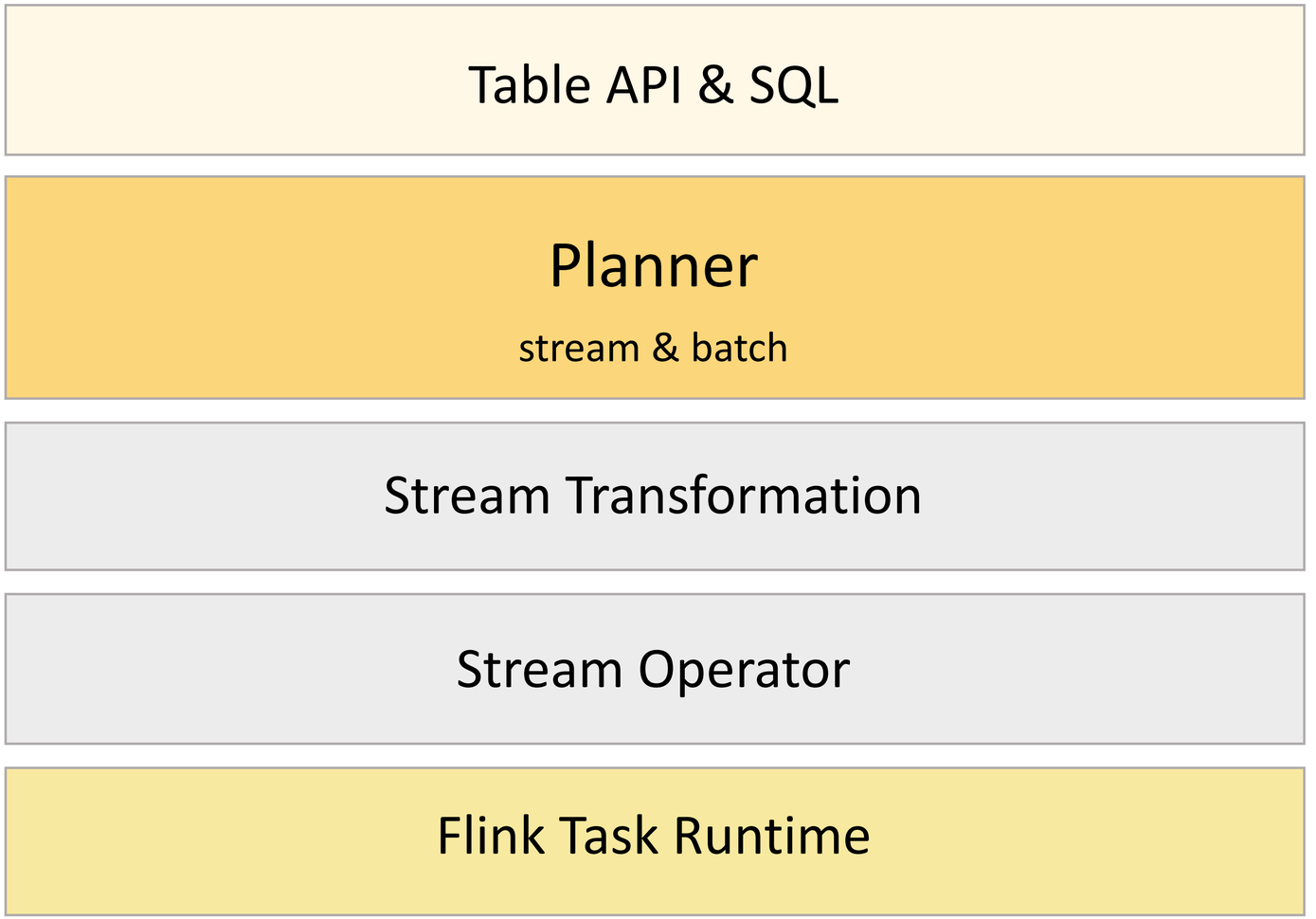


Does this look unified?

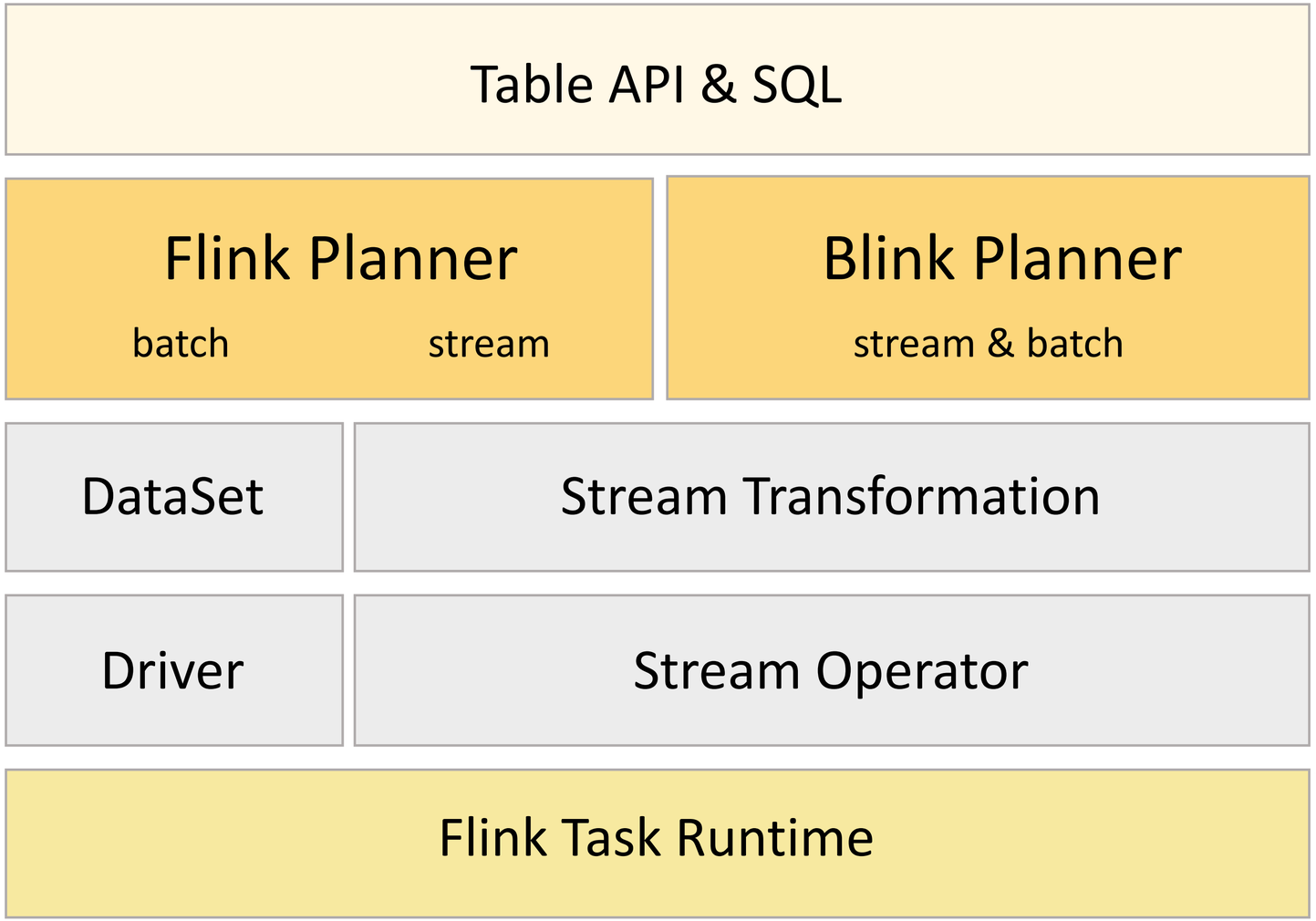
A Step Closer



Future Architecture

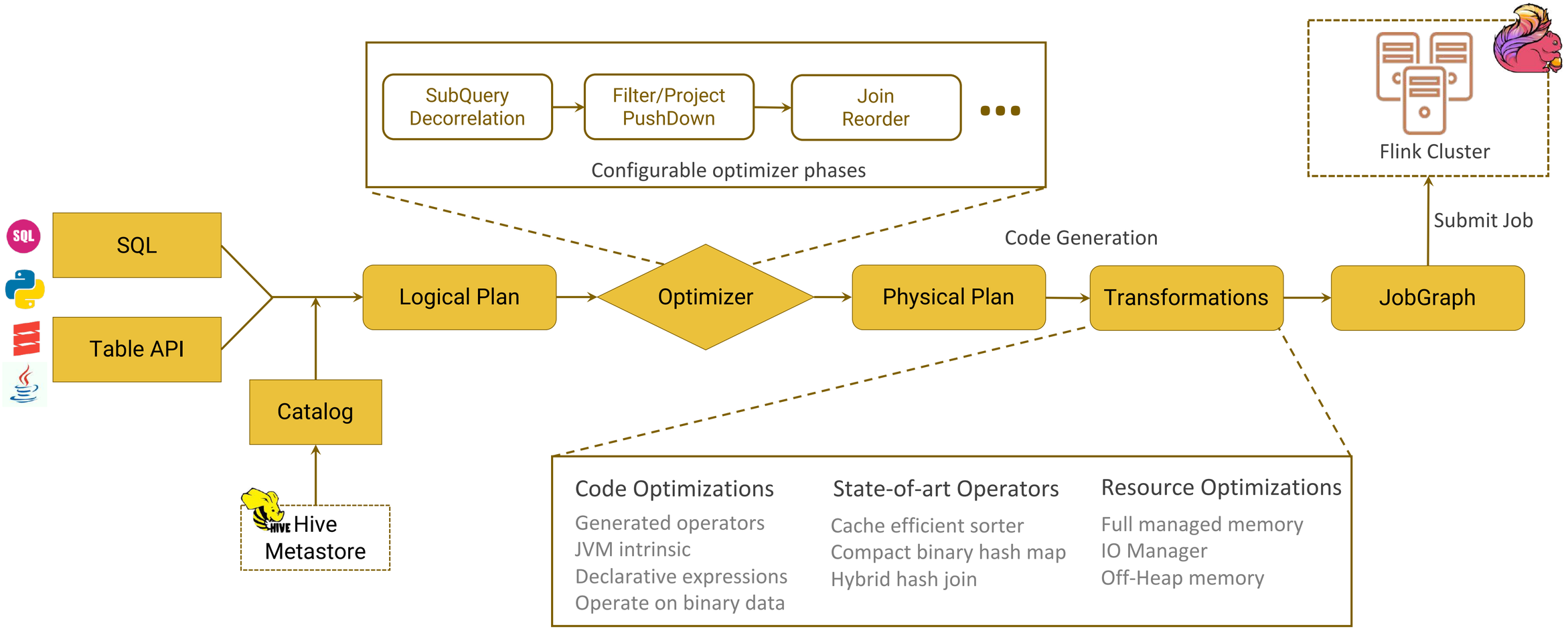


Architecture since Flink 1.9+



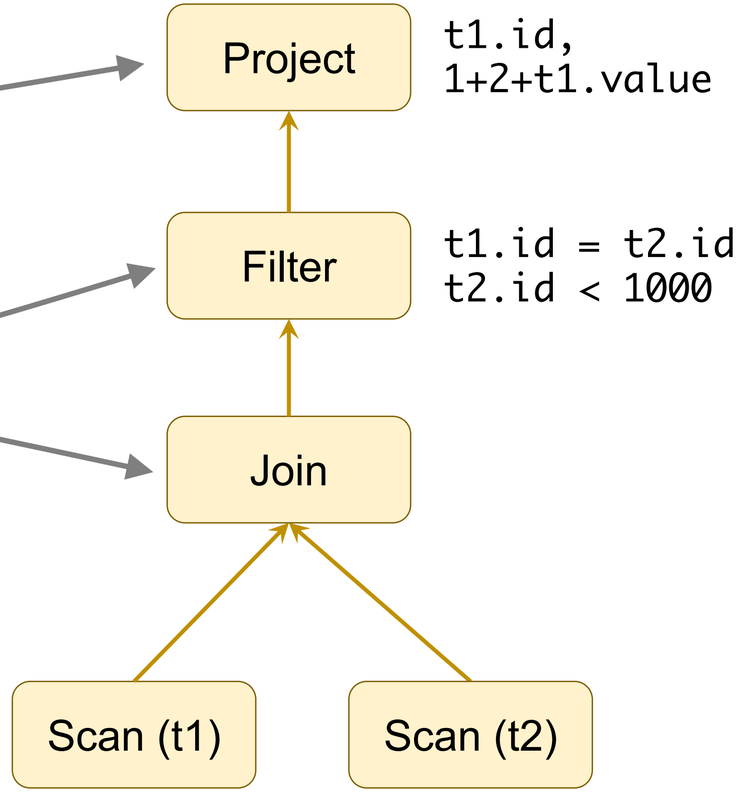
How Flink SQL Works ?

How Flink SQL Works?



An Example

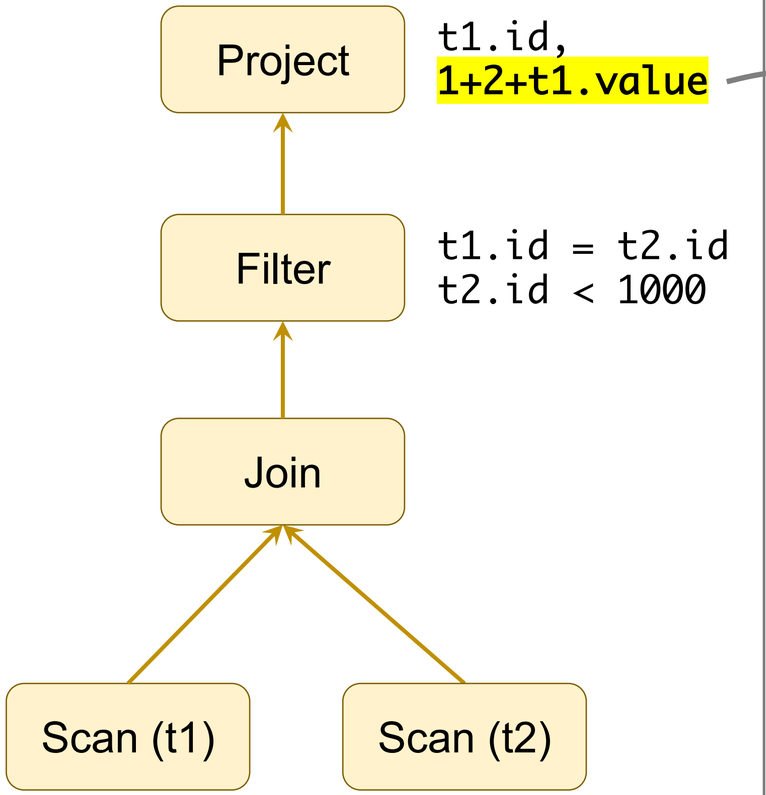

```
SELECT
  t1.id, 1 + 2 + t1.value AS v
FROM t1 JOIN t2
WHERE
  t1.id = t2.id AND
  t2.id < 1000
```



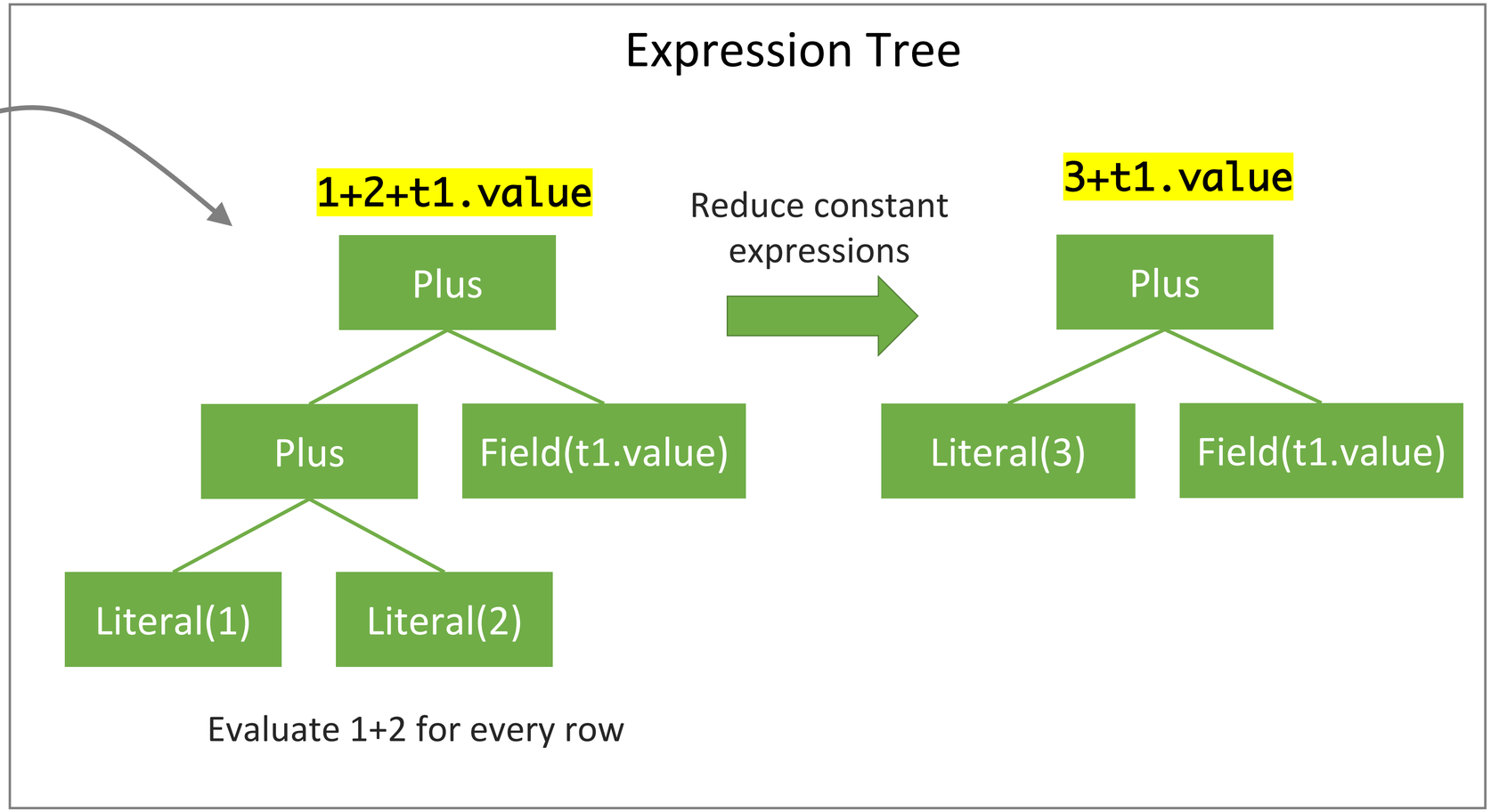
SQL Query

Logical Plan

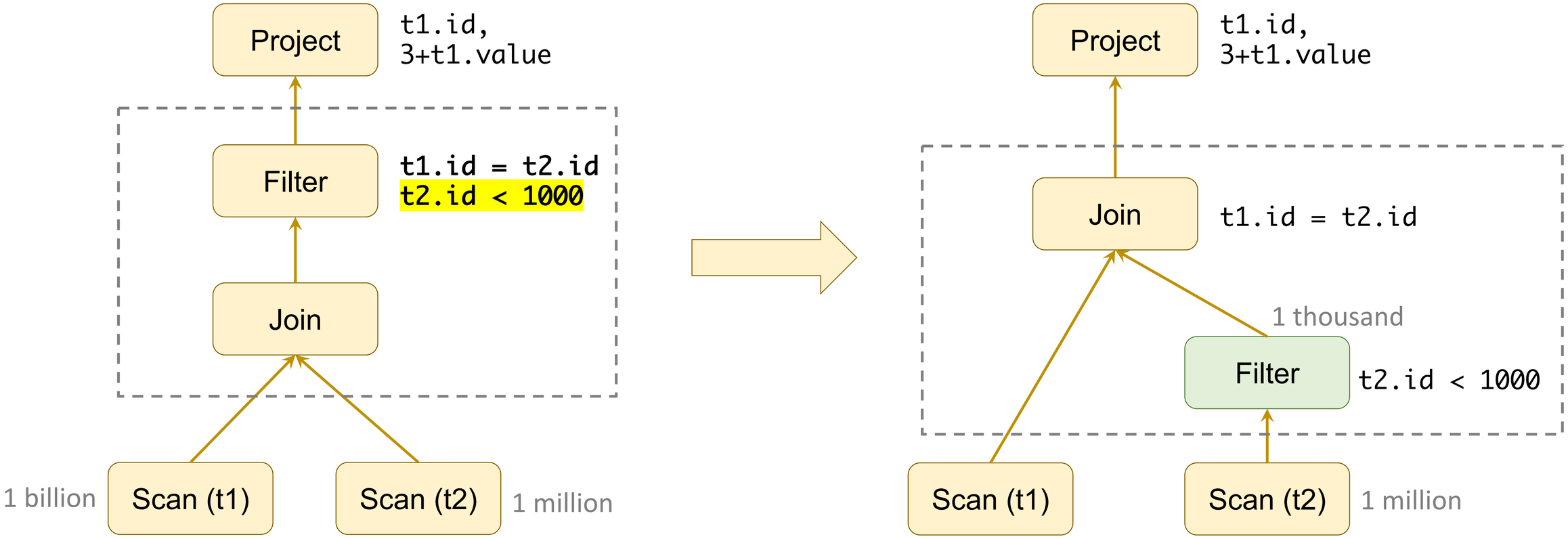
Expression Reduce



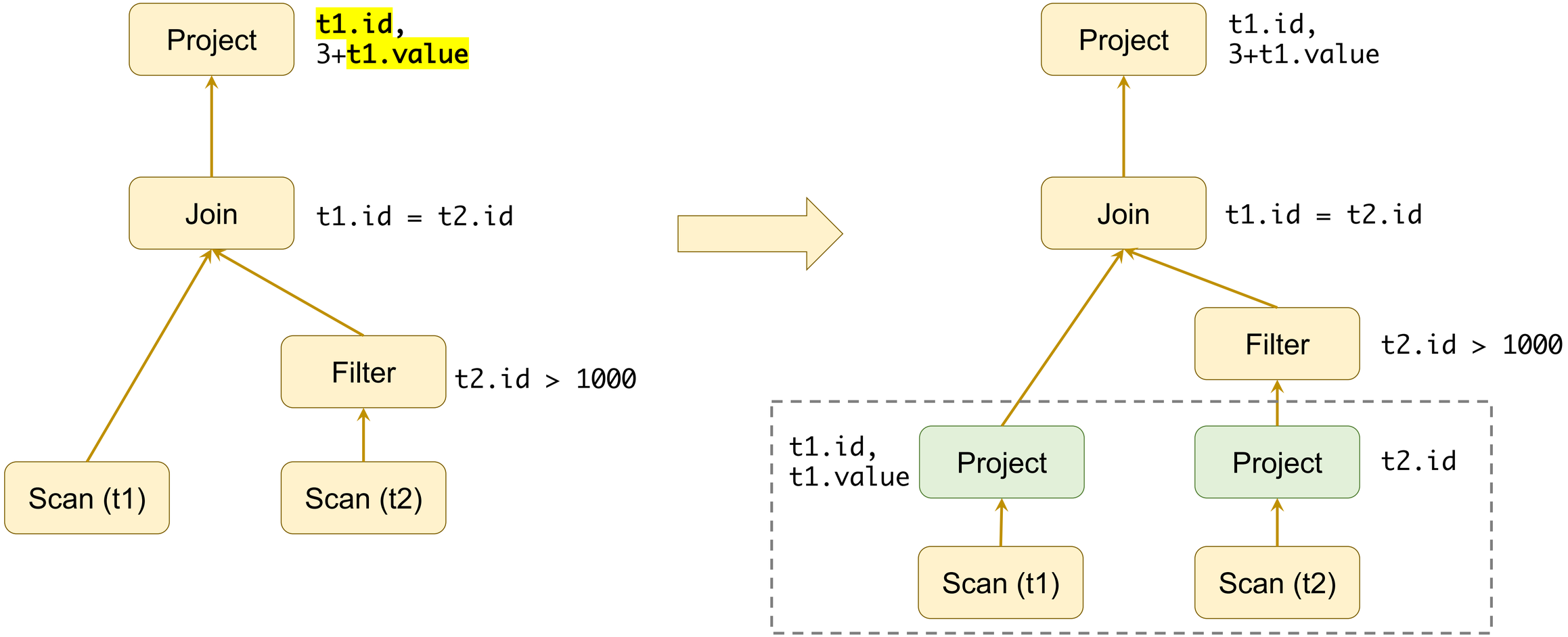
Logical Plan



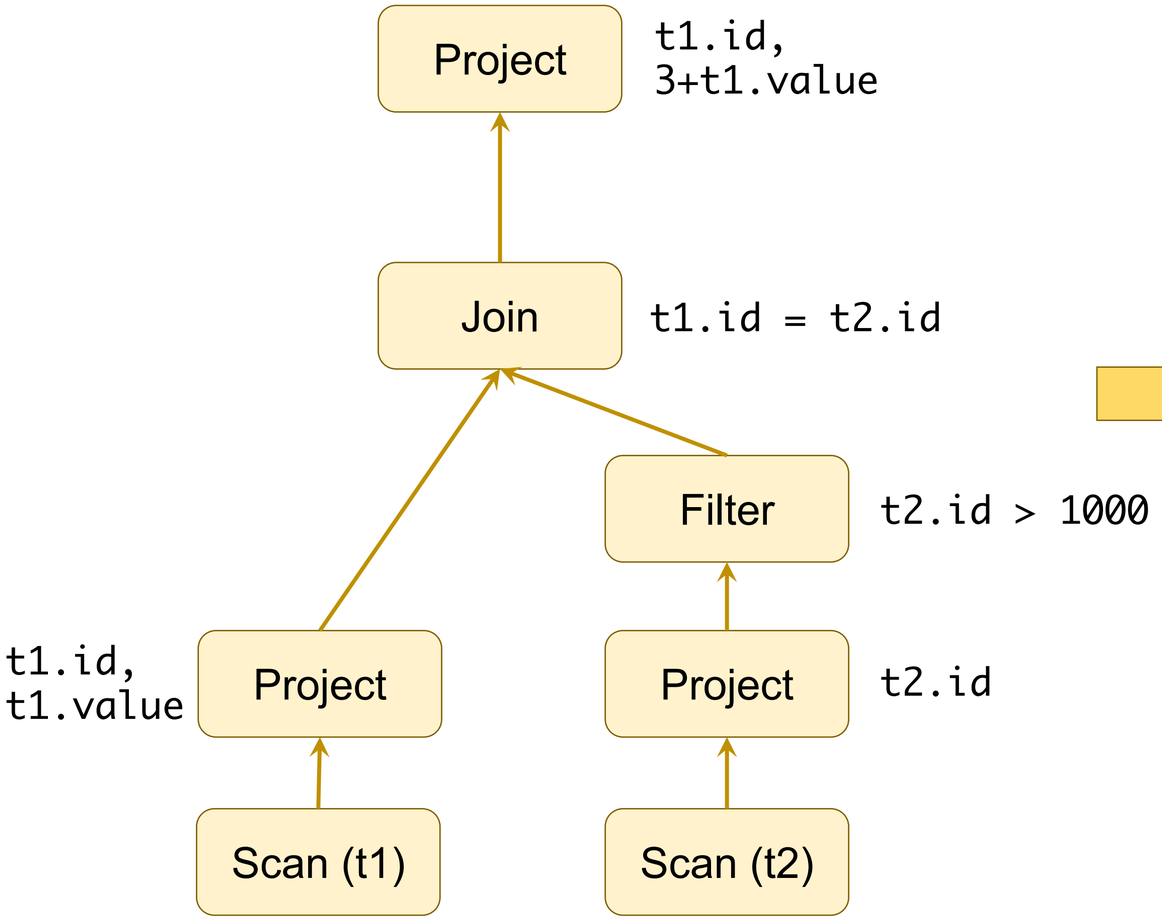
Filter Push Down



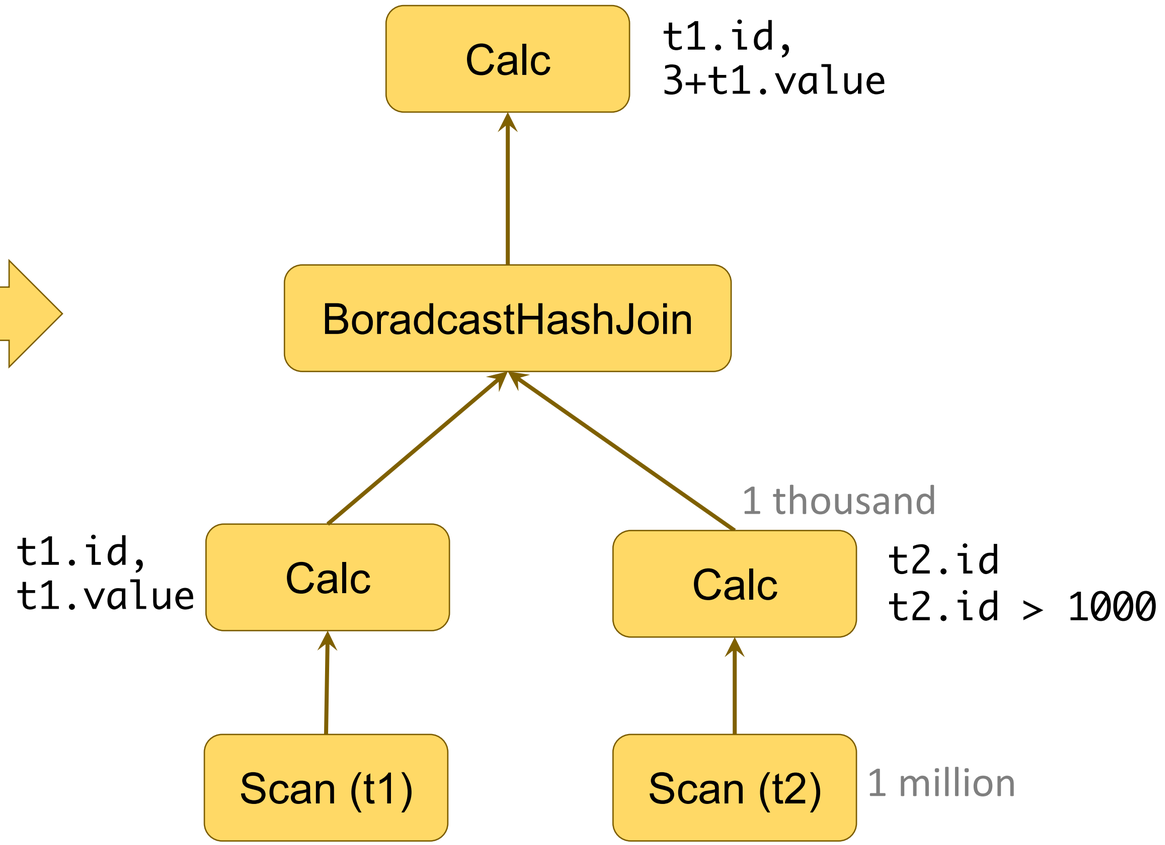
Projection Push Down



Physical Planning (Batch)

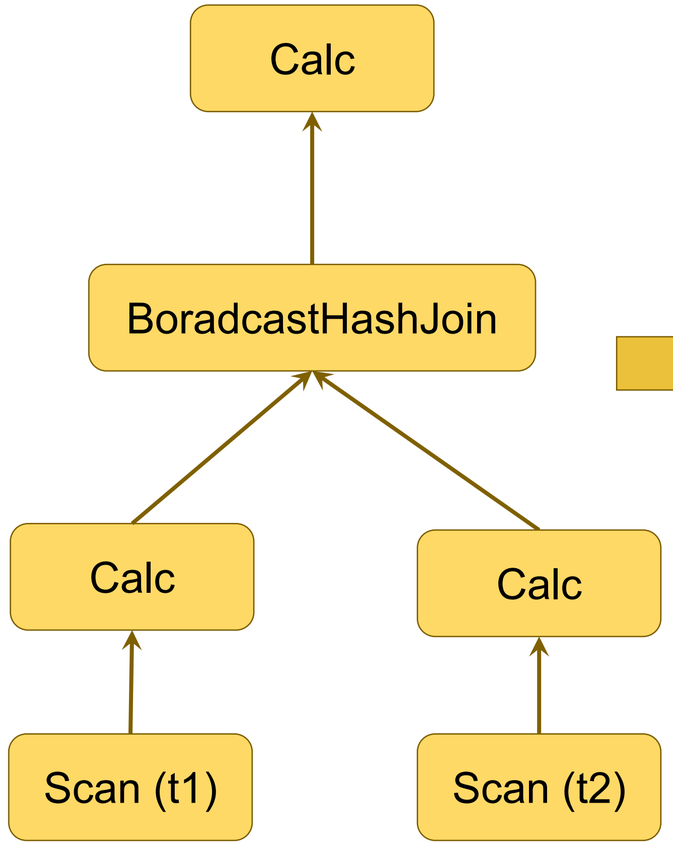


Optimized Logical Plan

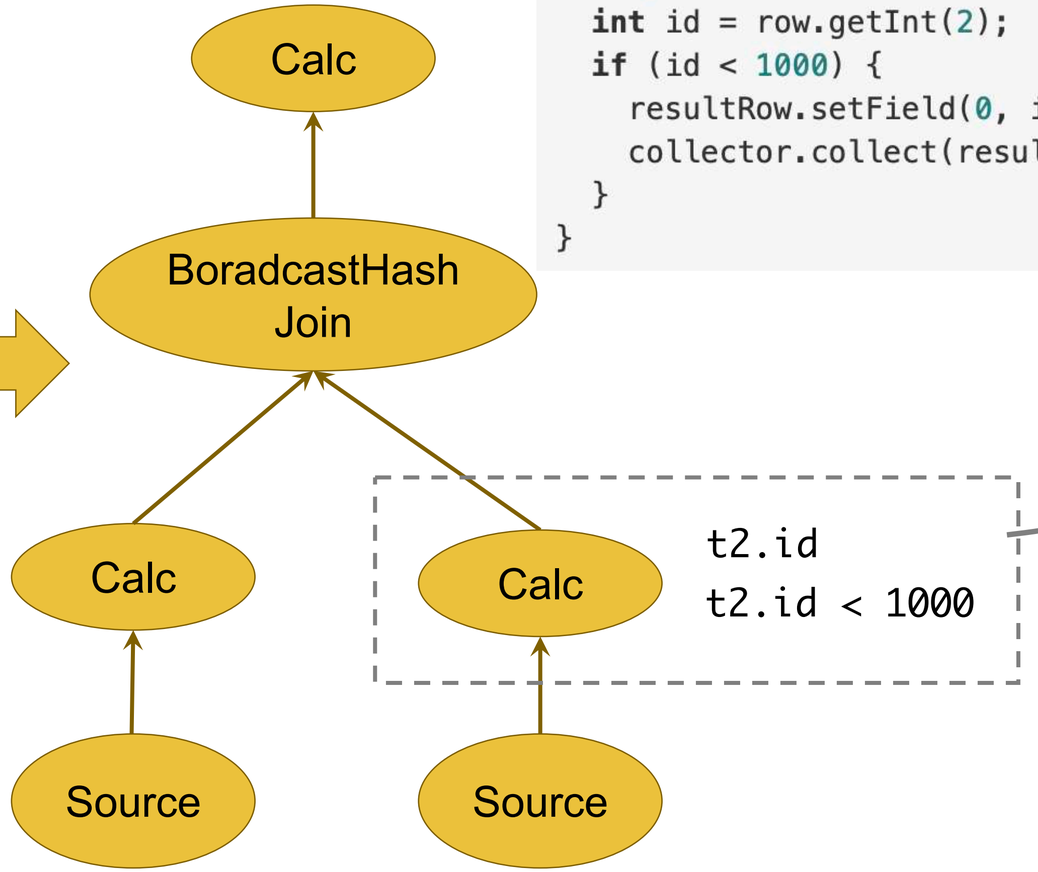


Physical Plan

Translation & Code Generation



Physical Plan

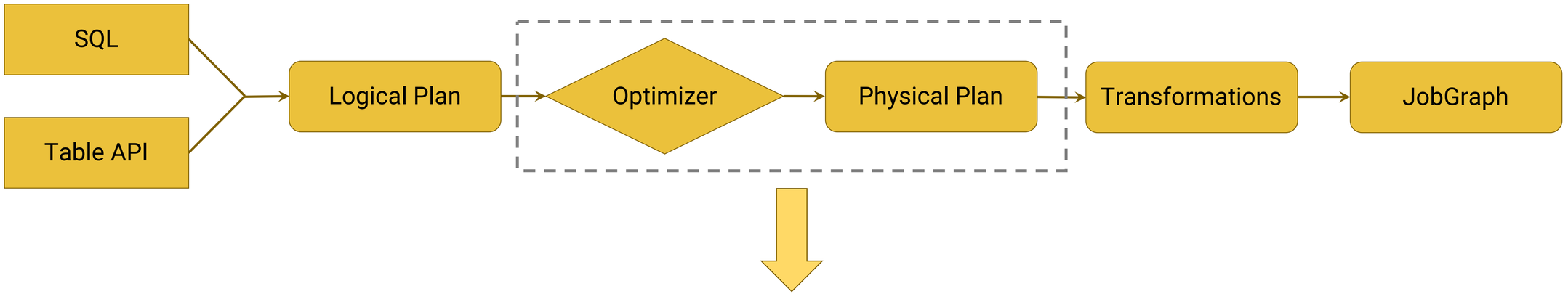


Transformation Tree

```
private final resultRow = new GenericRowData(1);
void processElement(RowData row, Collector<RowData> collector) {
  int id = row.getInt(2);
  if (id < 1000) {
    resultRow.setField(0, id);
    collector.collect(resultRow);
  }
}
```

code generation

Physical Planning (Stream)



Special things for streaming: Changelog Mechanism

aka Retraction Mechanism

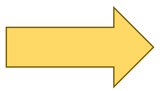
What is changelog and Why we need it?

Physical Planning (Stream): Changelog Mechanism

```
SELECT cnt, COUNT(cnt) as freq
FROM (
  SELECT word, COUNT(*) as cnt
  FROM words
  GROUP BY word )
GROUP BY cnt
```

Source

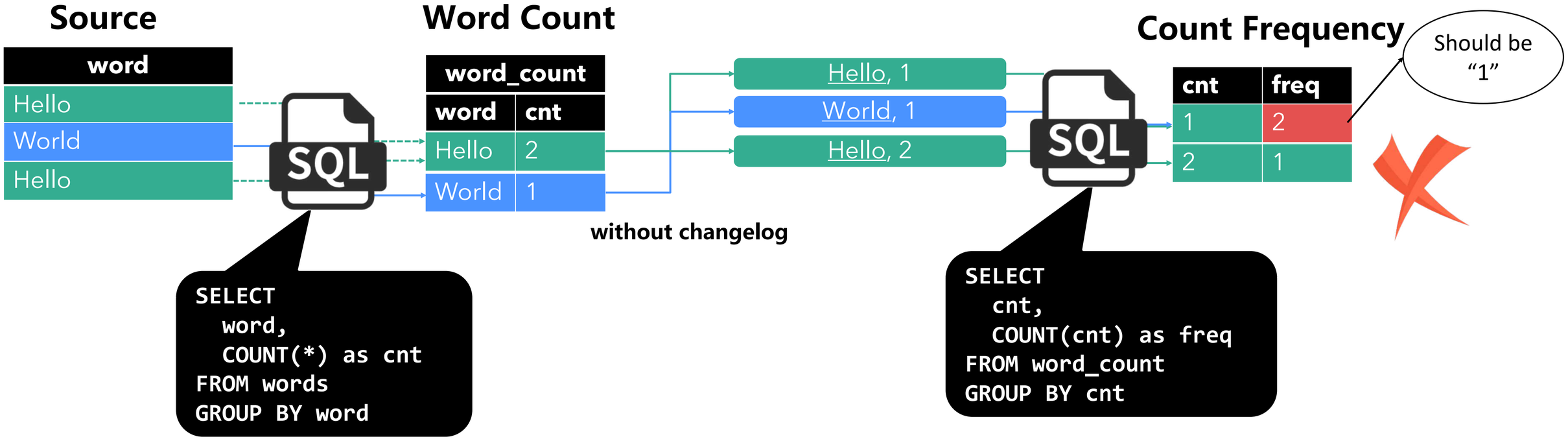
word
Hello
World
Hello



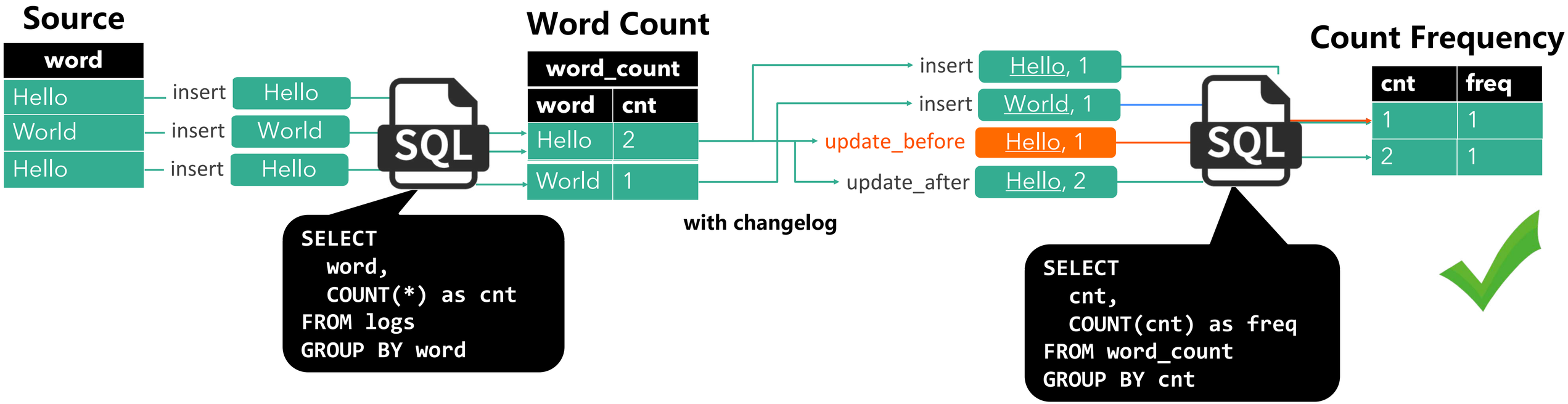
Expected Result

cnt	freq
1	1
2	1

Physical Planning (Stream): Changelog Mechanism



Physical Planning (Stream): Changelog Mechanism



① Changelog makes the streaming query result correct

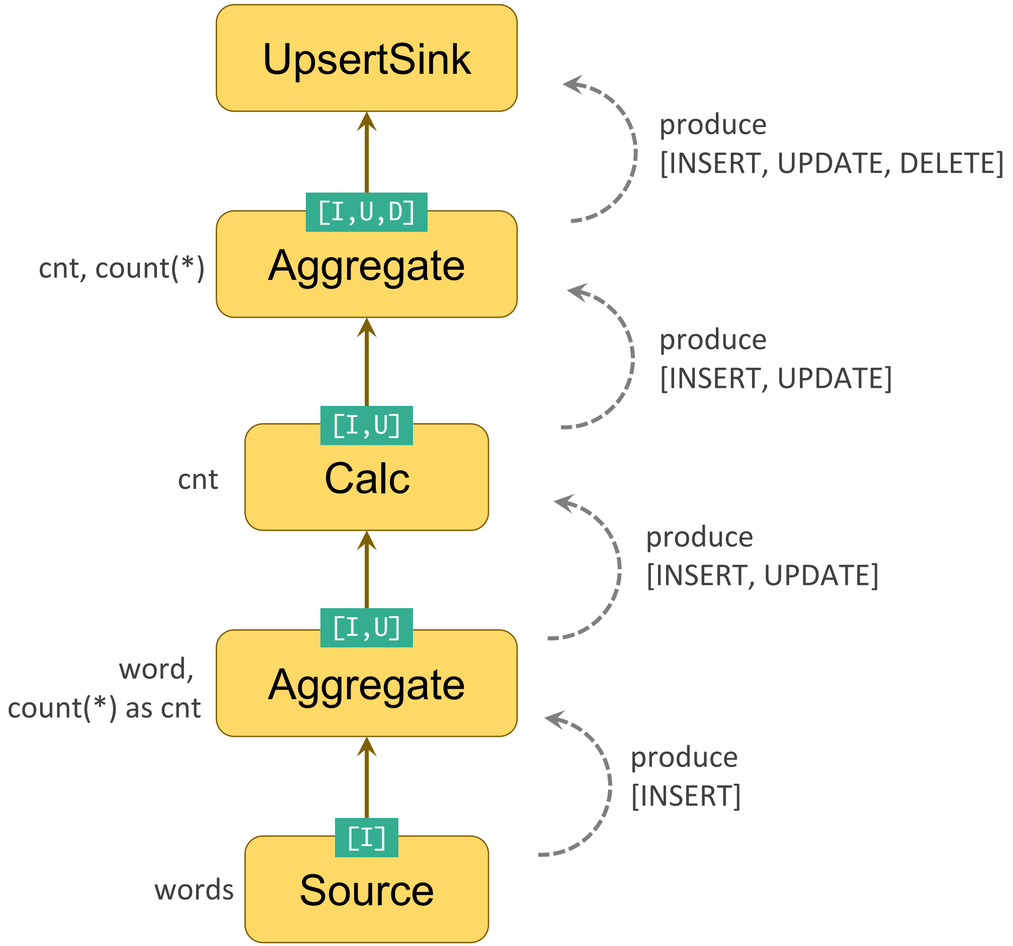
② Query optimizer determines whether update_before is needed

③ Users are not aware of it

Physical Planning (Stream): Changelog Mechanism

Step1: determine what changes will a node produce

```
[I]: Insert
[U]: Update
[D]: Delete
```

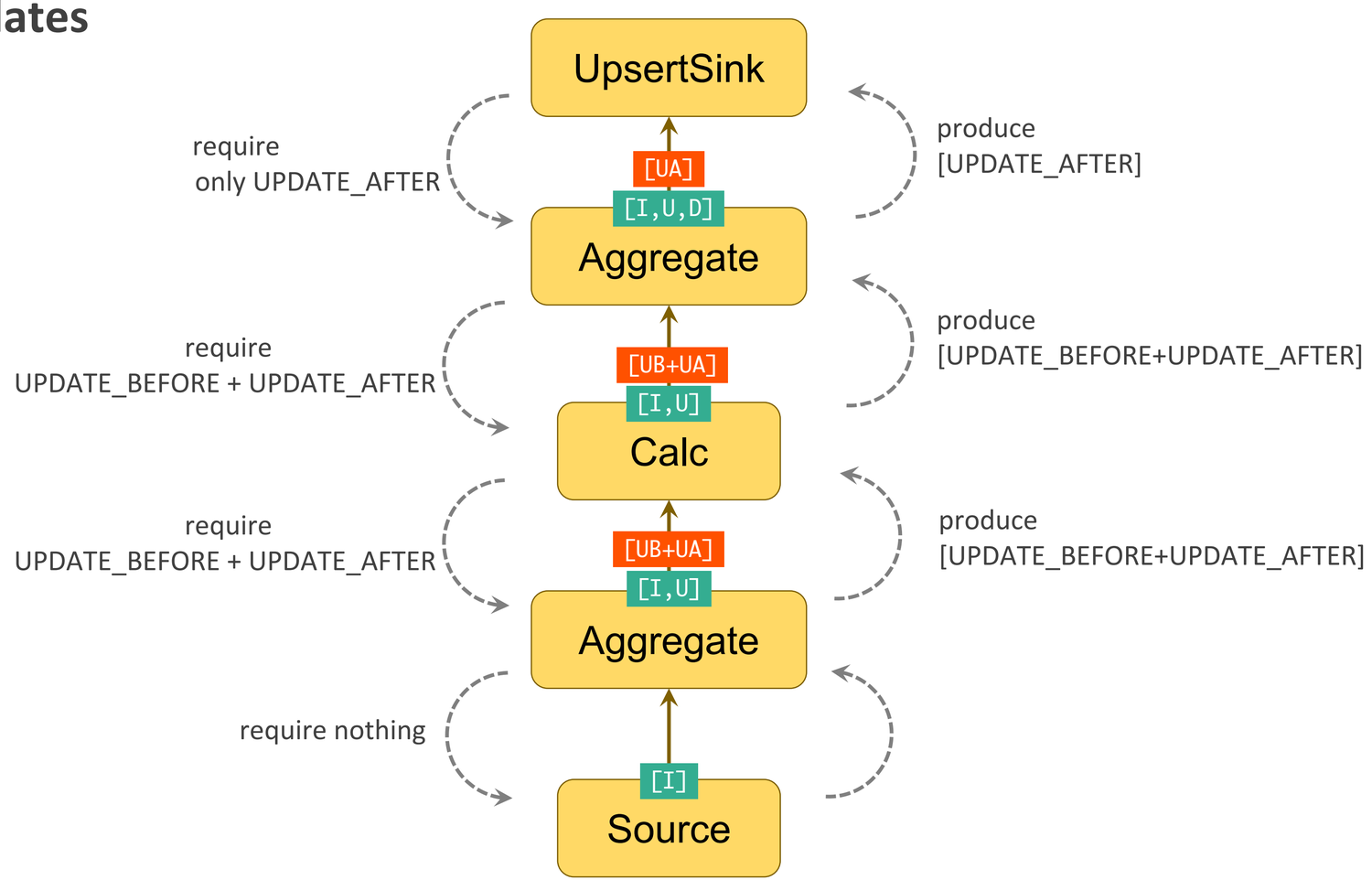


Physical Plan

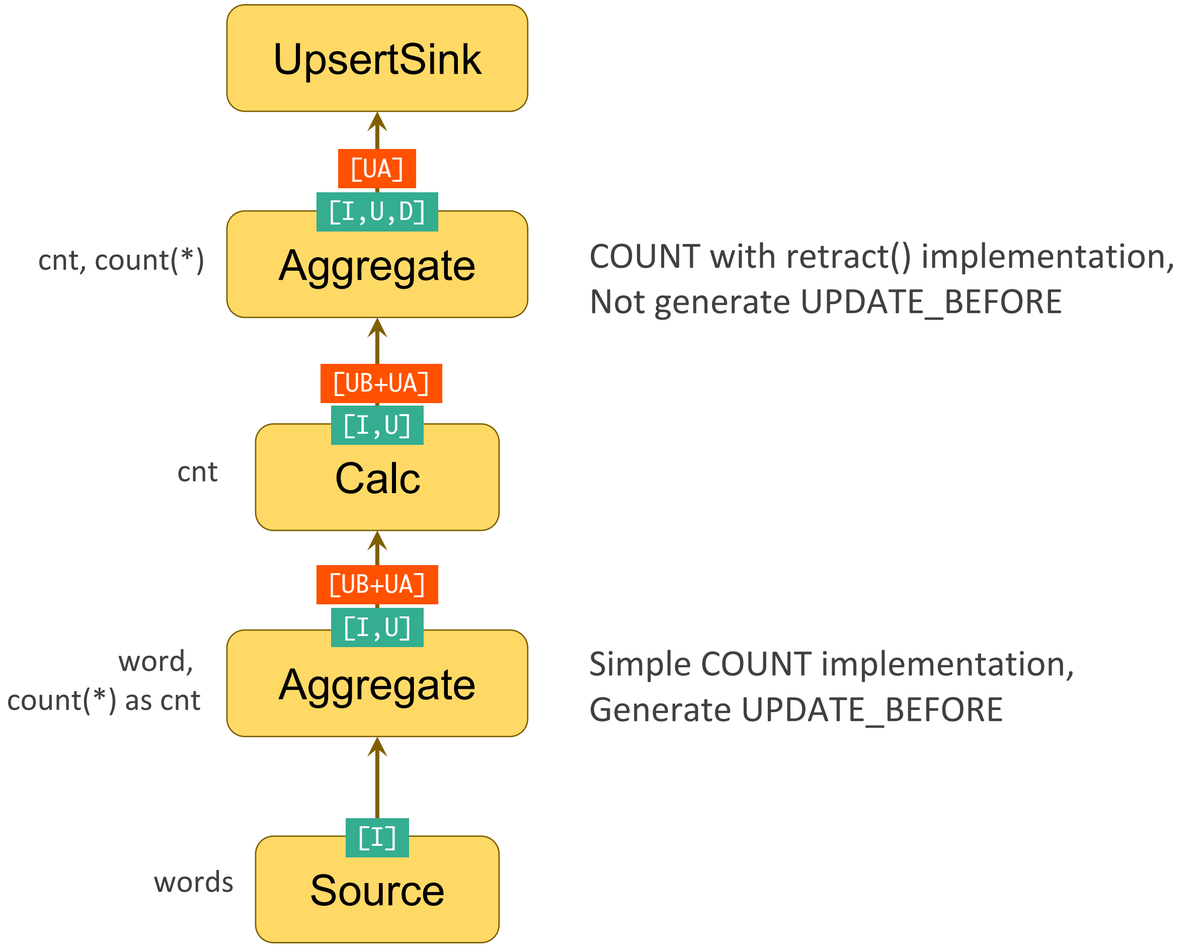
Physical Planning (Stream): Changelog Mechanism

Step2: determine how to produce updates

```
[UB]: Update_Before
[UA]: Update_After
```



Physical Planning (Stream): Changelog Mechanism



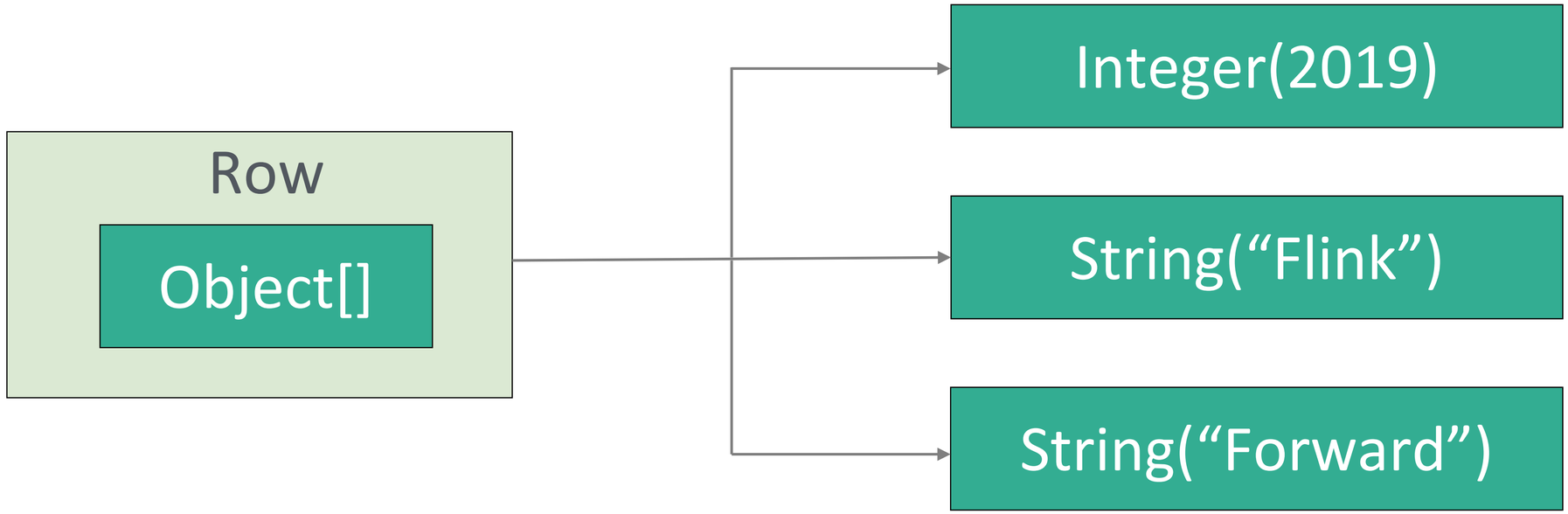
Final Physical Plan

Flink SQL Optimizations

- Internal Data Structure (BinaryRow)
- Mini-Batch Processing
- Aggregation Skew Handling
- Plan Rewrite

Old Planner: Row

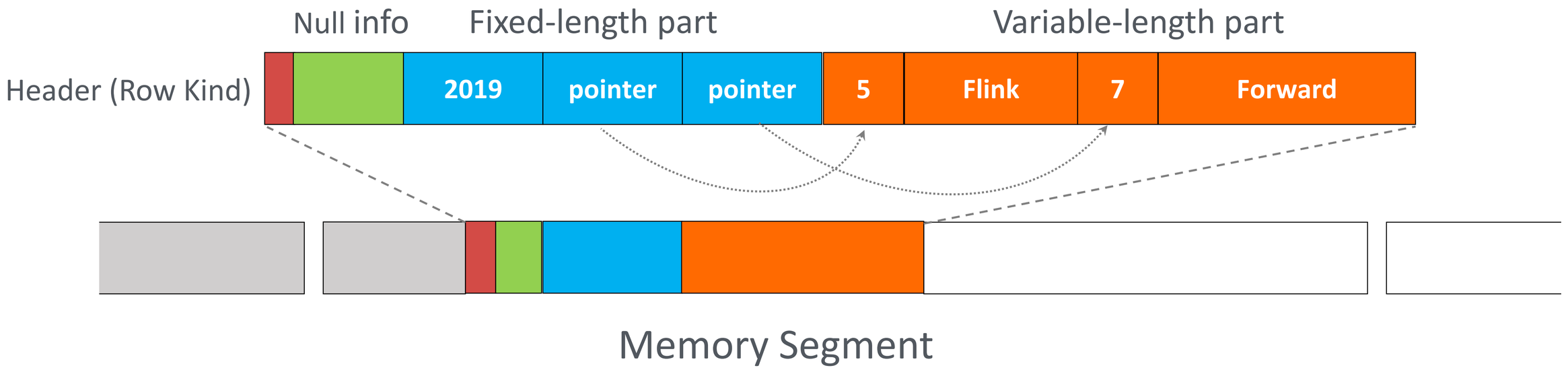
- Row(2019, “Flink”, “Forward”)



- Space inefficiency (object header)
- Boxing and unboxing
- Serialization and deserialization cost, especially when we want to access fields randomly

New Blink Planner: BinaryRow

- Deeply integrated with MemorySegment
- No need to deserialize / Compact layout / Random accessible
- Also have BinaryString, BinaryArray, BinaryMap

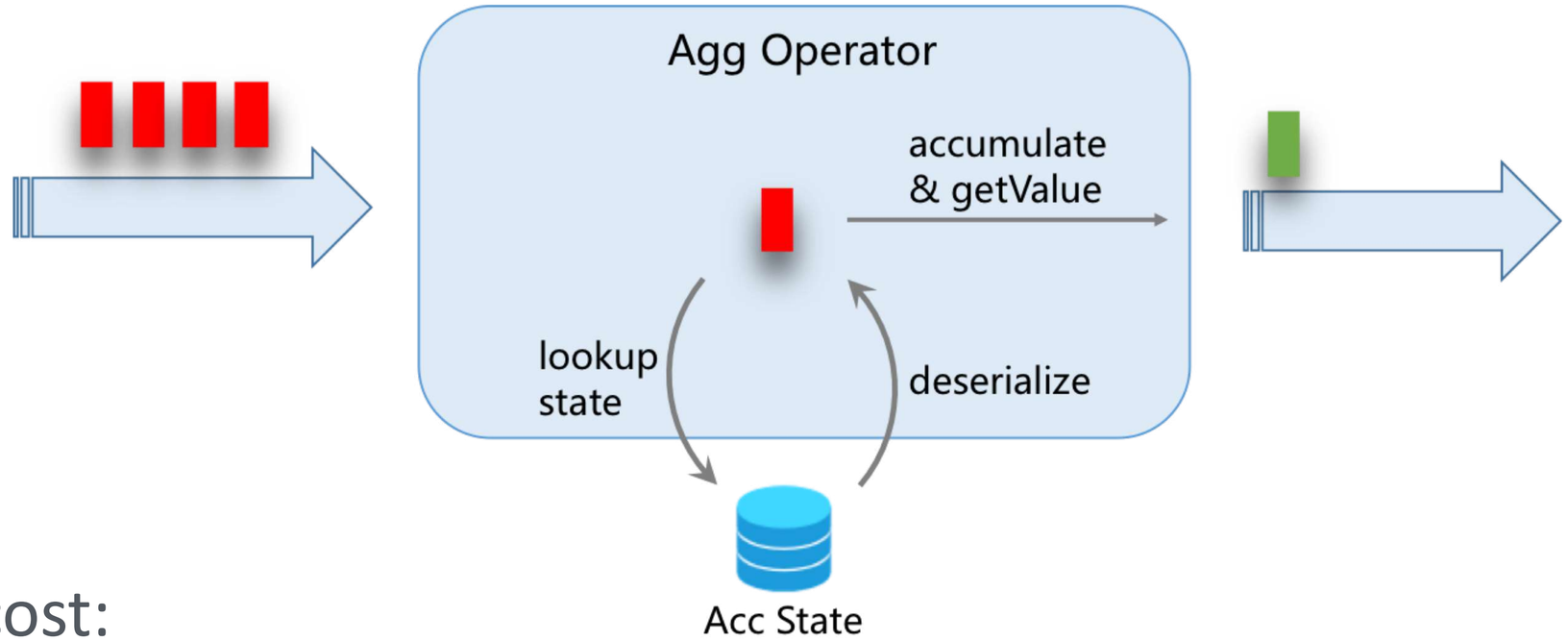


Blink planner is +54.6% than old planner when object reuse is enabled:
https://www.ververica.com/blog/a-journey-to-beating-flinks-sql-performance

Mini-Batch Processing


```
SELECT SUM(num) FROM T GROUP BY color
```

Normal aggregation:

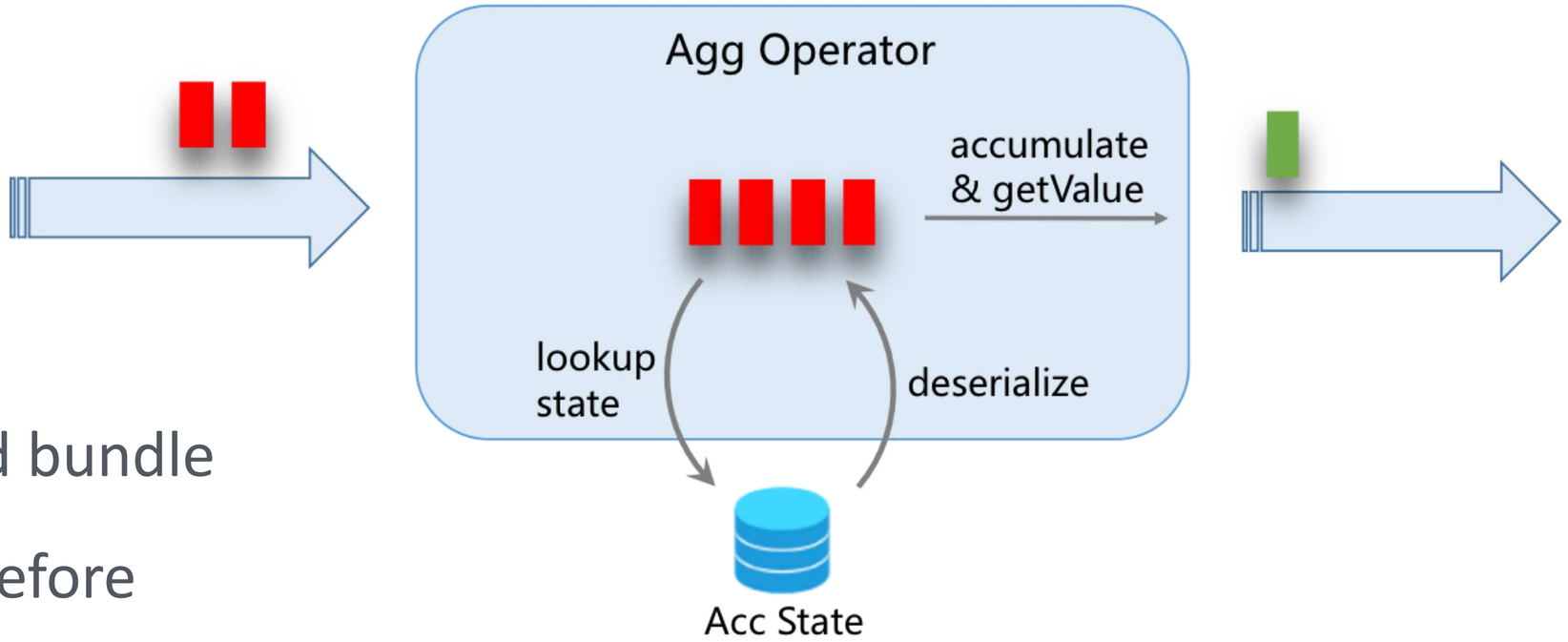


- Each record would cost:
  - One state reading and writing
  - One deserialization and serialization
  - One output

Mini-Batch Processing


```
SELECT SUM(num) FROM T GROUP BY color
```

Mini-Batch aggregation:



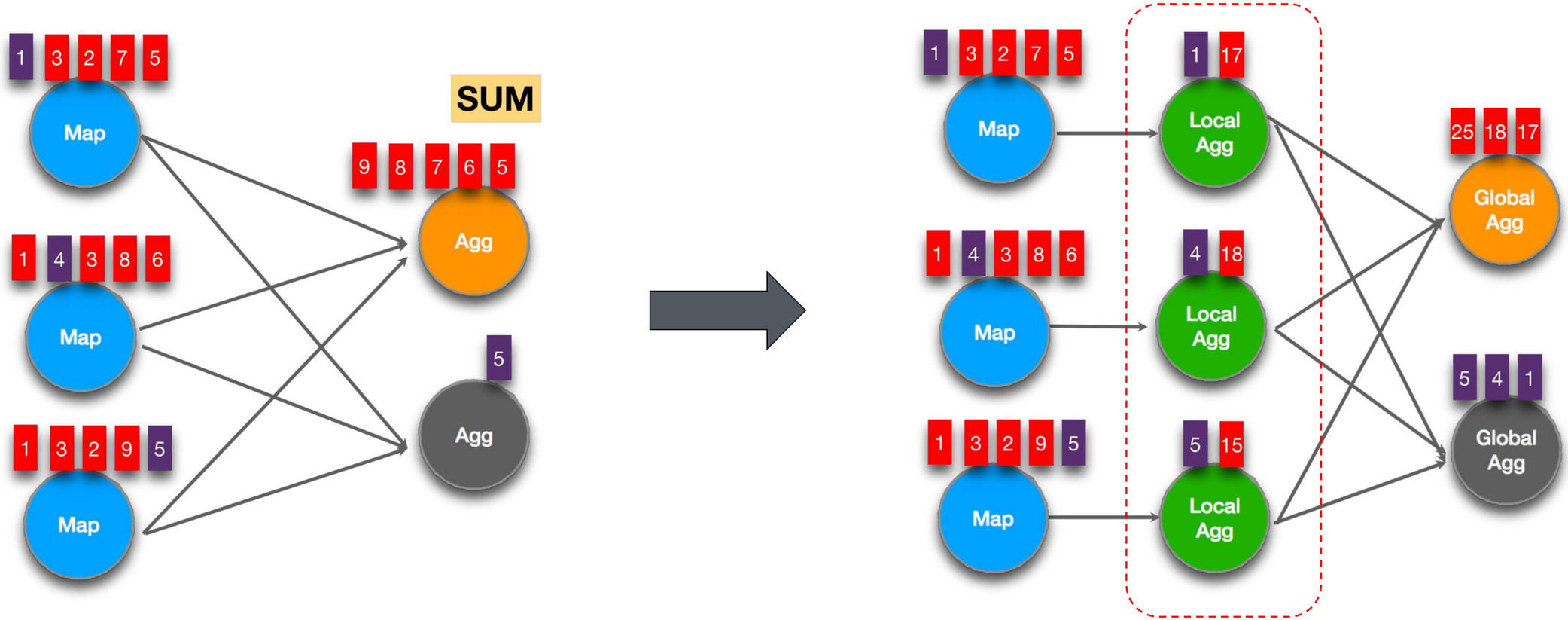
- Use heap memory to hold bundle
- In-memory aggregation before accessing states and serde operations
- Also ease the downstream loads

```
table.exec.mini-batch.enabled = true
table.exec.mini-batch.allow-latency = "5000 ms"
table.exec.mini-batch.size = 1000
```

Aggregation Skew Handling


```
SELECT SUM(num) FROM T GROUP BY color
```

```
table.optimizer.agg-phase-strategy = TWO_PHASE
```

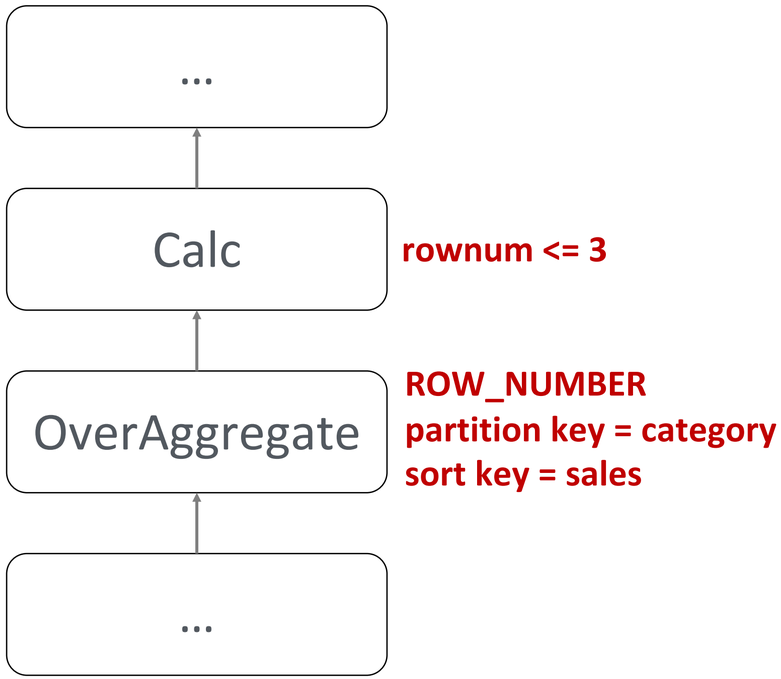


Plan Rewrite (Top-N)

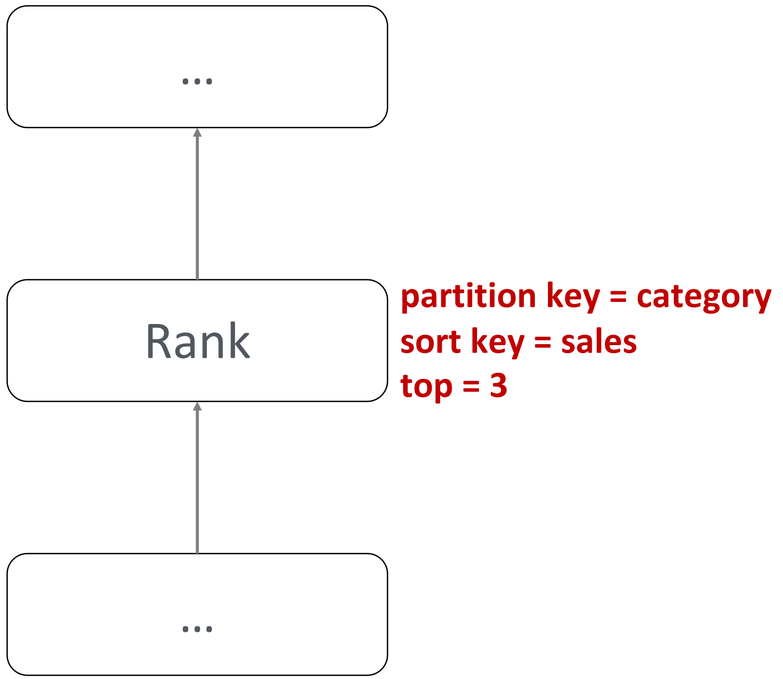
- It’s impractical to do a global streaming sort
- But it becomes possible if user only cares about the top n elements
- E.g. Calculate the top 3 shops for each category

```
SELECT *
FROM (
  SELECT *, // you can get like shopId or other information from this
    ROW_NUMBER() OVER (PARTITION BY category ORDER BY sales DESC) AS rowNum
  FROM shop_sales)
WHERE rowNum <= 3
```

Plan Rewrite (Top-N)



Original Plan



Optimized Plan

Summary & Futures

Summary & Futures

- Flink took a big step towards truly unified architecture
- Introduced how Flink SQL works step by step.
- Flink SQL does a lot of optimizations for users automatically
- Future (Flink 1.11+)
  - Blink planner will be the default planner and ready for production
  - New TableSource and TableSink interfaces (FLIP-95)
  - Support to read changelogs (FLIP-105)
  - Unified Batch and Streaming Filesystem connector (FLIP-115)
  - Hive DDL & DML compatible (FLIP-123)

Alibaba Cloud | Worldwide Cloud Services Partner

Thank You!
Questions?

Alibaba Cloud |

Worldwide Cloud Services Partner