StarRocks Query Optimizer

Kaisen Kang

About Me

StarRocks

- CelerData Query Team Leader
- StarRocks PMC & Committer
- Apache Kylin PMC & Committer (Inactive)
- Apache Doris PMC & Committer (Retired)

StarRocks Query Optimizer

01

Background

2

Representative Optimizations

3

CBO Challenges

4

Learning Thinking

Development History of StarRocks

StarRocks

1.x Extremely Fast Query

- CBO
- Runtime Filter
- Vectorized Engine
- Open Source

2.x Fusion and Unity

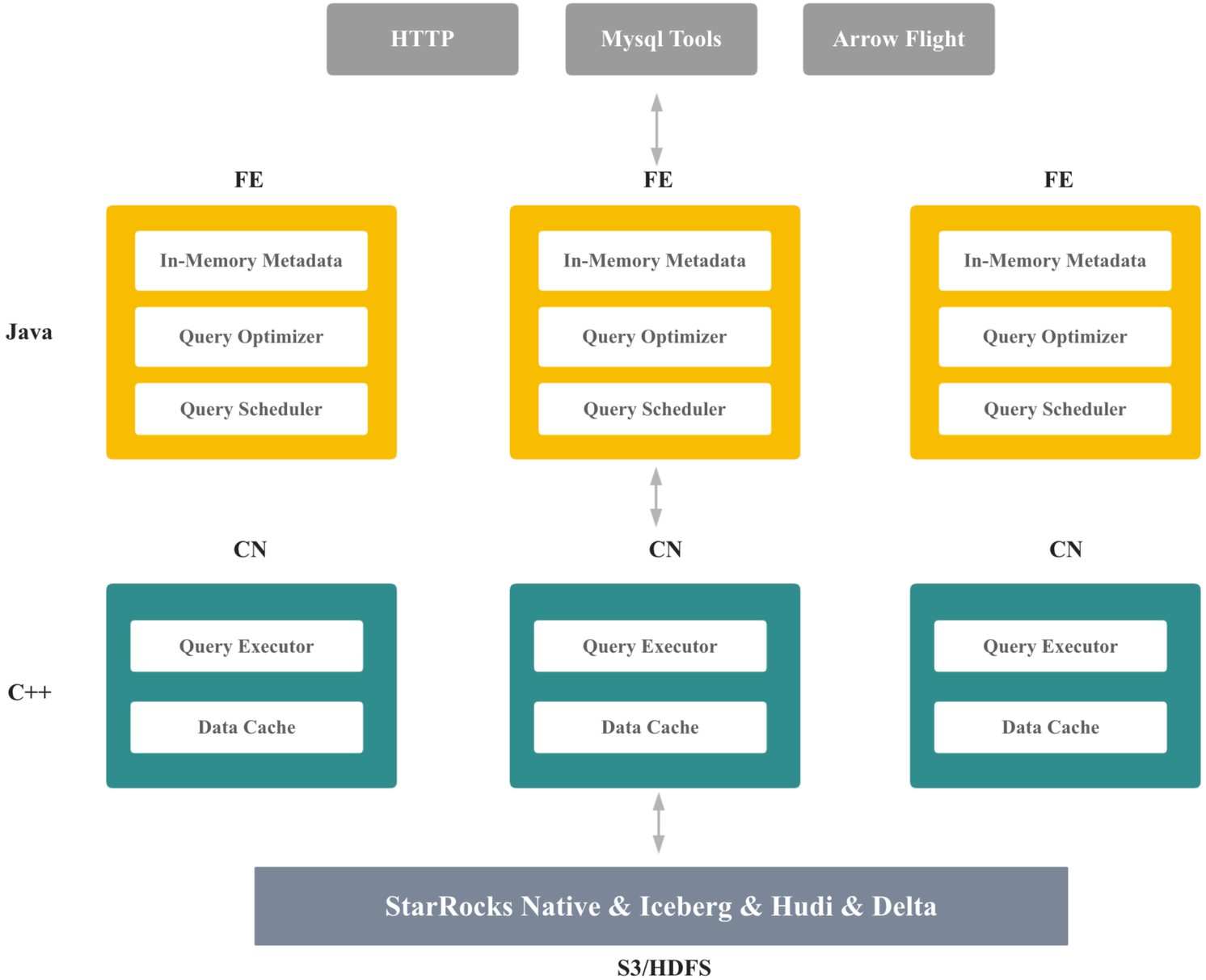
- Pipeline Engine
- Query Cache
- DataLake Analytic
- Resource Isolation
- Multi Table MV
- Global Dict
- Unified Catalog
- Primary Key Model

3.x LakeHouse

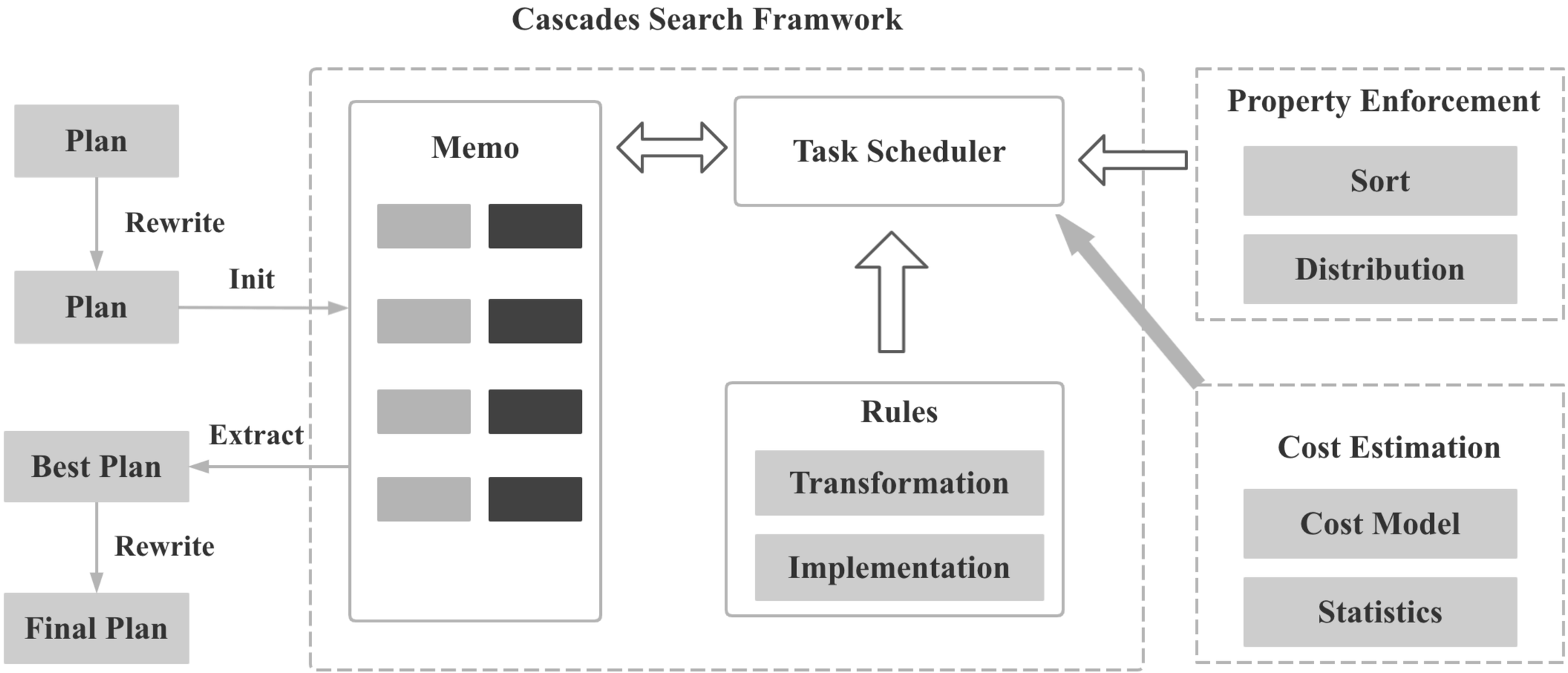
- Shared Data
- Semi Structured
- MV For Lake
- Full Text Search
- Data Lake Write
- Spill To Disk
- Flat Json
- Vector Search

From 2020 To 2025

StarRocks Architecture

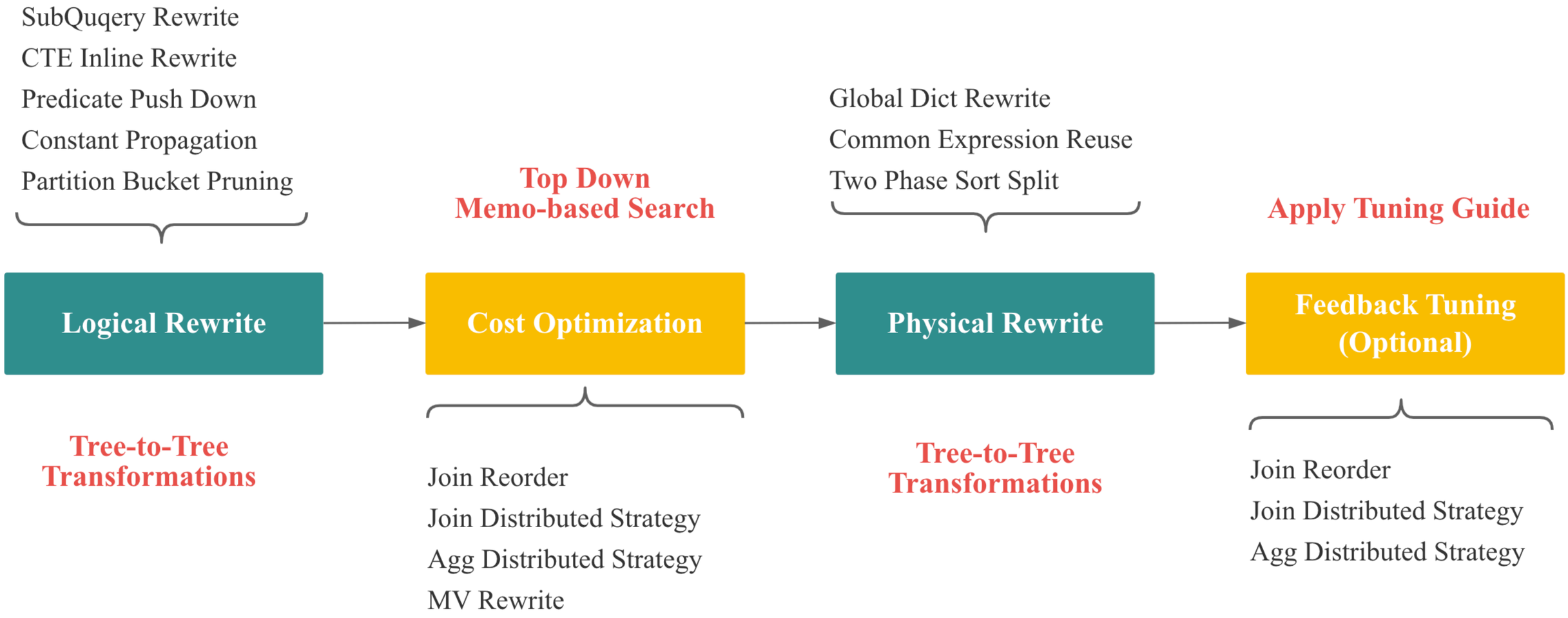


StarRocks Optimizer Overview 1



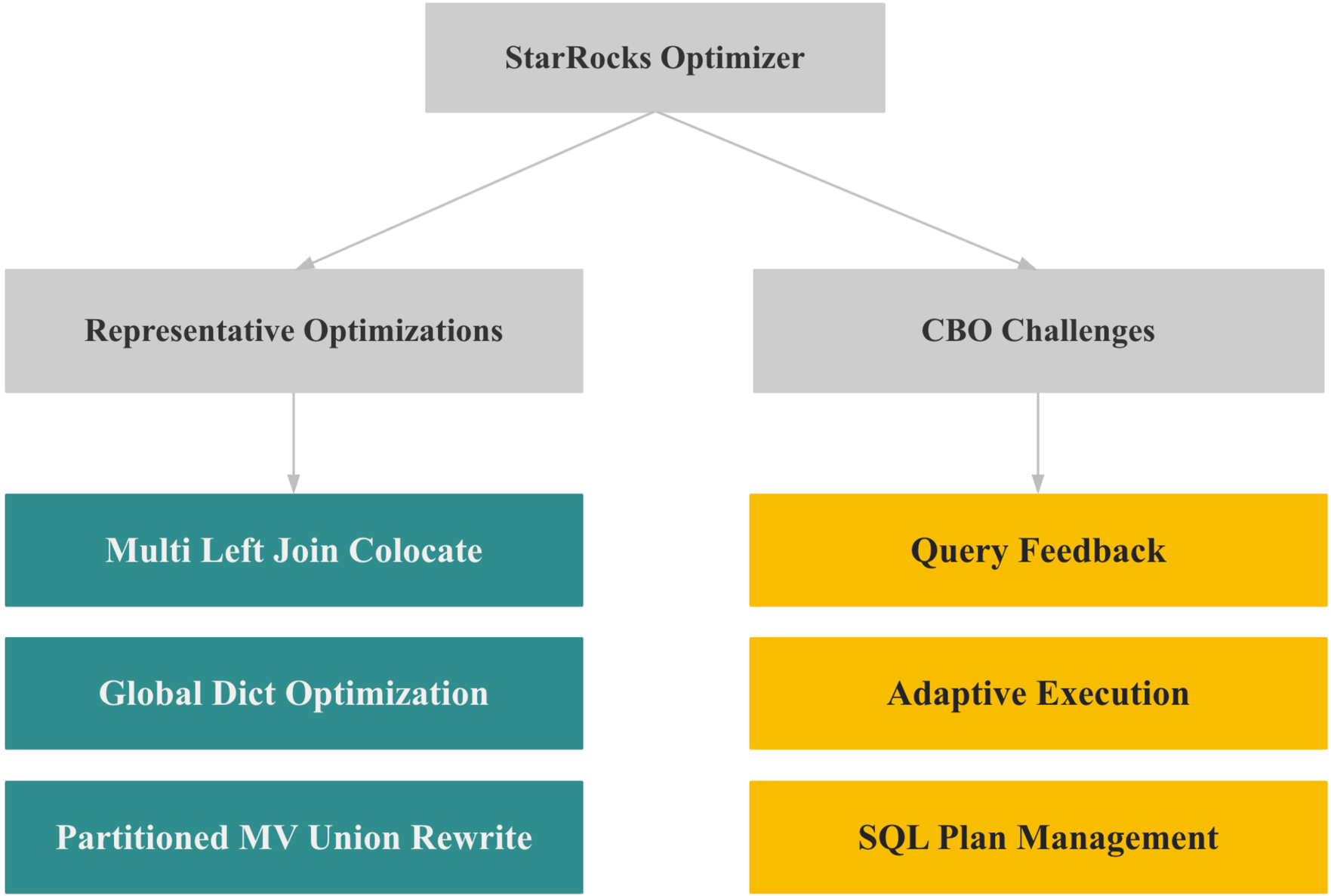
Based on the Cascades and Columbia Papers

StarRocks Optimizer Overview 2

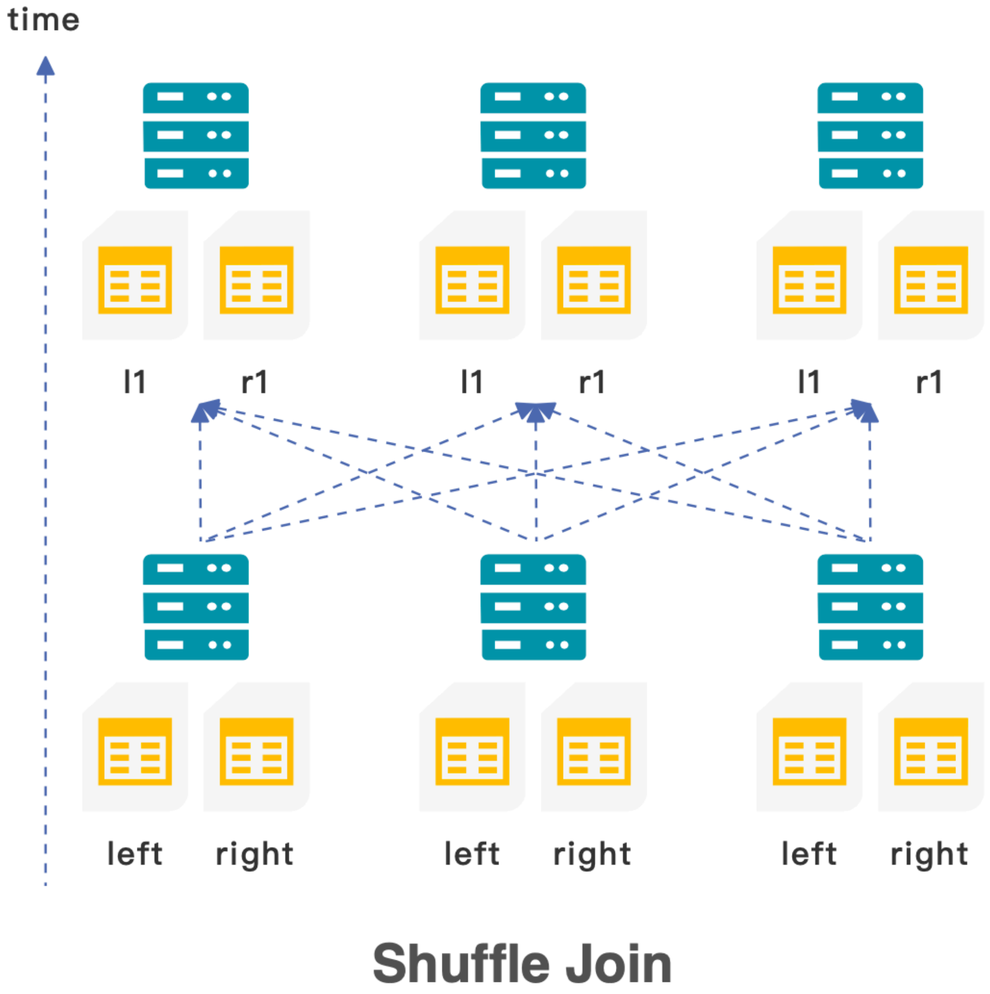


StarRocks Optimizer Introduction

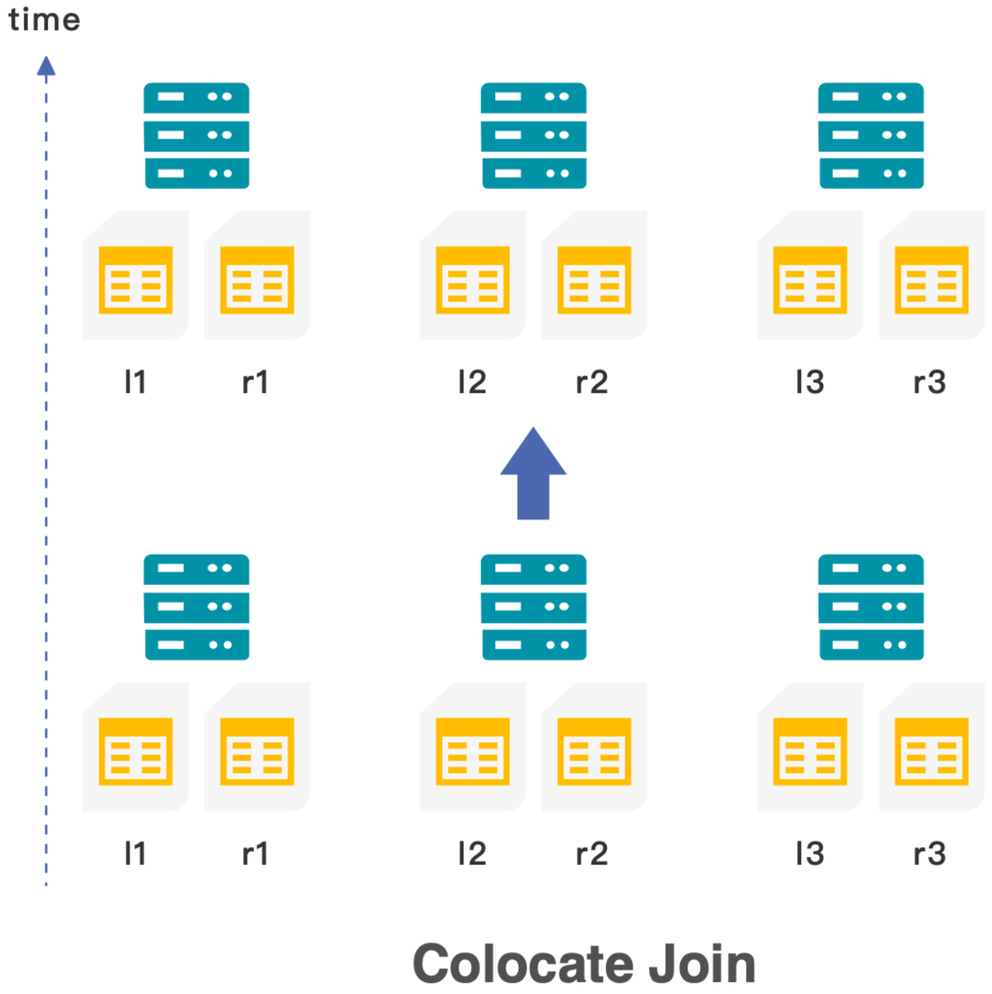
StarRocks



StarRocks Colocate Join



Shuffle Join

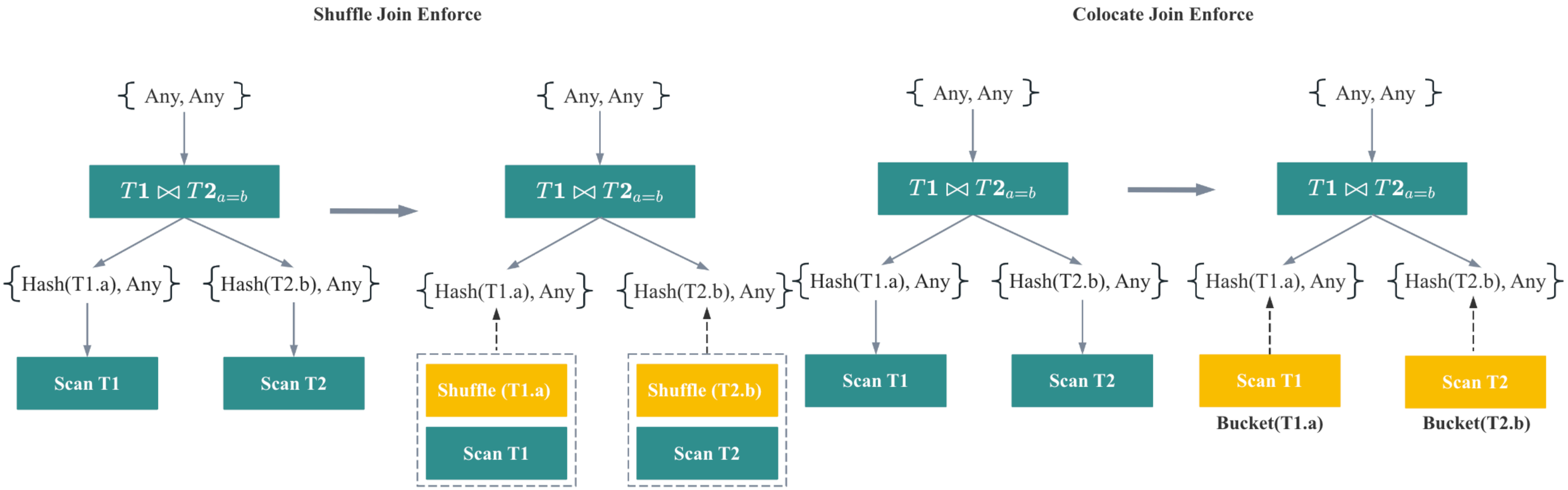


Colocate Join

Enforce For Shuffle and Colocate

StarRocks

SELECT * FROM T1 INNER JOIN T2 ON a = b



T1 and T2 are normal tables

T1.a Colocate With T2.b

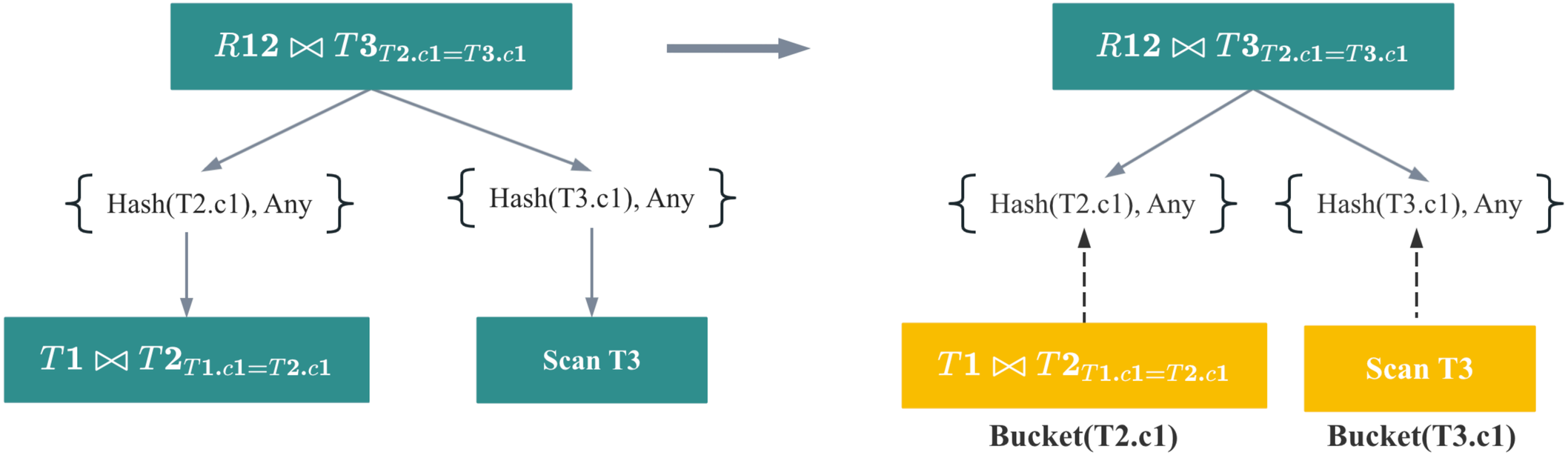
Scan satisfy the required property

Multi Inner Join Colocate

StarRocks

SELECT * FROM T1 JOIN T2 ON t1.c1 = t2.c1 Join T3 On T2.c1 = T3.c1

Multi Inner Join Colocate Enforce



T1.c1, T2.c1, T3.c1 are Colocated

Inner Join won't procude NULL
So Multi Inner Join Colocate is same as One Inner Join

Left Outer Join will Produce NULL

StarRocks

SELECT * FROM T1 Left JOIN T2 ON T1.c1 = T2.c1 Left Join T3 On T2.c1 = T3.c1

T1

C1	C2
1	a
2	b

T2

C1	C2
1	a
3	b

T3

C1	C2
1	a
3	b

T1 Left Join T2

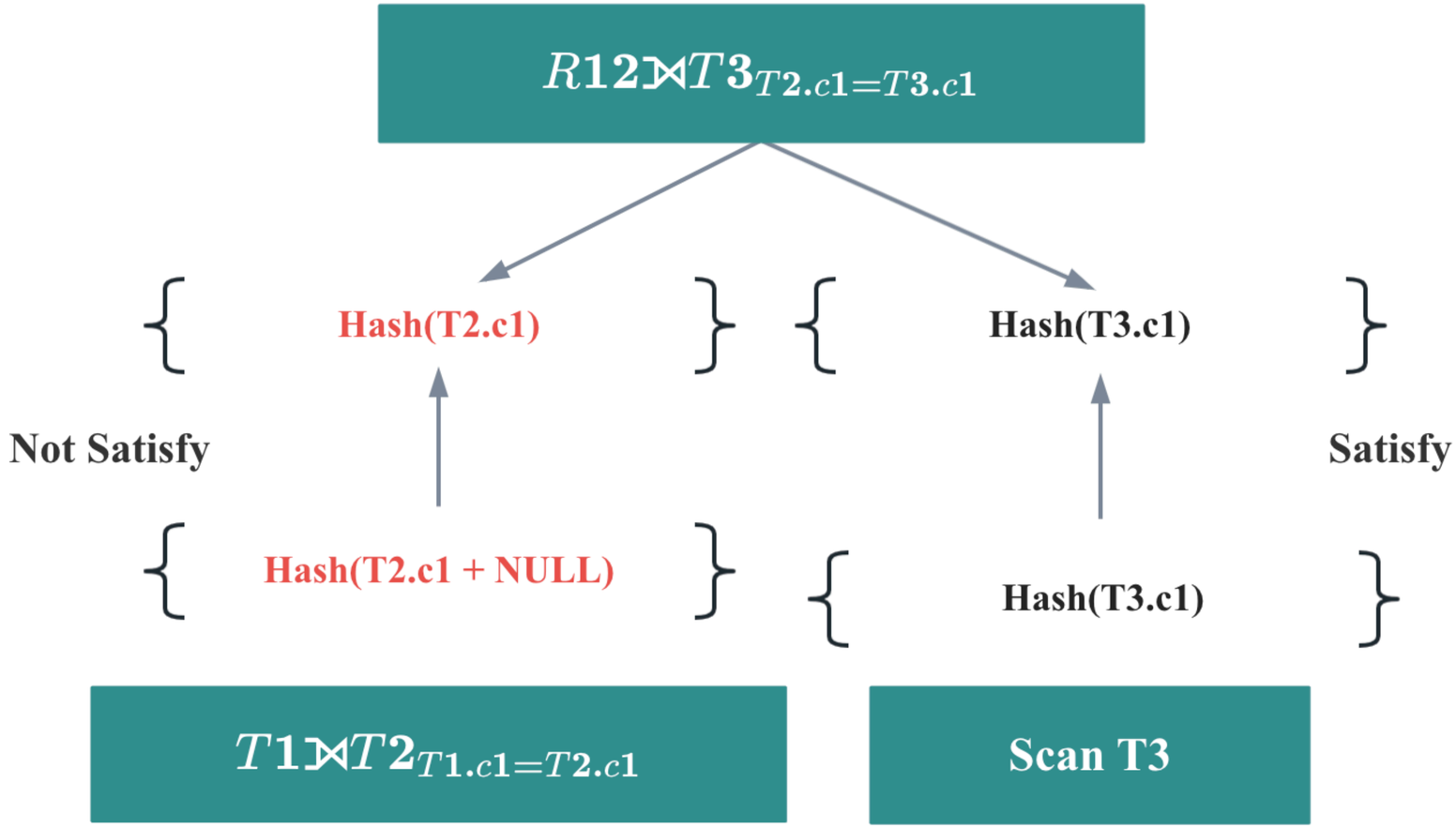
T1.C1	T2.C1
1	1
2	NULL

The right table of left outer Join will produce NULL

Left Outer Join will Produce NULL

StarRocks

SELECT * FROM T1 Left JOIN T2 ON T1.c1 = T2.c1 Left Join T3 On T2.c1 = T3.c1



The NULL generated by left join makes property enforce unable to satisfy

The Result of Colocate NULL and Shuffle NULL are the same

StarRocks

SELECT * FROM T1 Left JOIN T2 ON T1.c1 = T2.c1 Left Join T3 On T2.c1 = T3.c1

T1

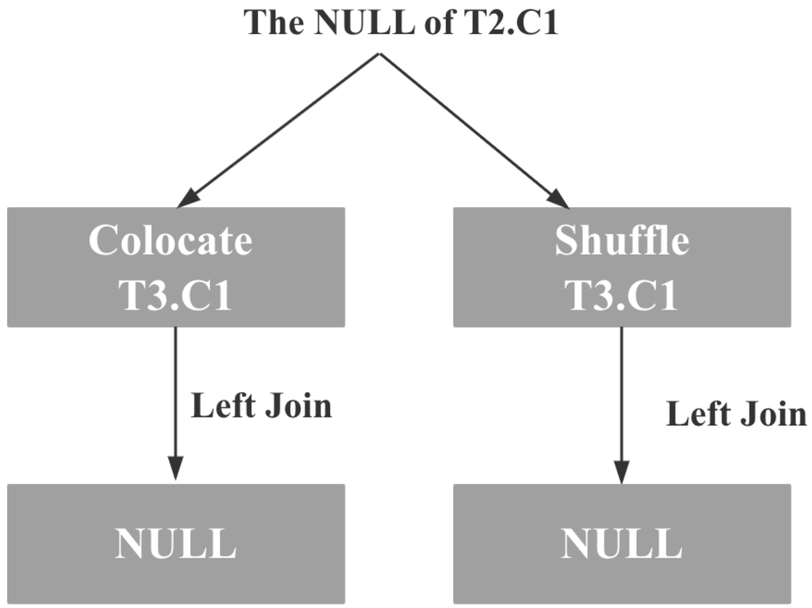
C1	C2
1	a
2	b

T2

C1	C2
1	a
3	b

T3

C1	C2
1	a
3	b



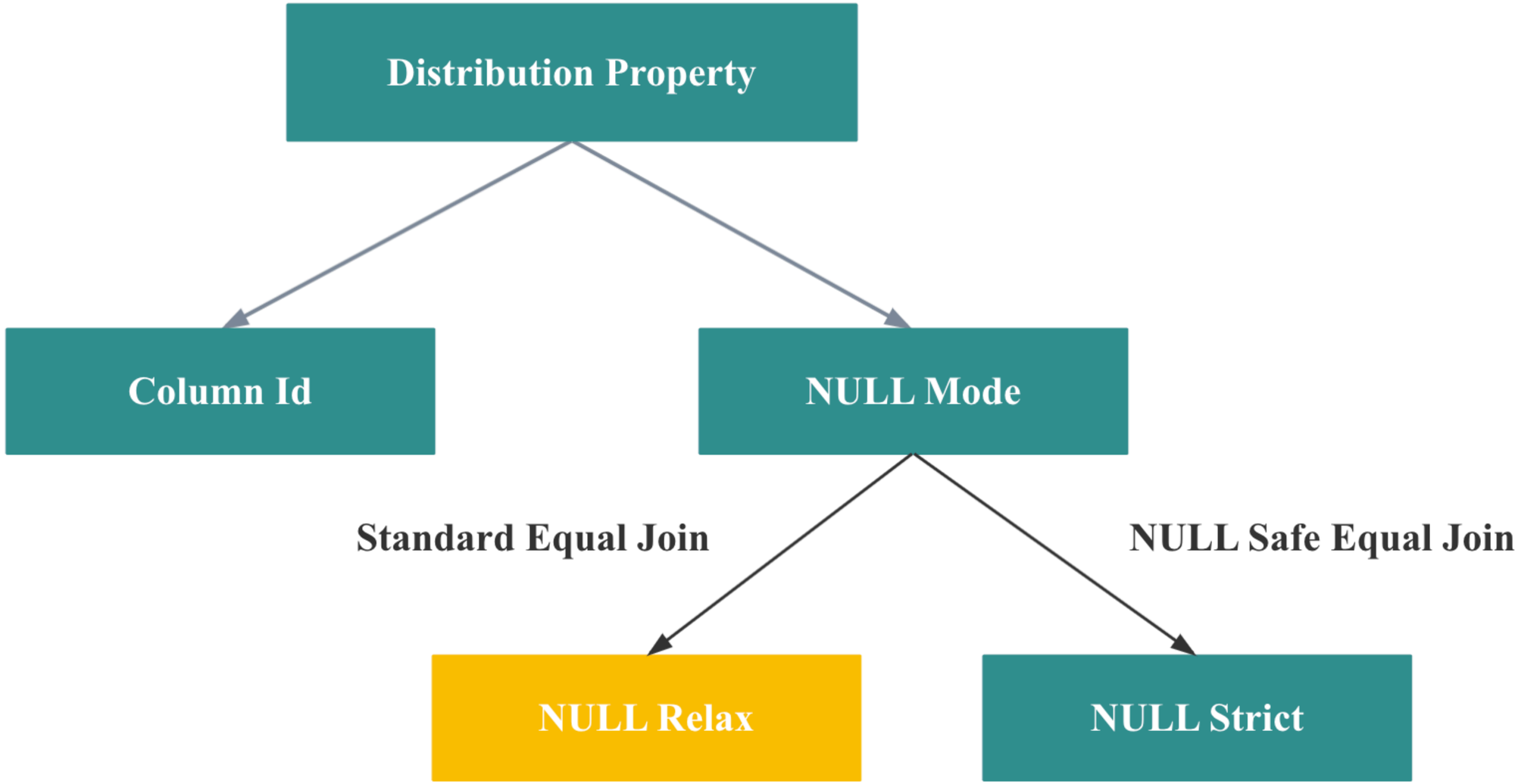
The Result Of
T1 Left JOIN T2 ON T1.c1 = T2.c1
Left Join T3 On T2.c1 = T3.c1

T1.C1	T1.C2	T2.C1	T2.C2	T3.C1	T3.C2
1	a	1	a	1	a
2	b	NULL	NULL	NULL	NULL

The NULL distribution of T2 does not affect the final result
So T2 and T3 still could do Colocate Join
We could add Null strict and Null relax mode to Distribution Property

Add Null strict and Null relax mode in DistributionProperty

StarRocks

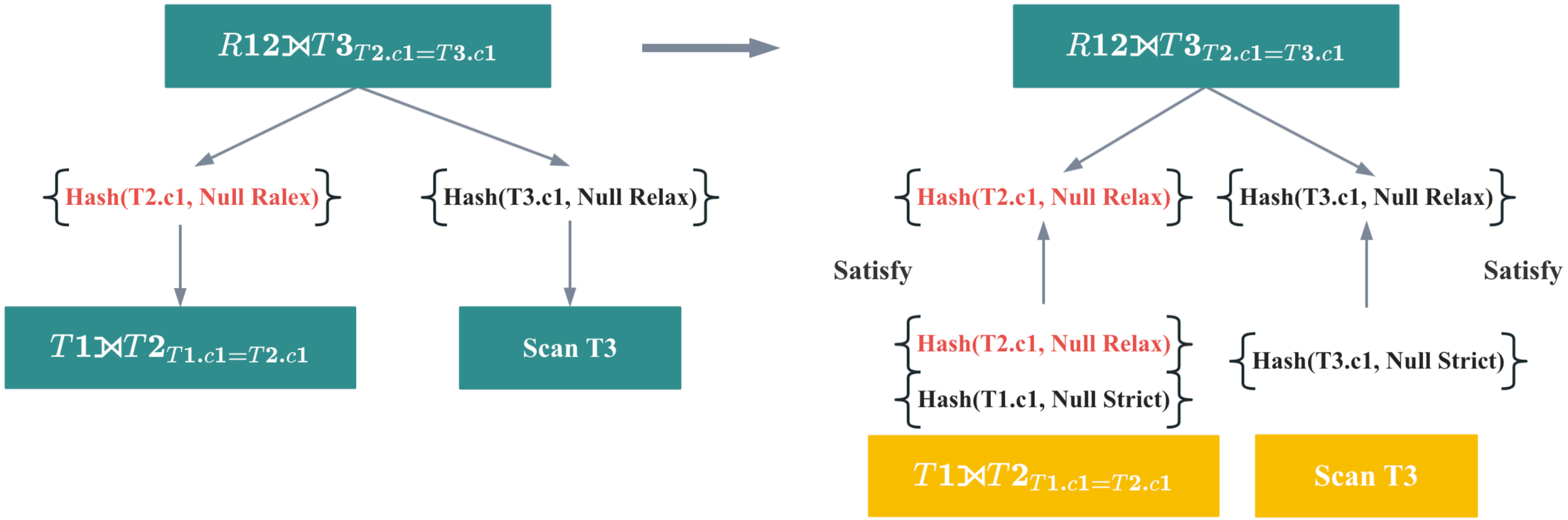


Add Null strict and Null relax mode in DistributionProperty

StarRocks

SELECT * FROM T1 Left JOIN T2 ON T1.c1 = T2.c1 Left Join T3 On T2.c1 = T3.c1

Multi Left Join Colocate Enforce

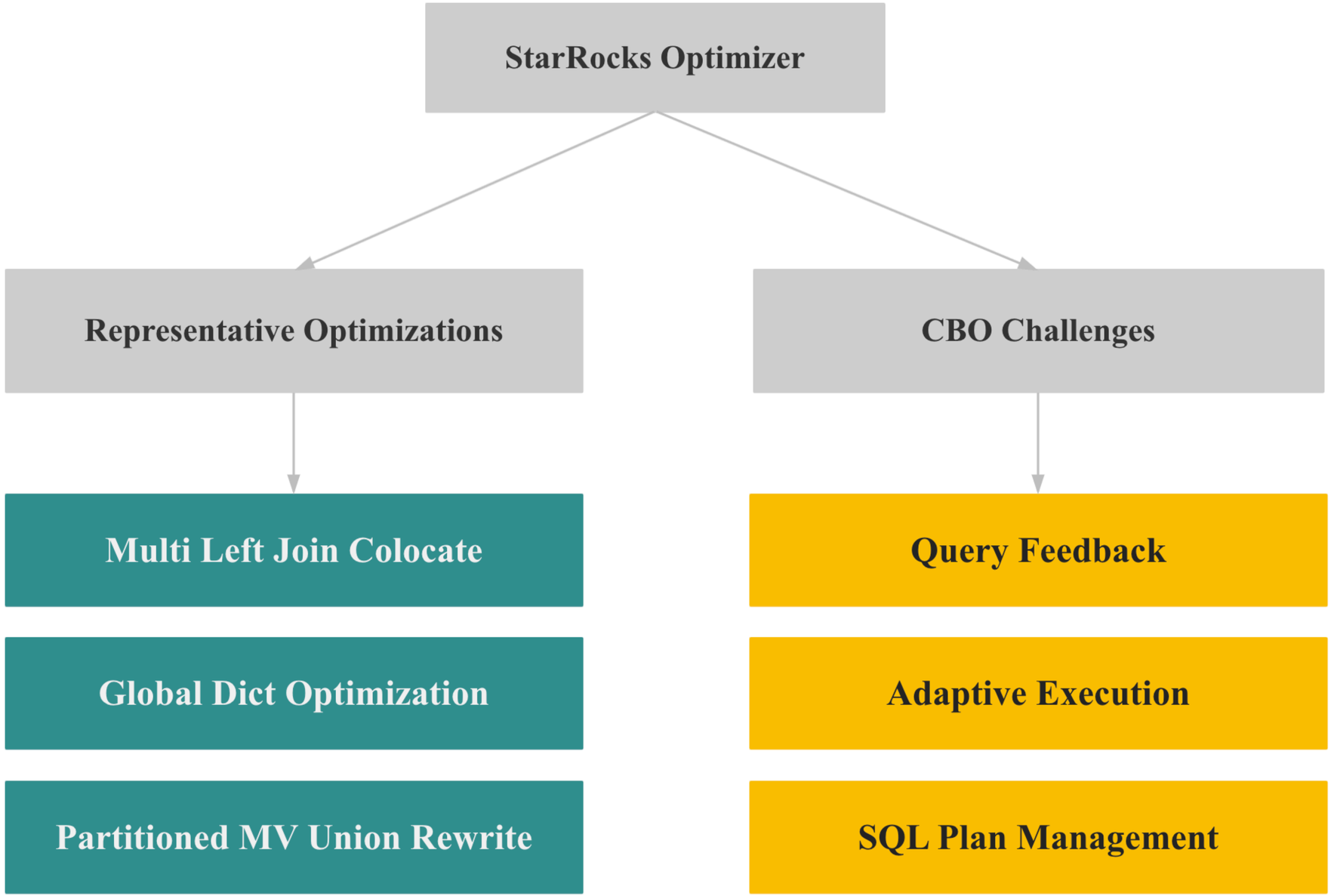


T1.c1, T2.c1, T3.c1 are Colocated

For the dirstribution column,
when Null relax mode, could ignore NULL value when enforce

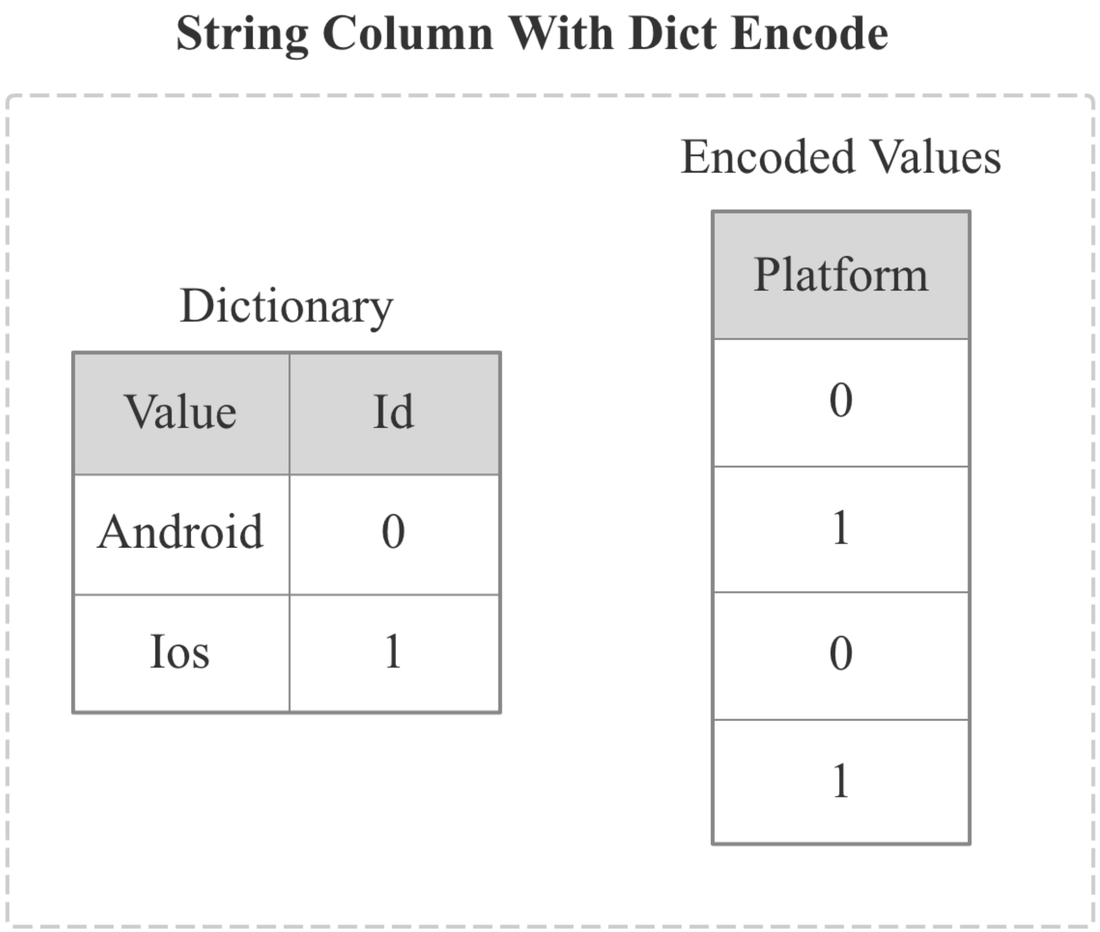
StarRocks Optimizer Introduction

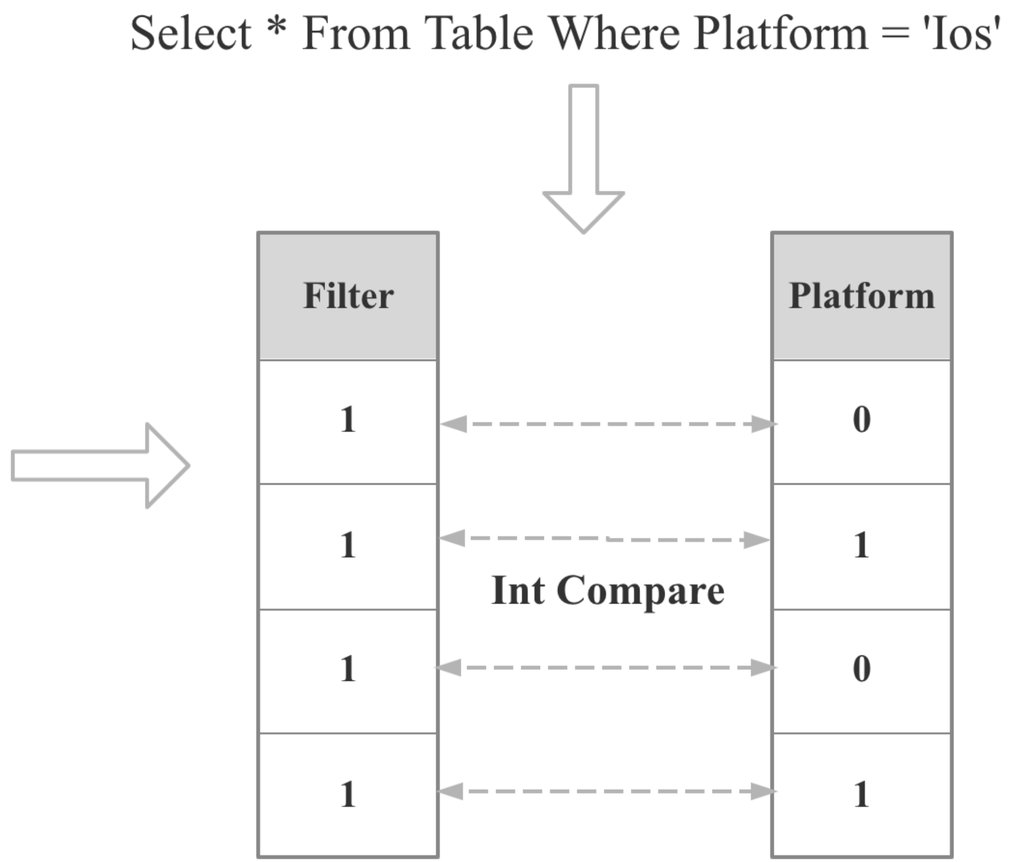
StarRocks



Dict Optimization (Operations on Encoded Data)

StarRocks





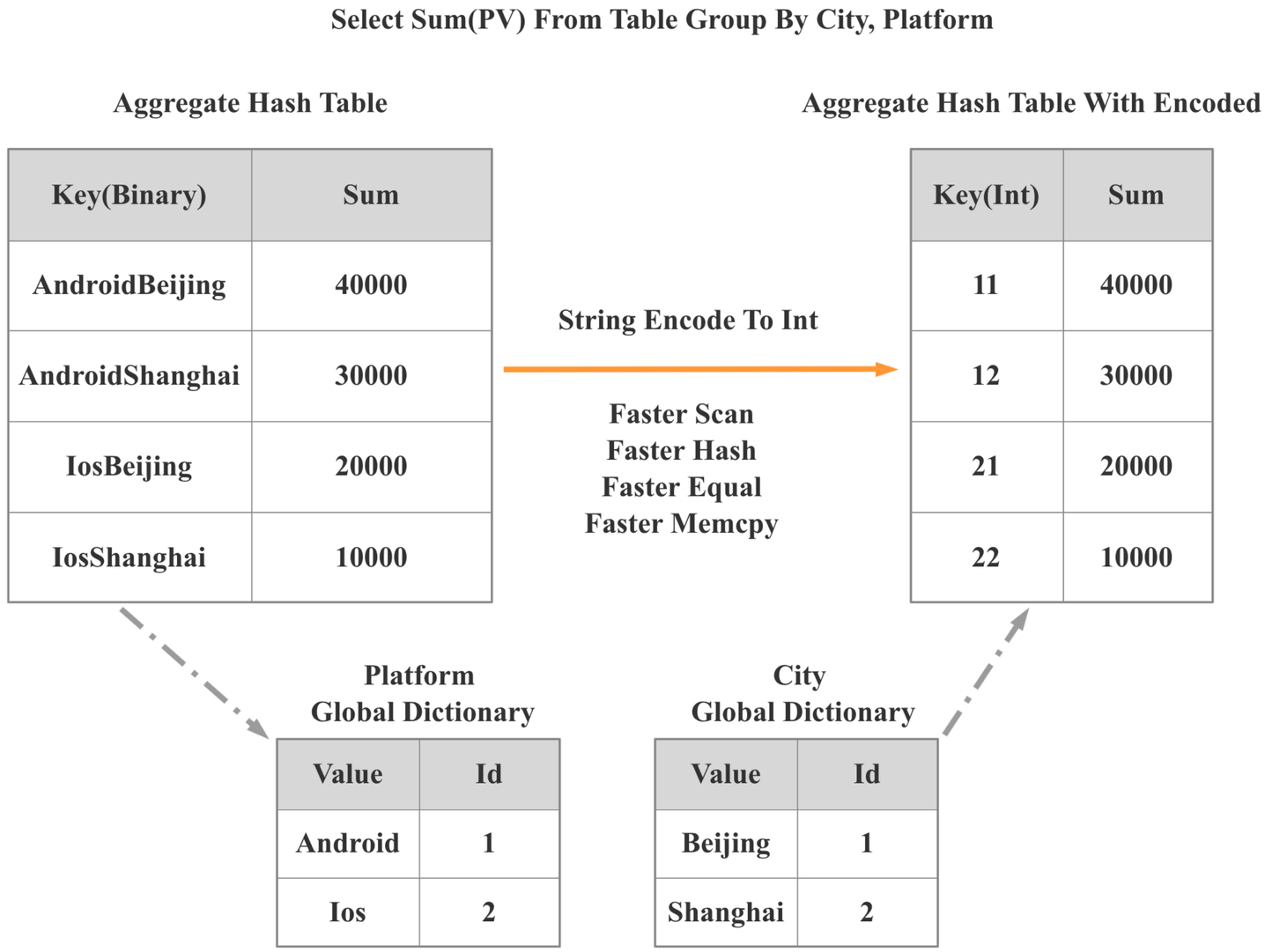
Local Dict in Storage Engine

Int Compare is Very Faster Than String

StarRocks Low Cardinality Global Dict Optimization

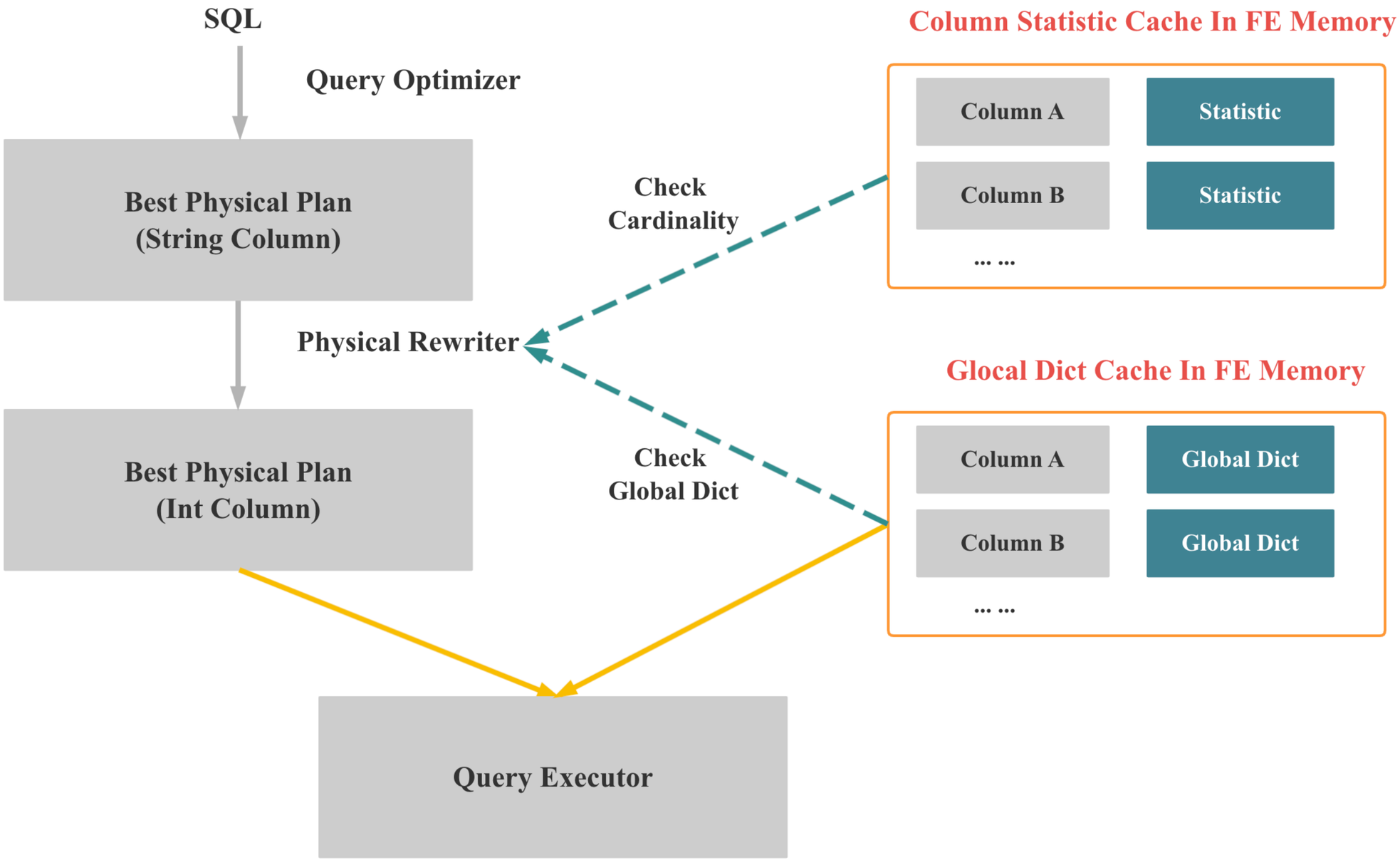
StarRocks

- Scan
- Filter
- Agg
- Sort
- Join
- String Functions



3X Performance Improvement For Aggregate

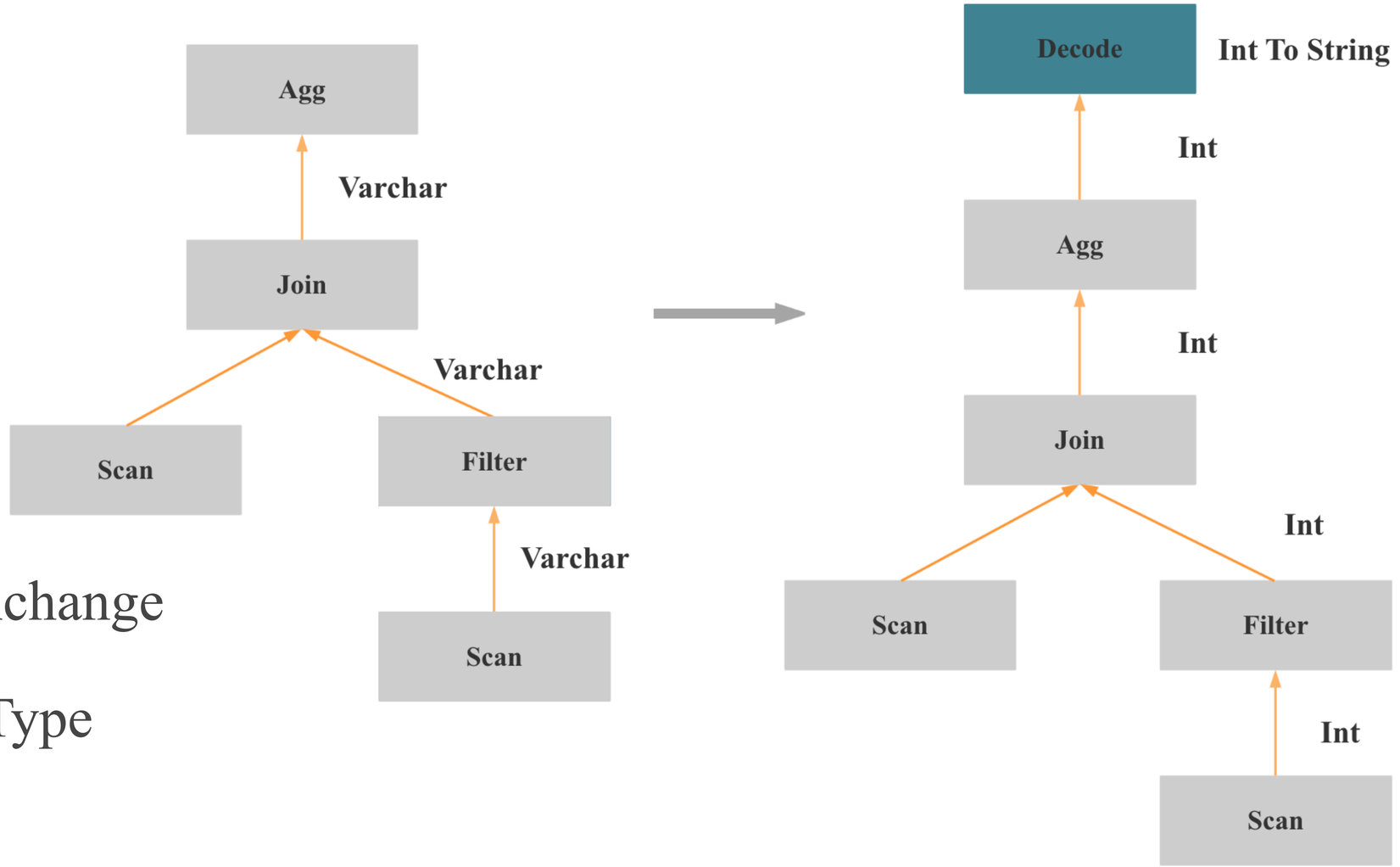
Low Cardinality Global Dict Rewrite Overview



Low Cardinality Global Dict Rewrite

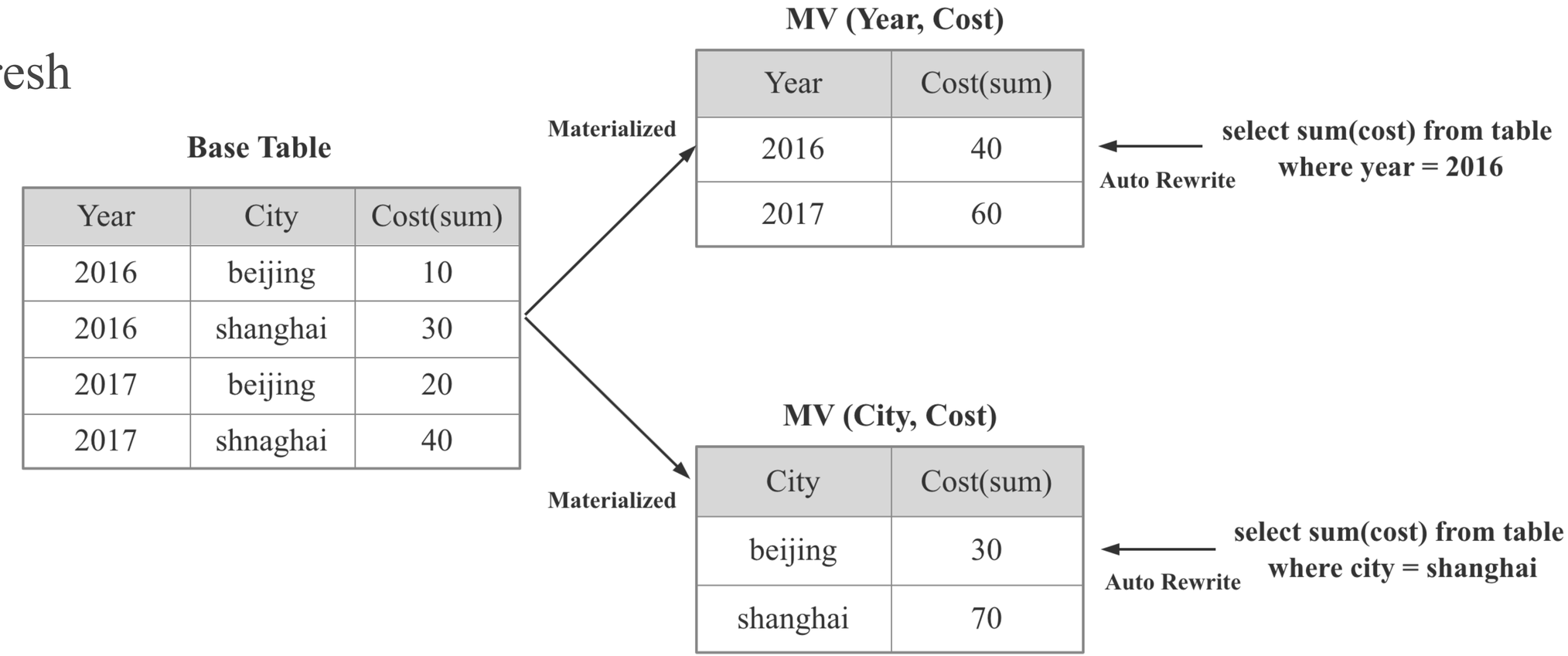
StarRocks

- Tree Based Rewrite
- Add Decode Node
- Attach Global Dict to Exchange
- String and Array String Type



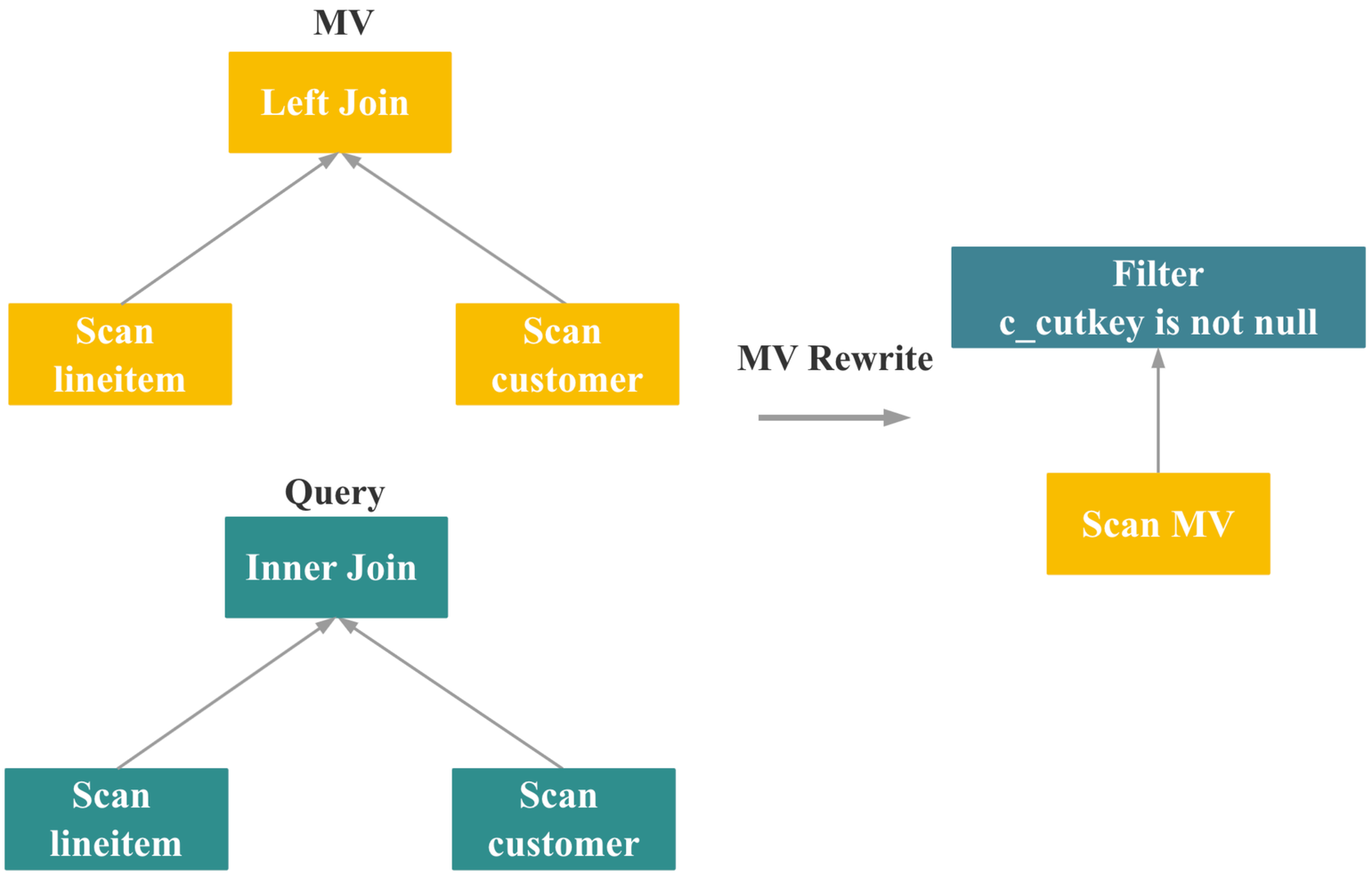
StarRocks MV

- Async and Sync
- Single Table and Multi Tables
- Auto Rewrite: No need to change the query
- Partition Refresh
- MV TTL



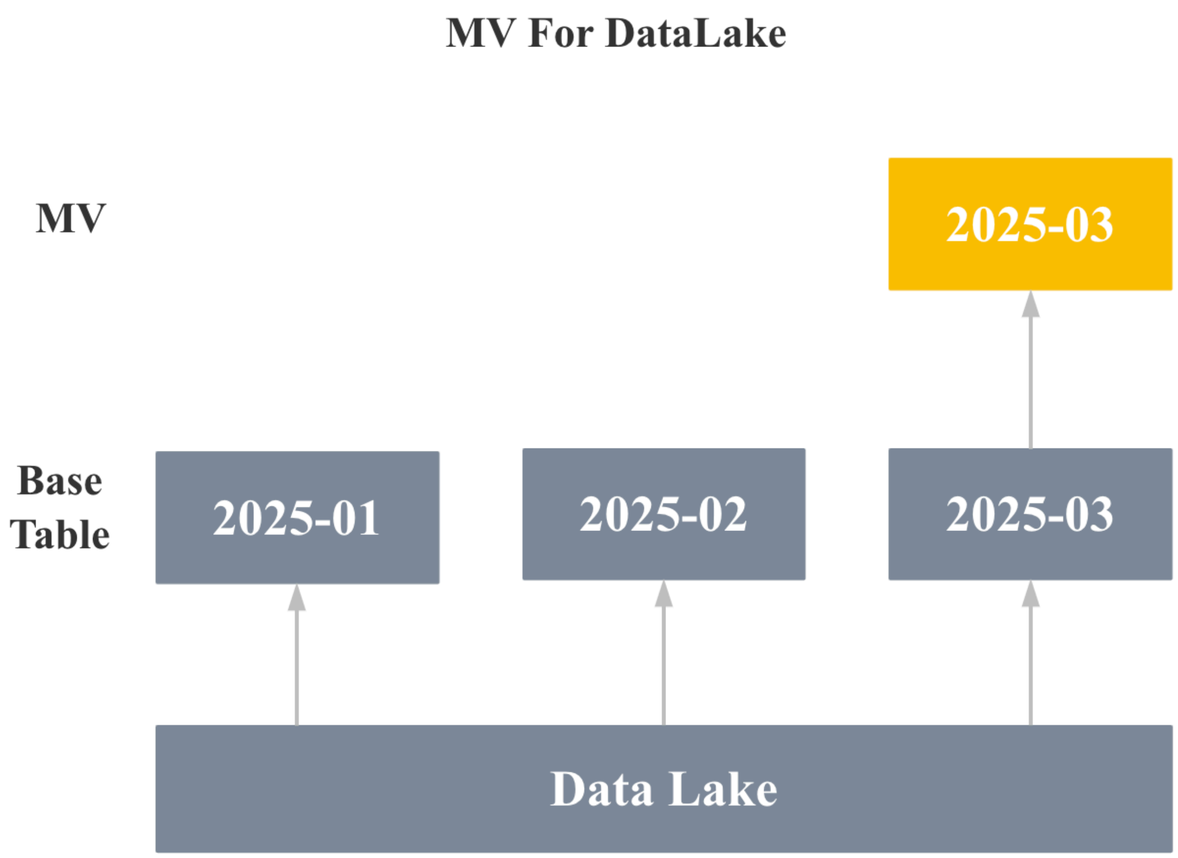
StarRocks MV Auto Rewrite

- *Agg Rewrite*
- *Join Rewrite*
- *Union Rewrite*
- *Nested MV Rewrite*
- *Text-Based Rewrite*
- *View-Based Rewrite*



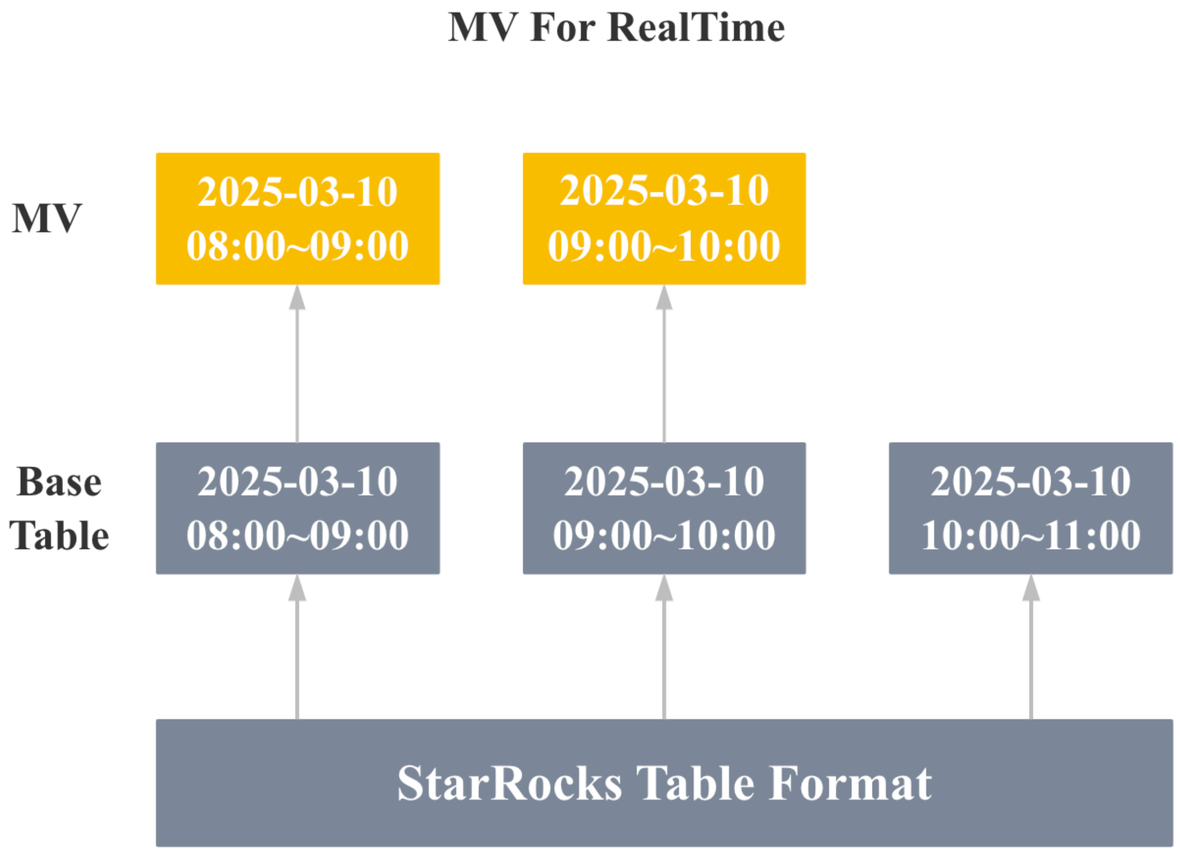
Why Need Auto MV Union Rewrite

StarRocks



MV speed up the recent data of Data Lake

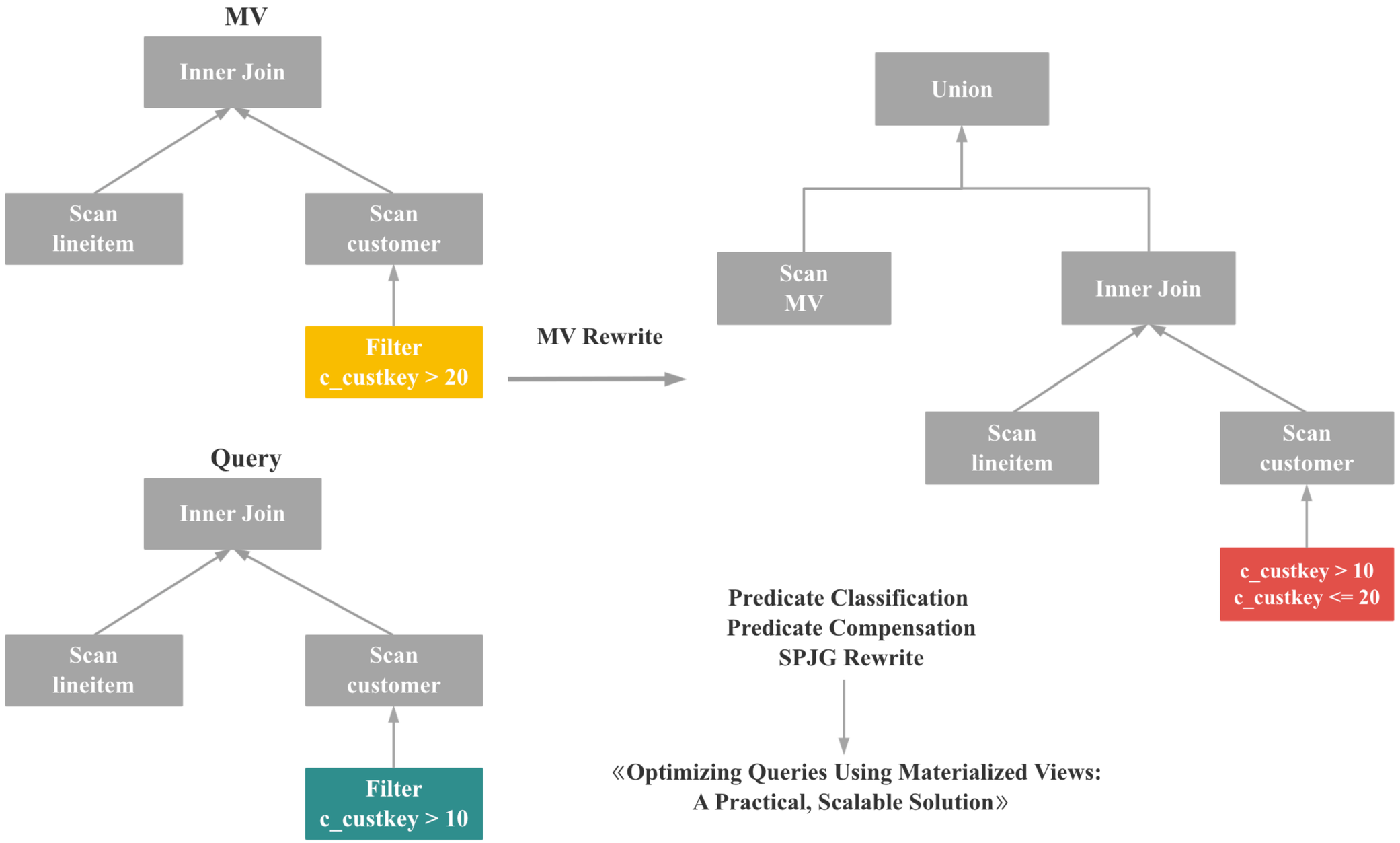
We need to union the MV and Historical Table Data



MV speed up the historical data

We need to union the MV and RealTime Table Data

MV Union Rewrite 1: MV is up-to-date



MV Union Rewrite 2: Partial Partitions Refreshed

StarRocks

MV:

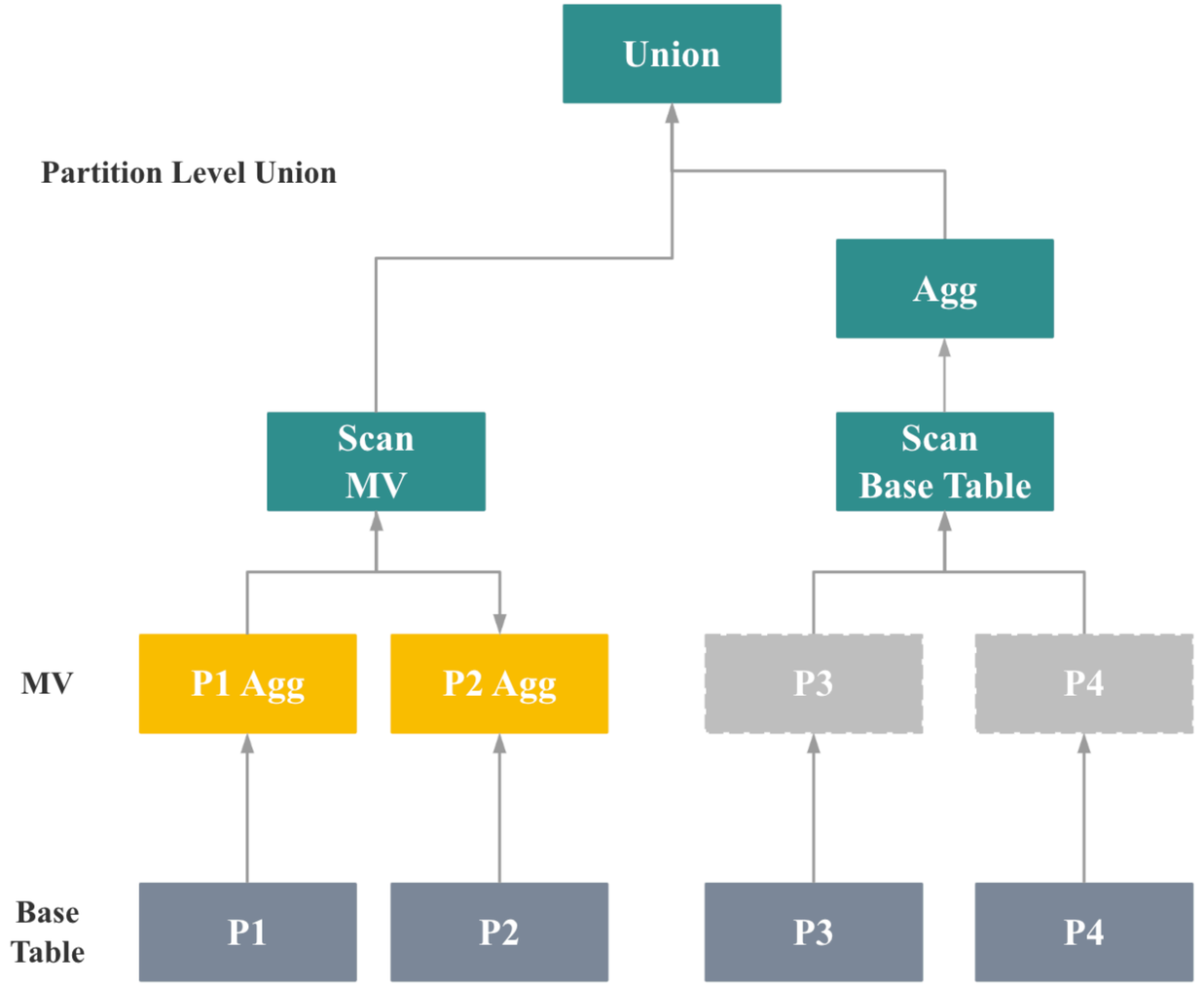
```
select count(*) from t group by a
```

Query:

```
select count(*) from t group by a
```

MV and Base Table have four partitions:

P1	P2
Refreshed	Refreshed
P3	P4
To Refresh	To Refresh



MV Union Rewrite 3: Partitioned MV With Predicate

StarRocks

MV:

Select * from t where a > 20

Query:

Select * from t where a > 10

MV and Base Table have two partitions:

Dt=20250201 — **Refreshed**

Dt=20250202 — **To Refresh**

Step1: Following Microsoft's Paper

Assuming All MV Partitions are refreshed

```
select * from t
where a > 10 and a <= 20
union all
select * from "MV"
```

Step2: Compensating Partition Predicates

Consider MV Partitions refresh status

```
select * from "MV"
=>
select * from t where a > 20 and
dt='20250202'
union all
select * from MV
```

Step3: Get the final result

Merge the result from step1 and step2

```
select * from t where a > 10 and a <= 20
union all
(select * from t where a > 20 and
dt='20250202'
union all
select * from mv)
```

StarRocks Query Optimizer

1 Background

2 Representative Optimizations

03 CBO Challenges

4 Learning Thinking

Challenges Of CBO

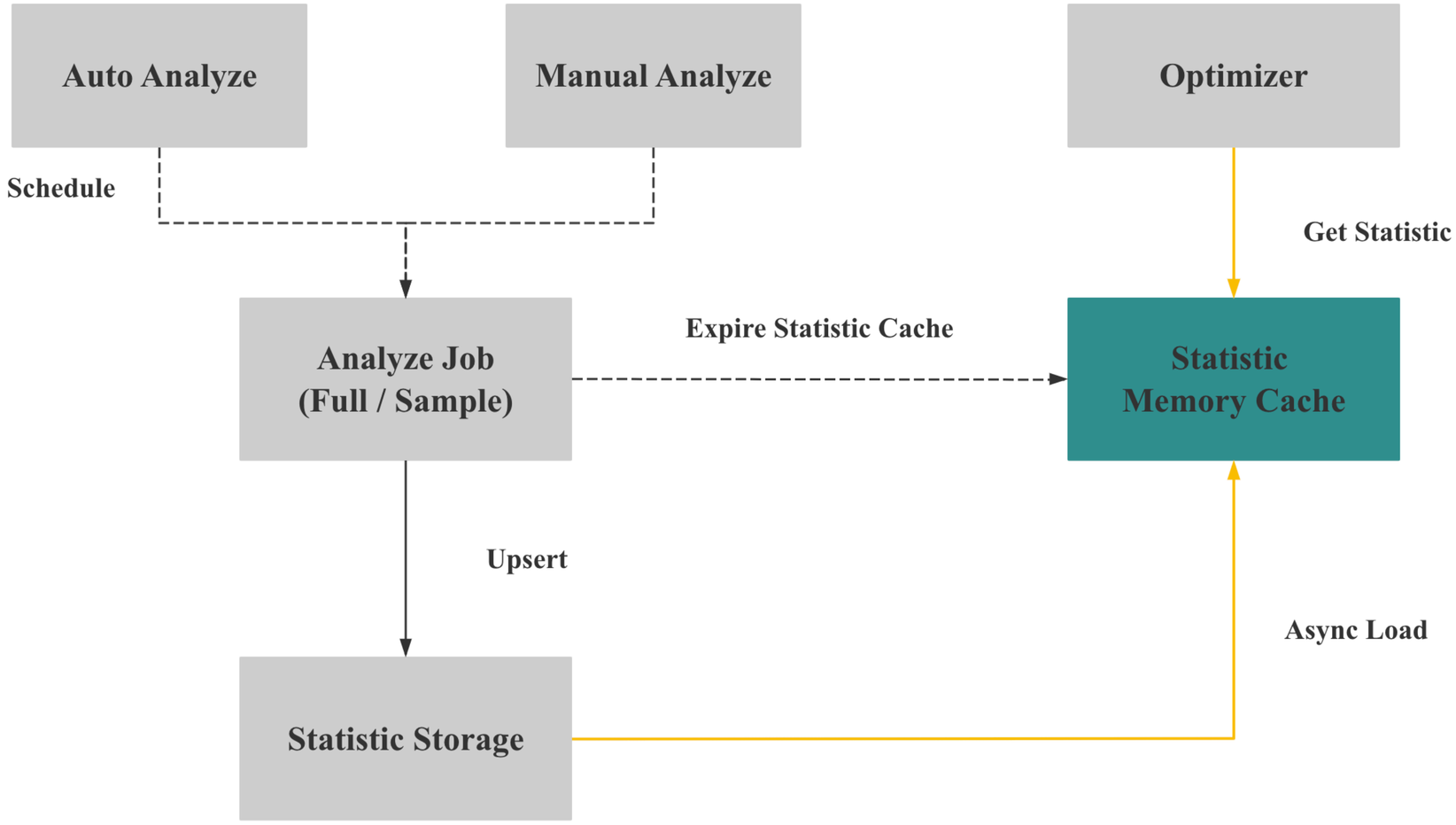
StarRocks

CBO Challenges	StarRocks Solutions
No Statistics	Auto Analyze
Inaccurate Sampling Statistics	Table Sample
Statistics Estimation Errors	Full Analyze Predicate Columns
Cost Estimation Errors	Query Feedback Adaptive Execution
Data Skew	Histogram Skew-Awared Shuffle Join
Query Plan Jitter	SQL Plan Management Hint

StarRocks Statistics Overview

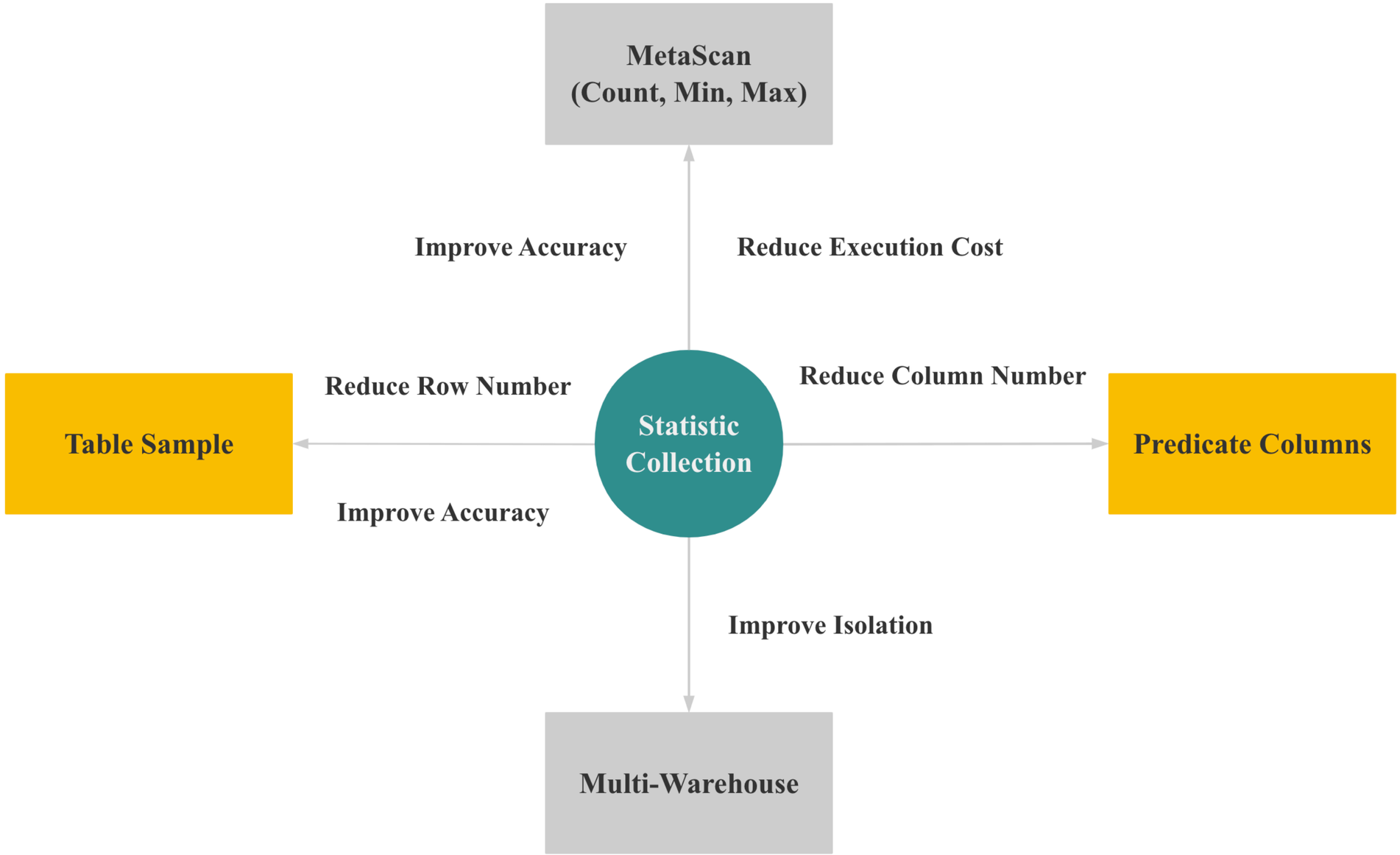
- Count
- Max, Min
- Null Fraction
- Distinct Count
- Avg Row Size

- Histogram
  - MCV
  - Bucket Statistics



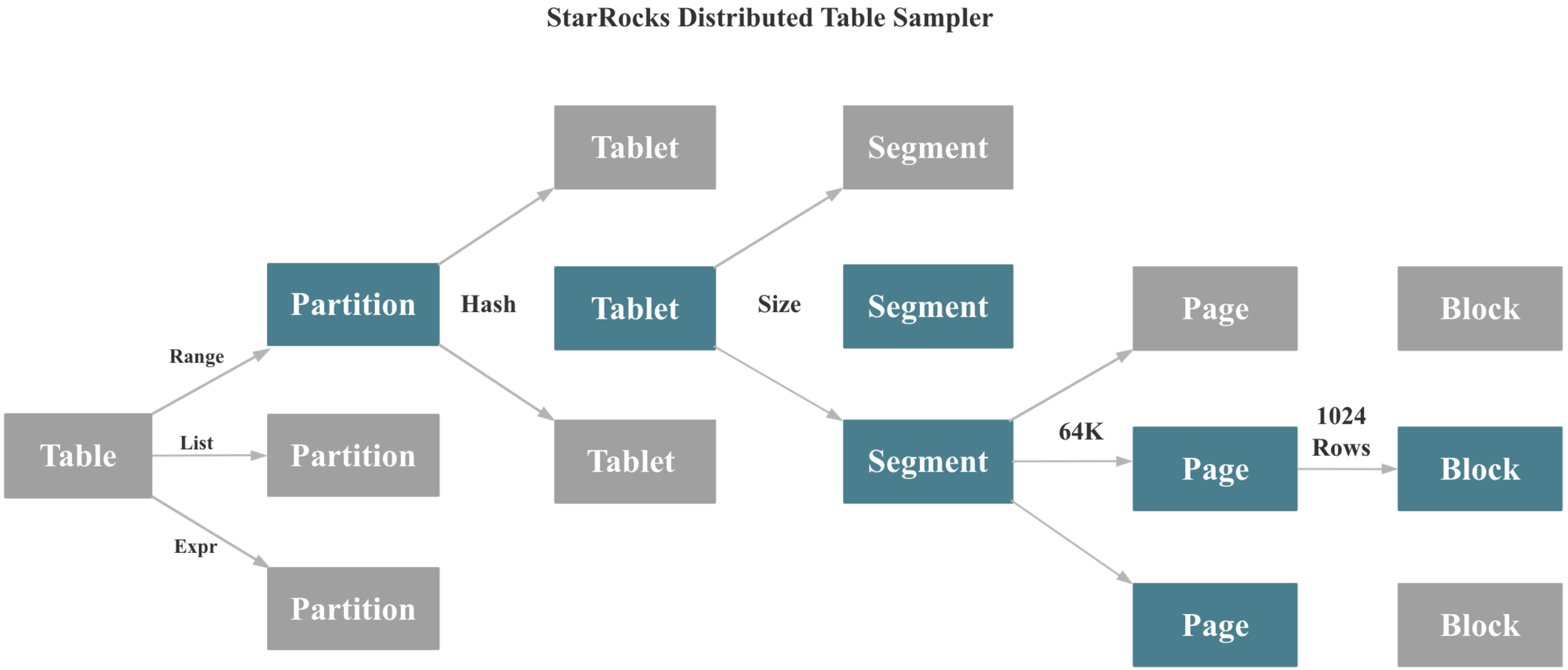
Optimizes Performance and Accuracy of Statistics Collection

StarRocks

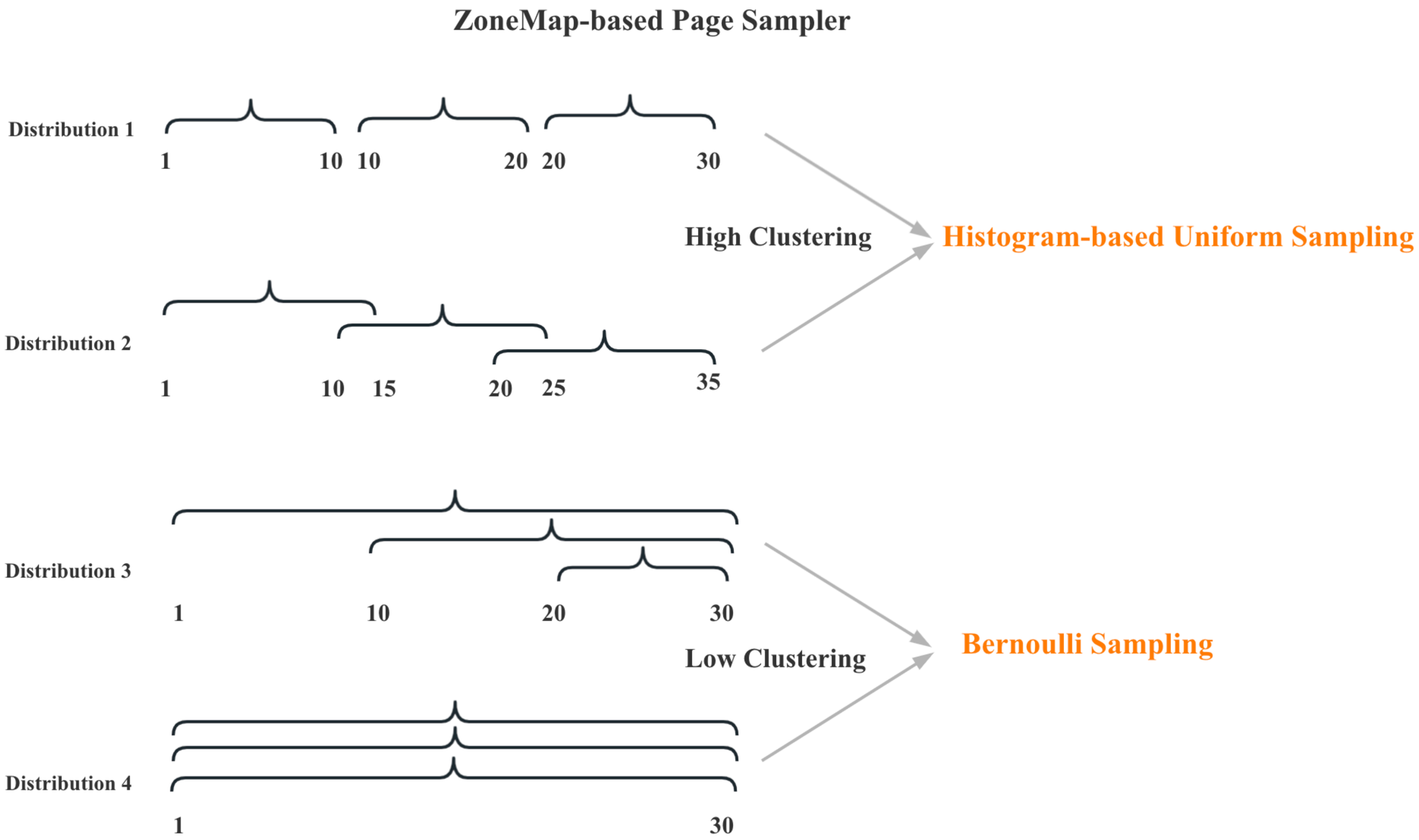


StarRocks Table Sample 1

StarRocks



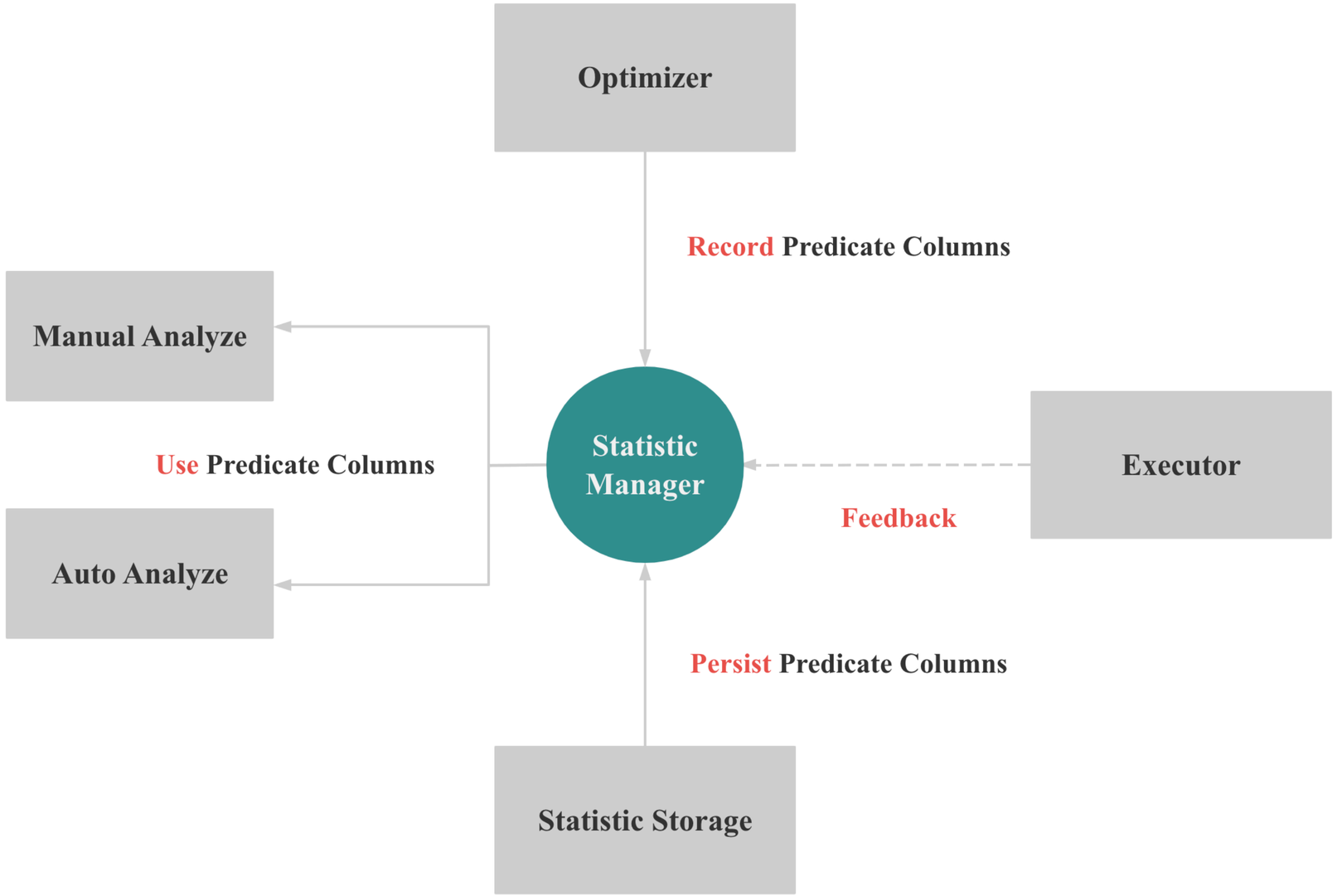
StarRocks Table Sample 2



StarRocks Predicate Columns

StarRocks

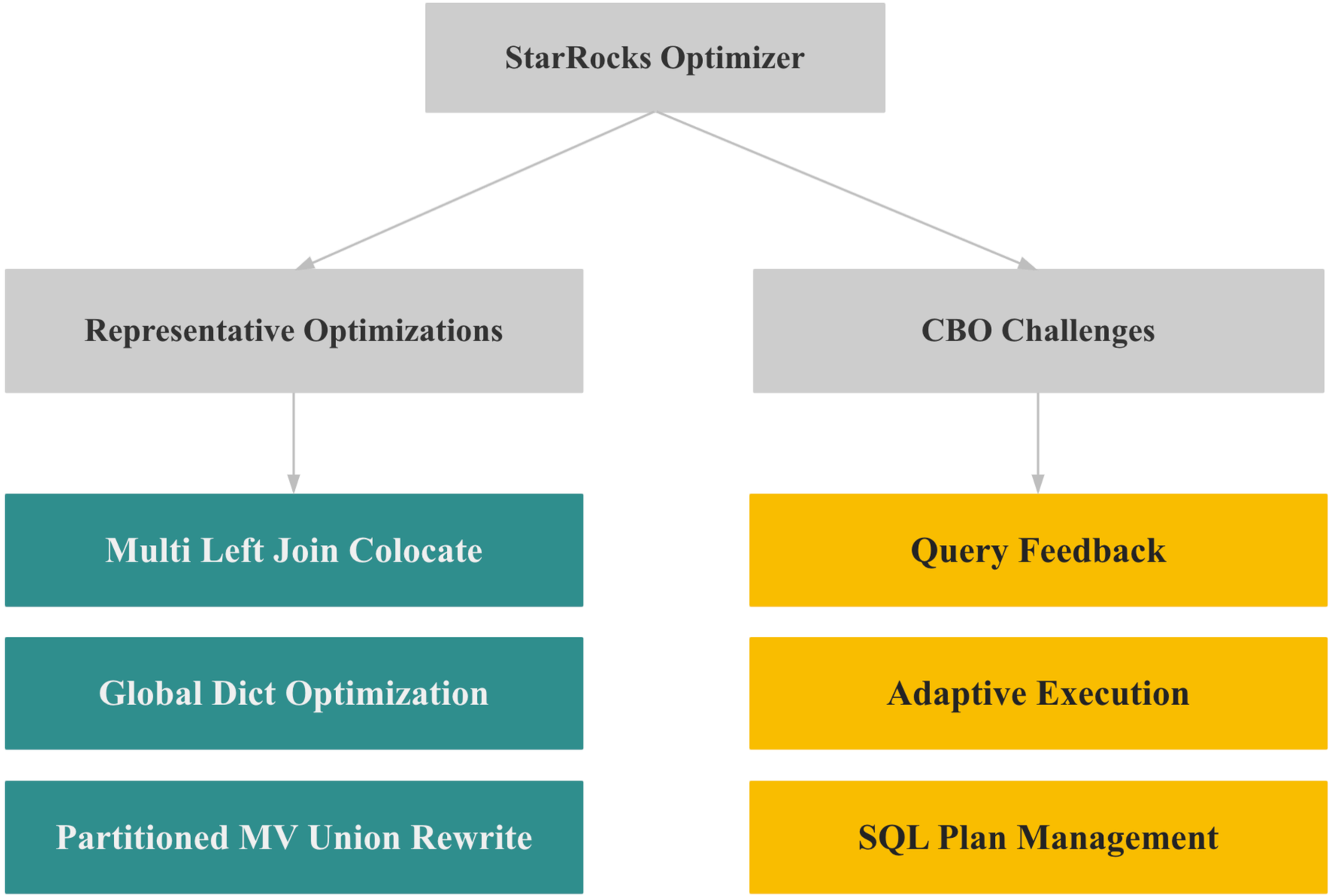
- *Where Filter Columns*
- *Group By Columns*
- *Distinct Columns*
- *Join On Columns*



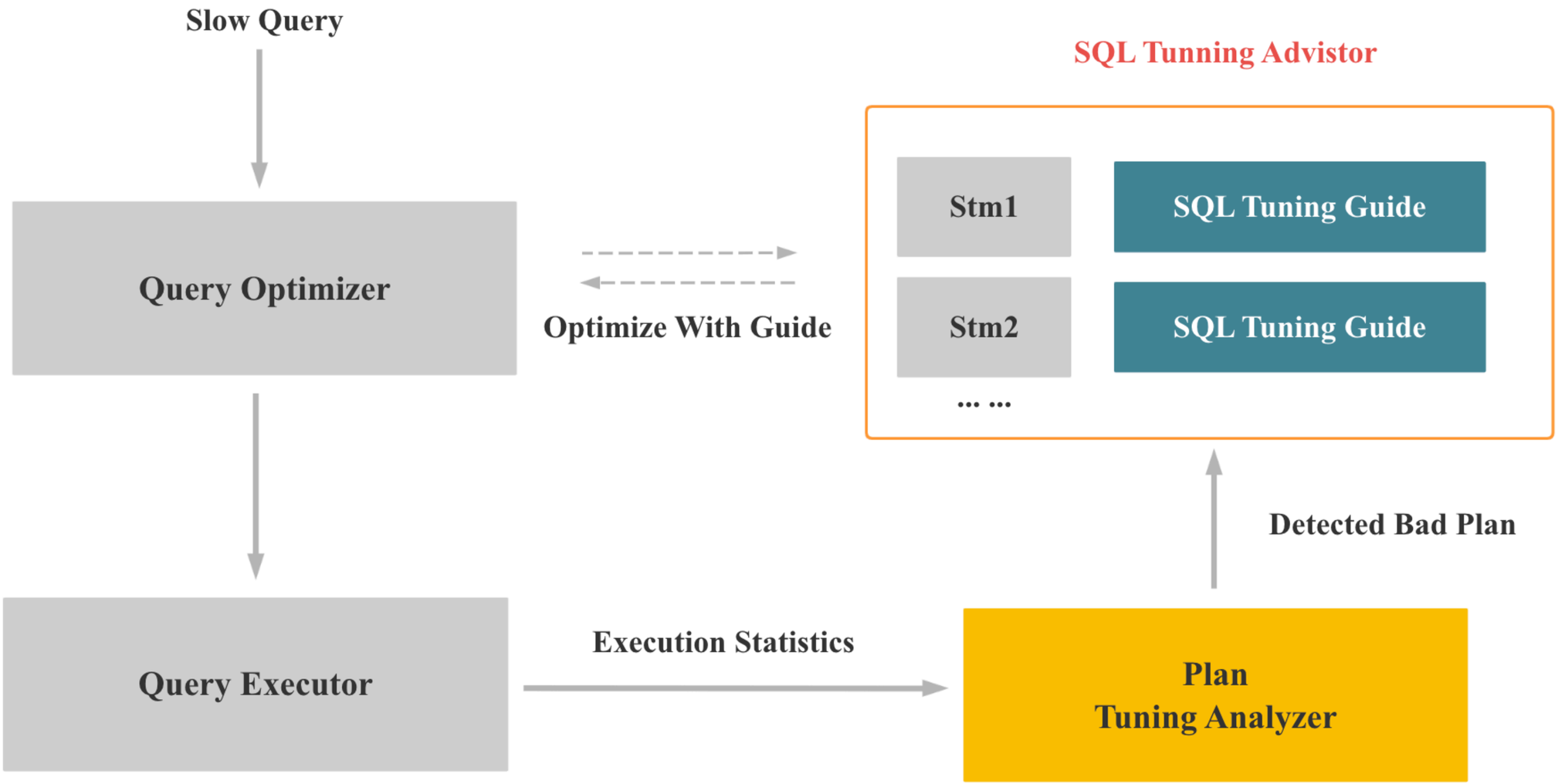
Column Statistics need to be collected Only for Predicate (Dimension) columns

StarRocks Optimizer Introduction

StarRocks



Query Feedback: Overview



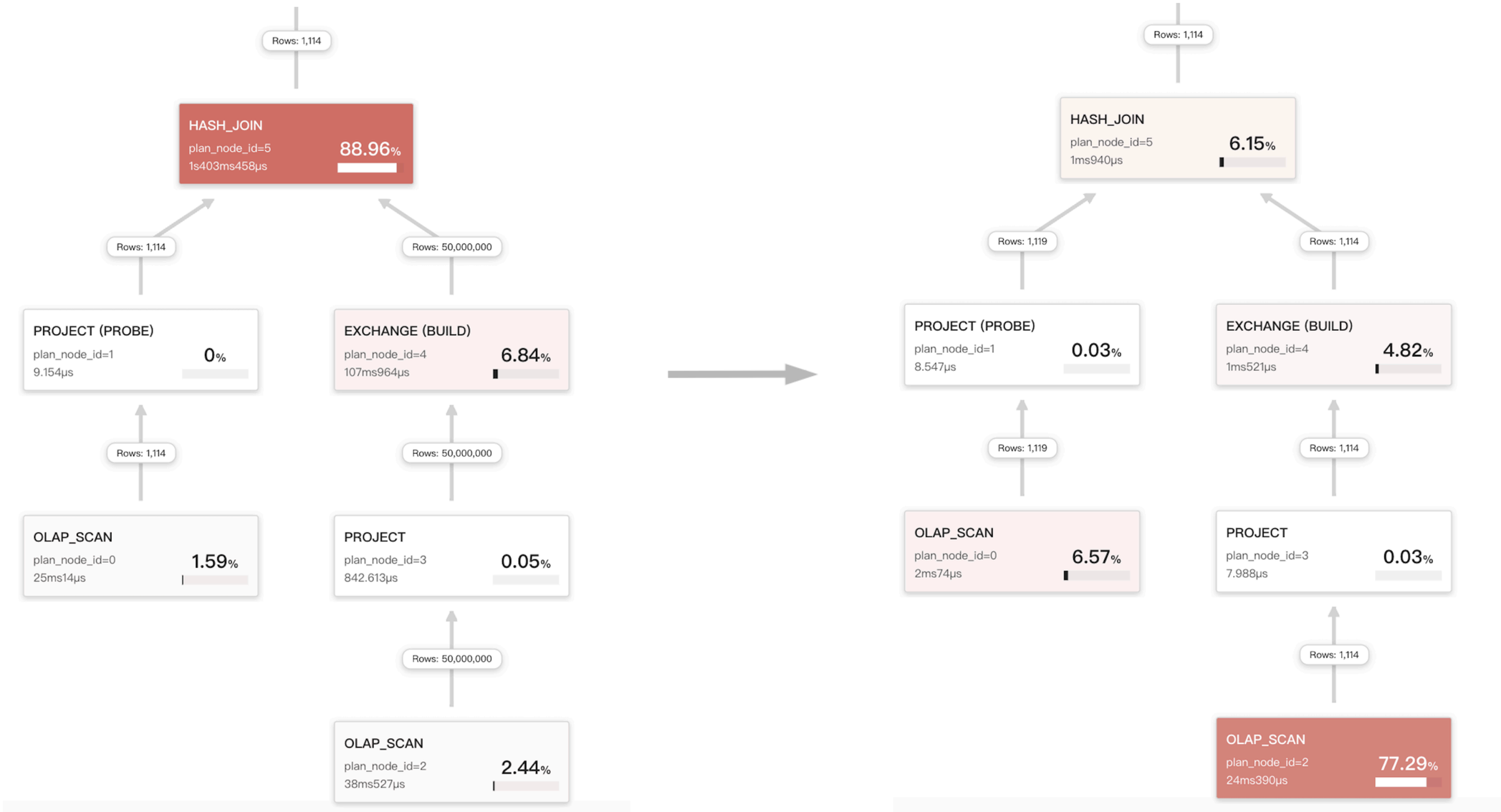
The result of Query Feedback is the tuning guide

Query Feedback: Use Case & What's the Tuning Guide

StarRocks

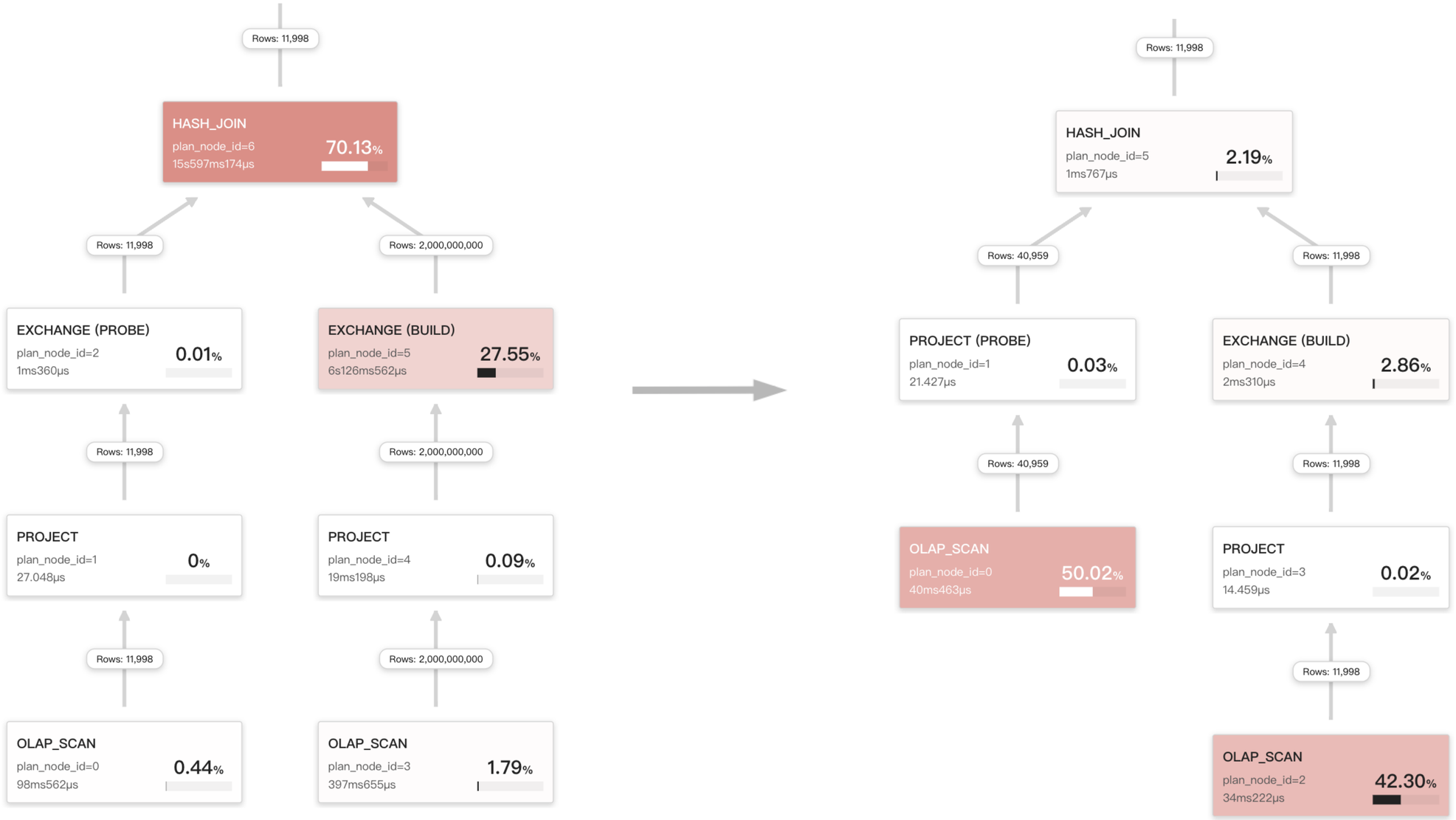
- *Join tuning guide*
  - *Optimize the left and right order of Join*
  - *Optimize the distributed execution mode of Join （eg: broadcast -> shuffle）*
- *Agg tuning guide*
  - *For cases where first-phase aggregation works well, disable streaming aggregation and aggregate more data in first phase.*

Query Feedback: Use Case



Optimize the left and right order of Join nodes

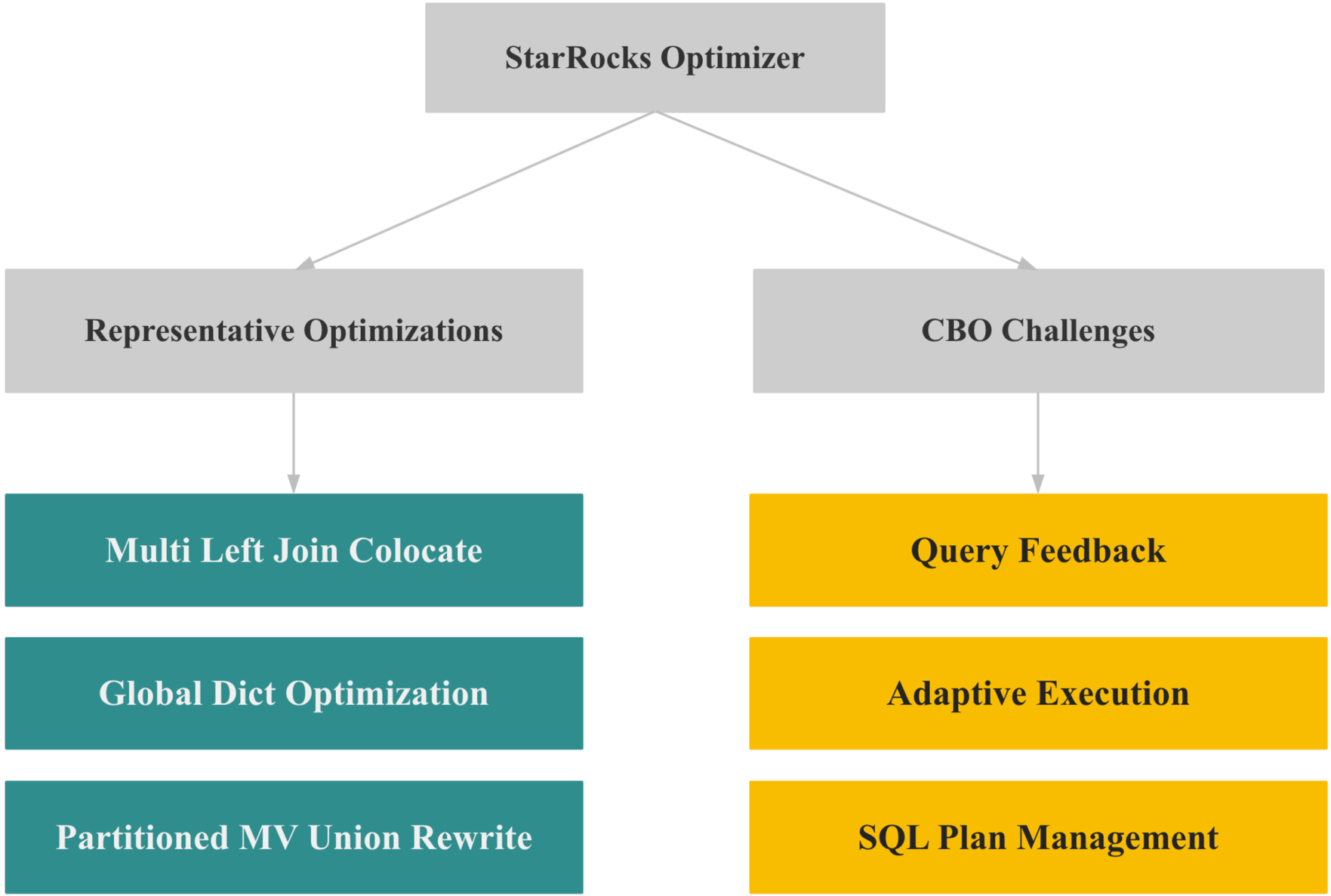
Query Feedback: Use Case



Optimize the distributed execution mode of Join nodes: Shuffle -> Broadcast

StarRocks Optimizer Introduction

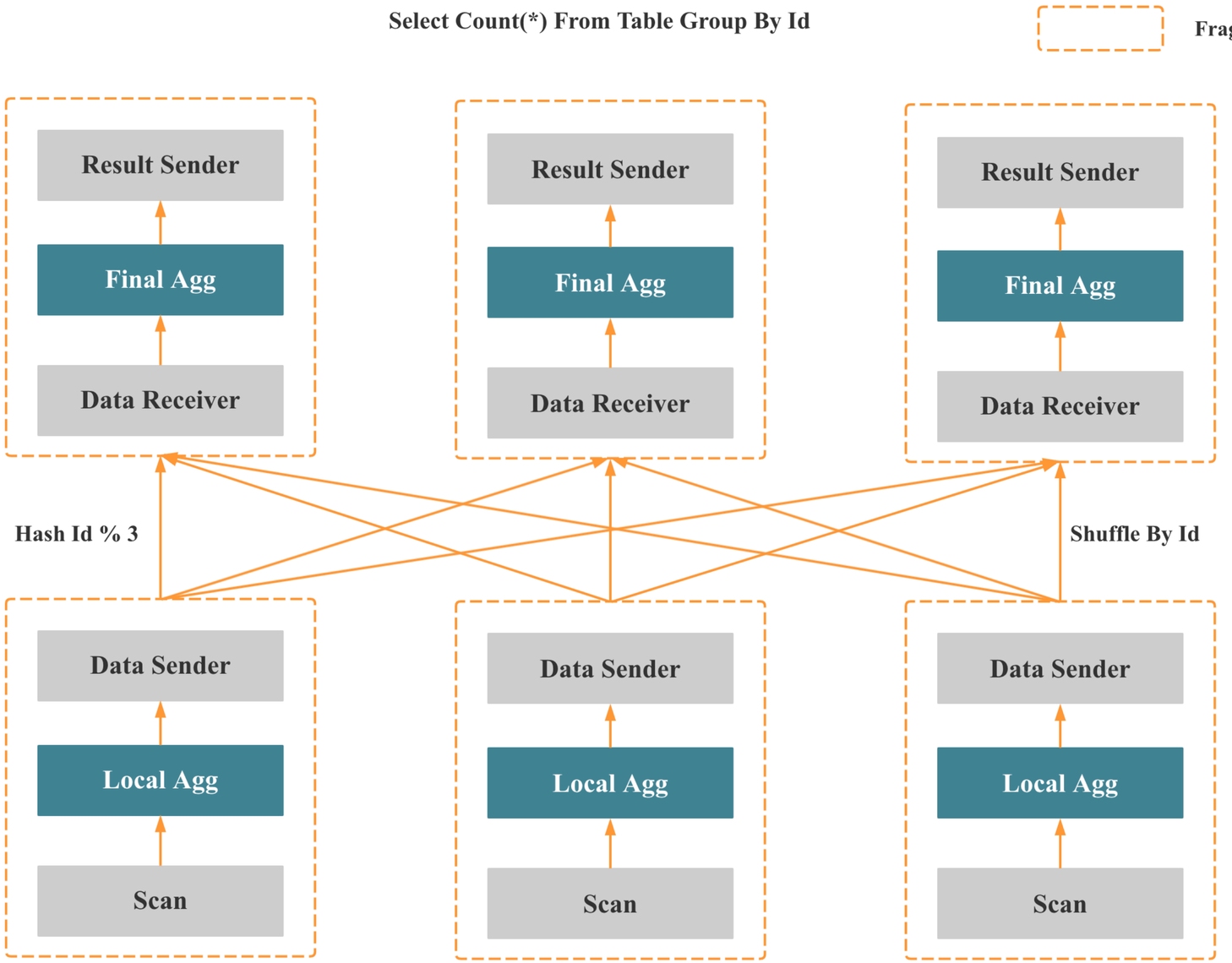
StarRocks



Adaptive Streaming Aggregate 1

StarRocks

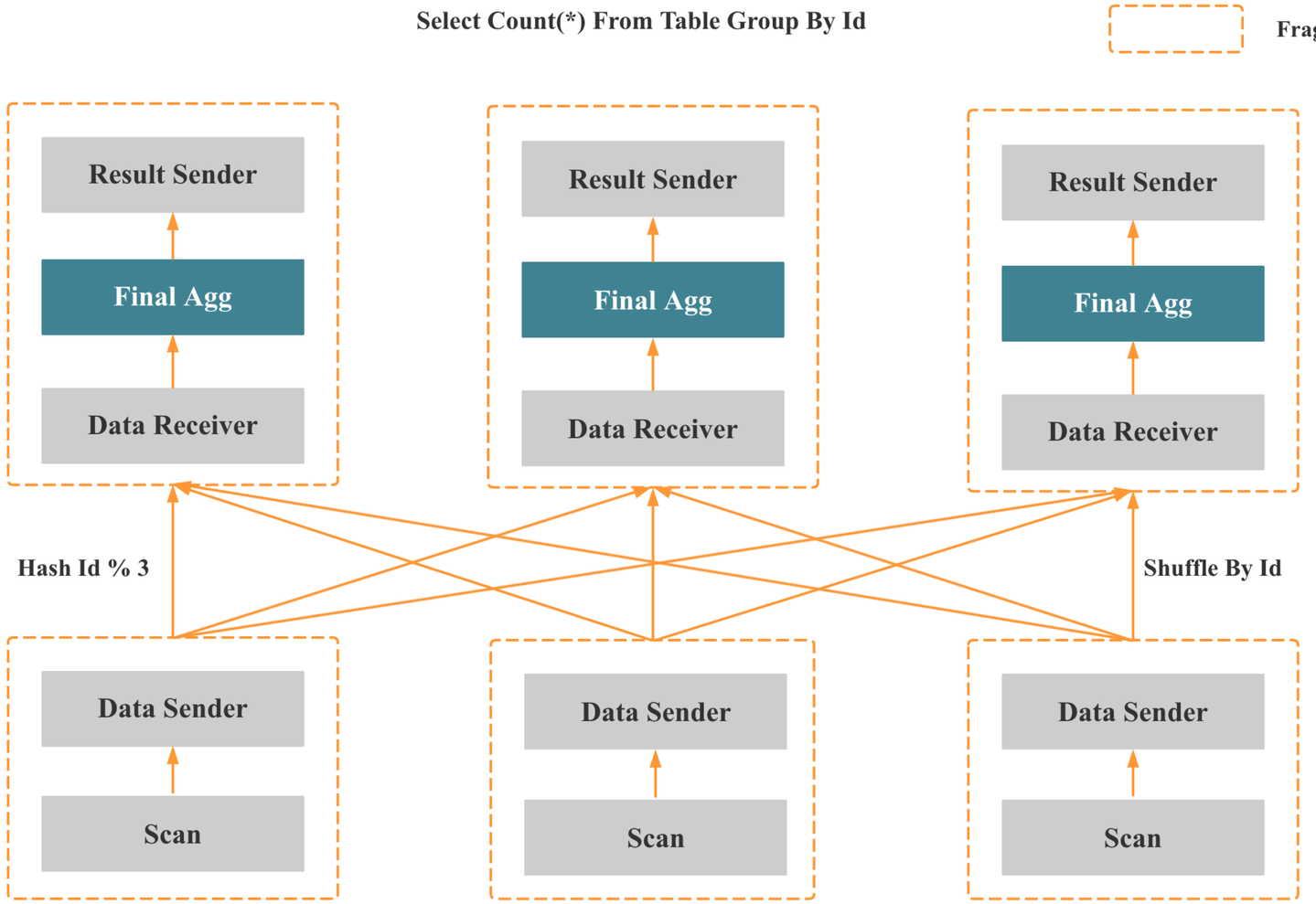
Distributed Two-Phase Aggregation



Local Agg has aggregation effect:
It can reduce the amount of shuffle data

Adaptive Streaming Aggregate 2

Distributed One-Phase Aggregation



Local Agg has no aggregation effect:

- **Reduces the build of useless HashTable**
- **Reduces CPU and Memory resources**

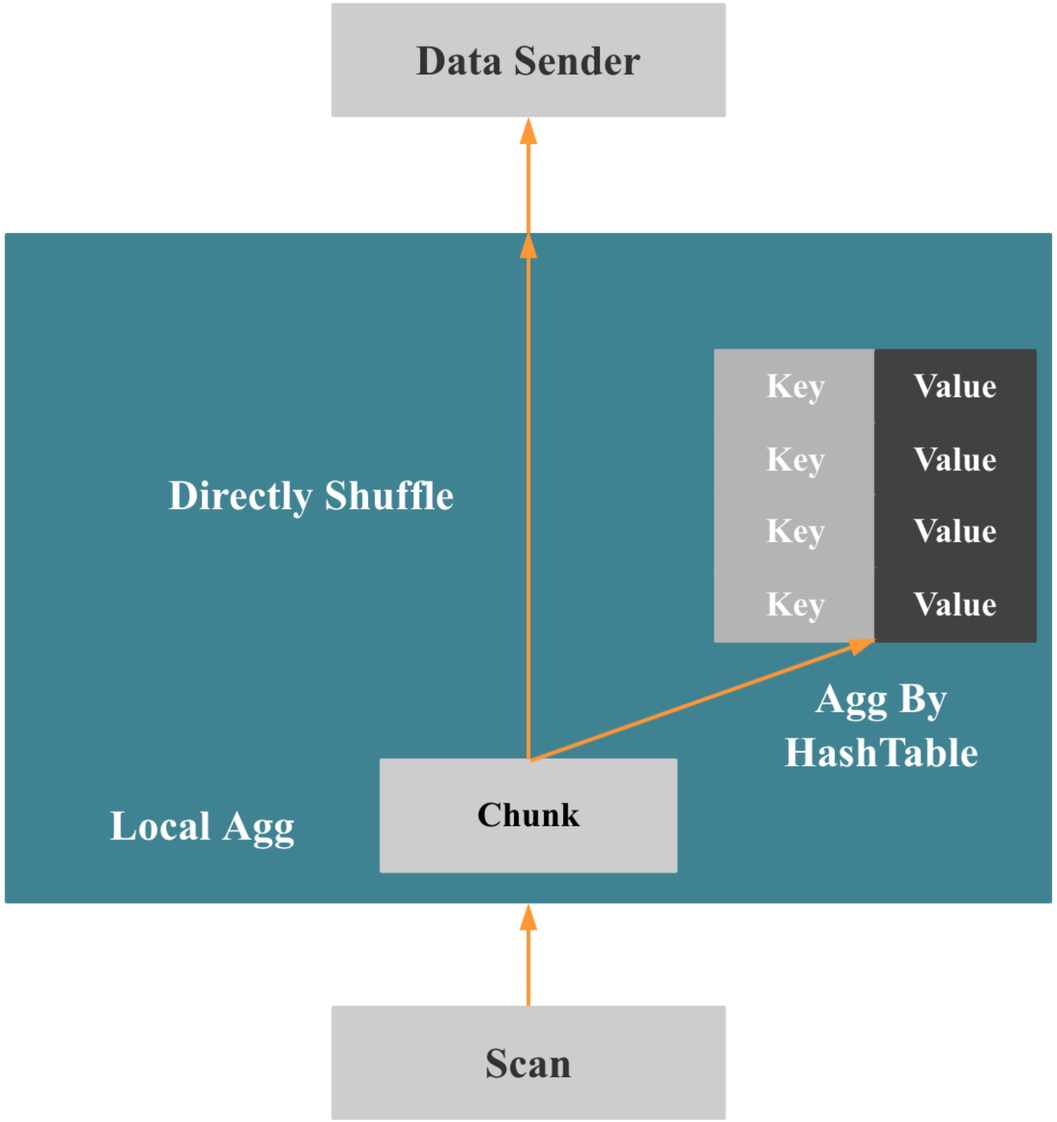
Adaptive Streaming Aggregate 3

StarRocks

- Prefer two-phase aggregation
- Runtime adaptive
- Good aggregation effect: use hash table
- Bad aggregation effect: directly shuffle data

Do not rely on error-prone cardinality estimates

Rely on real runtime statistics



Adaptive Runtime Filters Selection 1

StarRocks

- Support Join, TopN, Agg Runtime Filter
- Support Local And Global Runtime Filter
- Shuffle Aware
- Push down Max/Min, In Filter To Storage Engine
- Support Runtime Filter Cache
- Push Runtime Filter To Two Sides
- SIMD Bloom Filter
- Adaptive Join Runtime Filters Selection

Reduce Disk IO, Reduce Network Transport， Reduce Join Probe Rows

10 x ~ 100x Performance Improvement For Complex Join Query

Adaptive Runtime Filters Selection 2

StarRocks

Issues:

- *Some filters are ineffective*
- *Filters with low selectivity are applied last*

Filter compute has cost,

we should use most selective filters

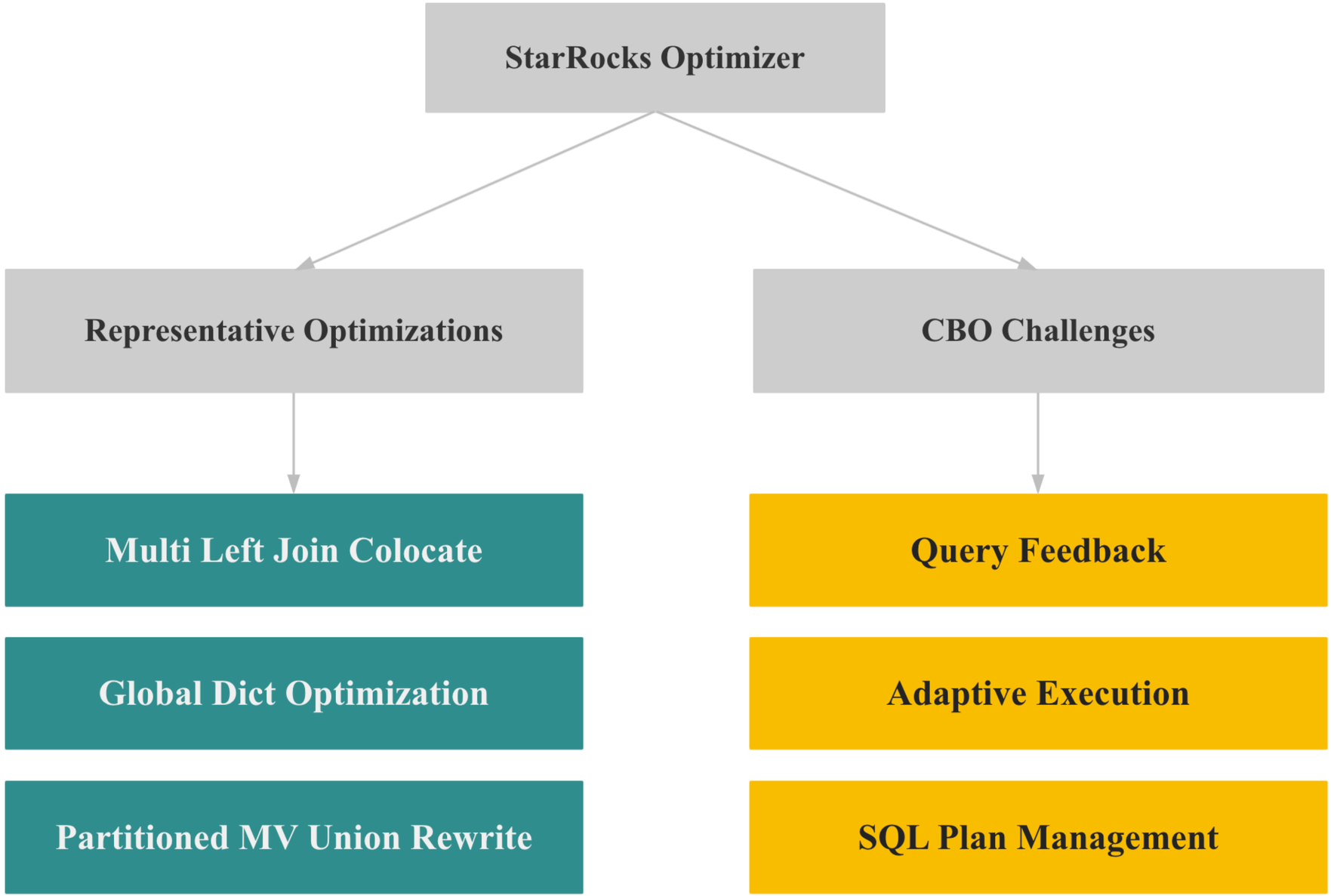
```
double selectivity = true_count * 1.0 / chunk_size;
if (selectivity <= 0.5) {      // useful filter
    if (selectivity < 0.05) { // very useful filter, could early return
        _selectivity.clear();
        _selectivity.emplace(selectivity, rf_desc);
        chunk->filter(new_selection);
        return;
    }

    // Only choose three most selective runtime filters
    if (_selectivity.size() < 3) {
        _selectivity.emplace(selectivity, rf_desc);
    } else {
```

Do not rely on error-prone selective estimates

Rely on real runtime statistics

StarRocks Optimizer Introduction

StarRocks



The Goal of SQL Plan Management

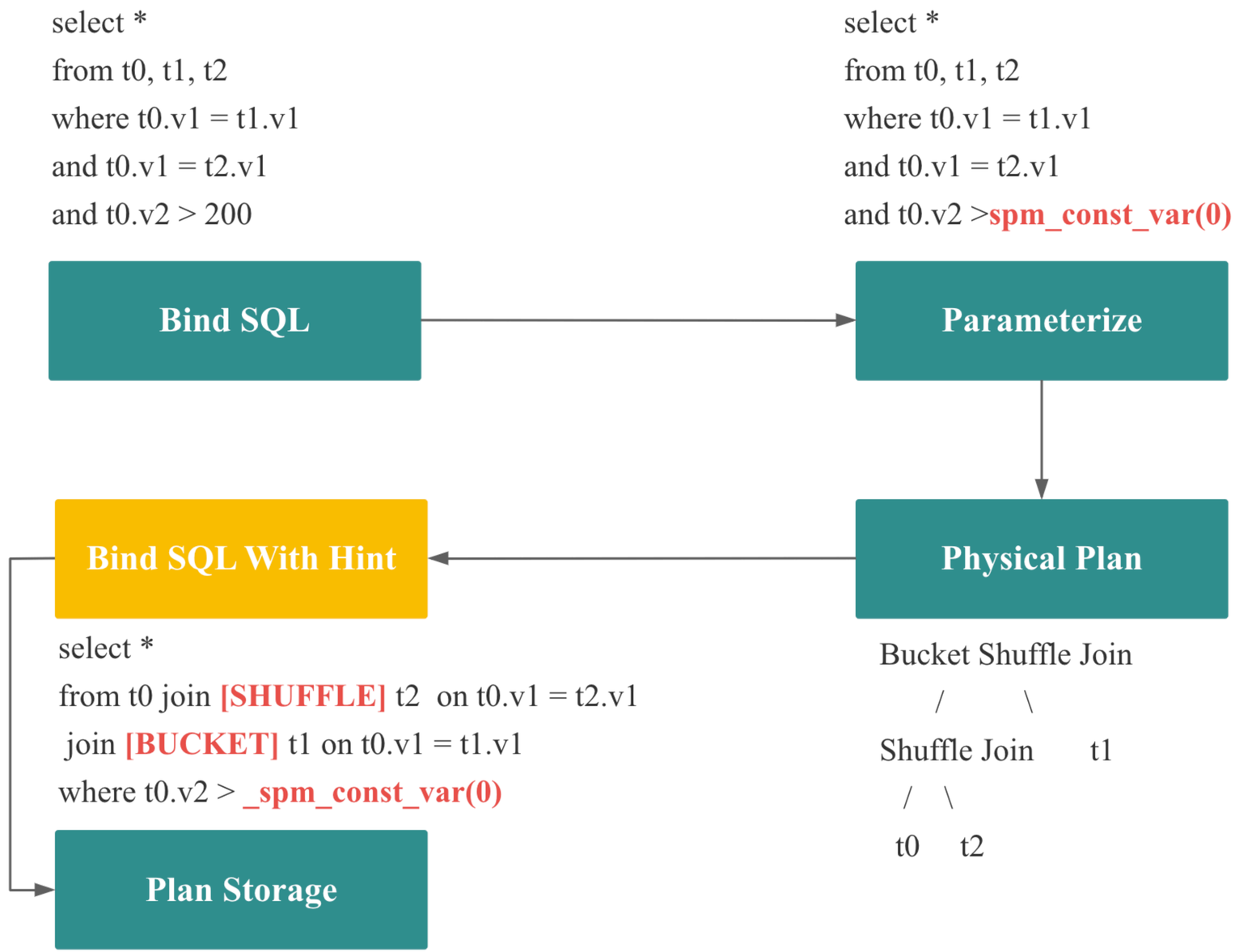
StarRocks

- Ensure no performance regression after upgrading
  - It is difficult to ensure that the changes to the optimizer are 100% positive optimization.

- Ensure that the query plan for online business remains unchanged
  - Plan changes may lead to big queries, which in turn may cause production accidents

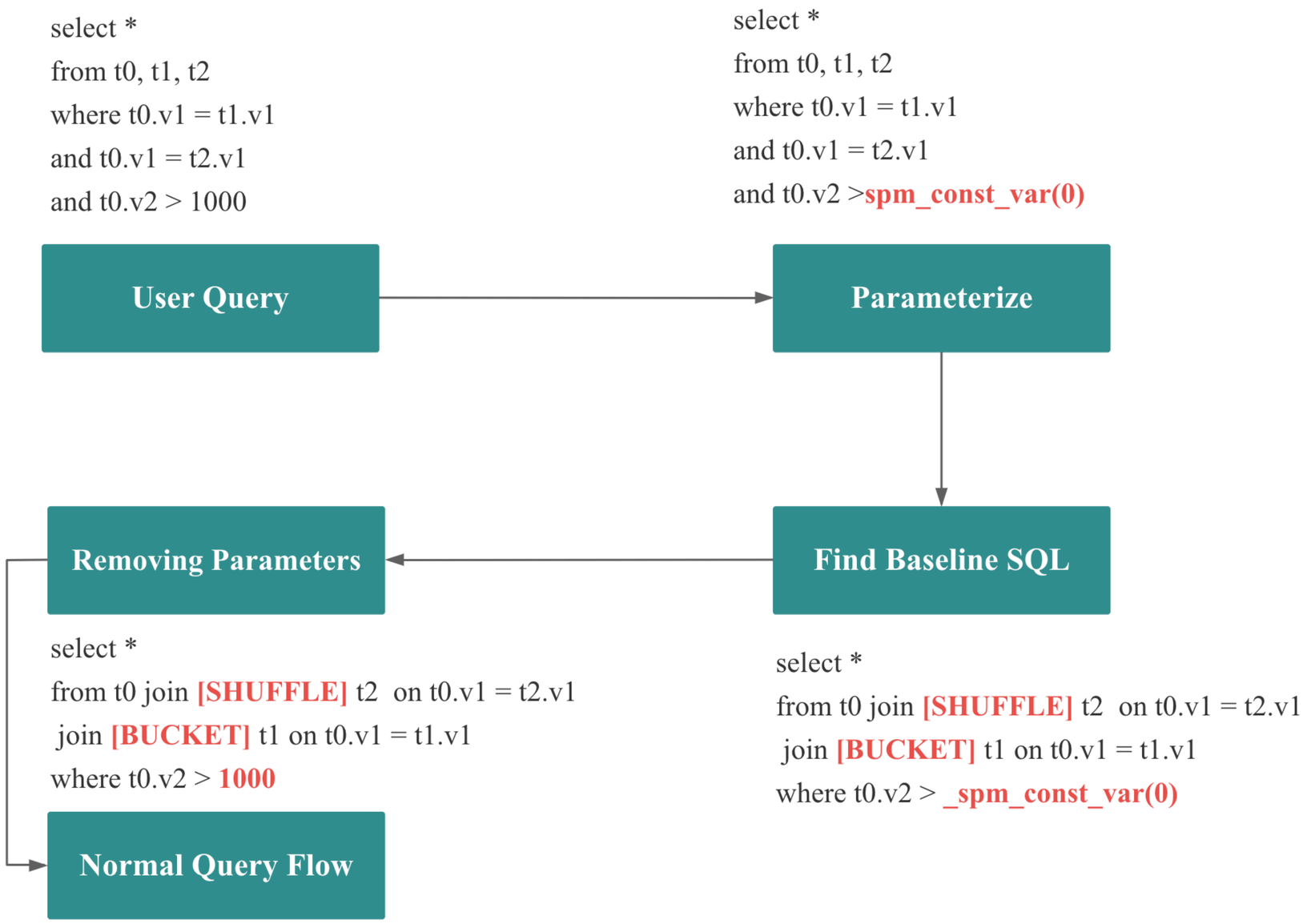
SQL Plan Management 1: How to Generate Baseline Plan

StarRocks



SQL Plan Management 2: How to Use Baseline Plan

StarRocks



Lessons From StarRocks Query Optimizer

StarRocks

- *Errors in optimizer cost estimation are unavoidable, and the executor needs to be able to make autonomous decisions and provide timely feedback.*
- *In engineering, the optimizer testing system is as important as the optimizer itself. The optimizer requires correctness testing, performance testing, plan quality testing, etc.*
- *Null and Nullable is Interesting and Annoying*
  - *For Performance, we need to special handle Null and nullable*
  - *For Correctness, we need to take care of Null and nullable in Optimizer and Executor*
- *Integrated optimizer could be more powerful than Standalone optimizer*

2025

Thanks

kangkaisen.com