Apache Flink SQL & Calcite

The integretion of Apache Flink SQL and Apache Calcite

陈玉兆 Danny Chan
Apache Calcite PMC & Committer

FLINK FORWARD # ASIA
实时即未来 # Real-time Is The Future

FLINK FORWARD

Contents
目录

FLINK FORWARD

Calcite 基础介绍

The background of Apache Calcite

01

Calcite 基础介绍 | 项目背景

Apache Calcite 是一个用于构建数据管理系统的可扩展框架

Apache Calcite is an extensible framework for building data management systems

起源 2012 - 进入 Apache 孵化器孵化

Apache incubator in 2012

问题 构建一个高质量的数据库需要 ~ 20 自然人年(的努力)

Building a high-quality database requires ~ 20 person years (effort)

方案 开源框架构建数据库系统

Create an open source framework to build their own DBMS

设计 Flexible → Relational algebra

Extensible → Volcano style planner

替待 PostgreSQL, Apache Spark, GPORCA

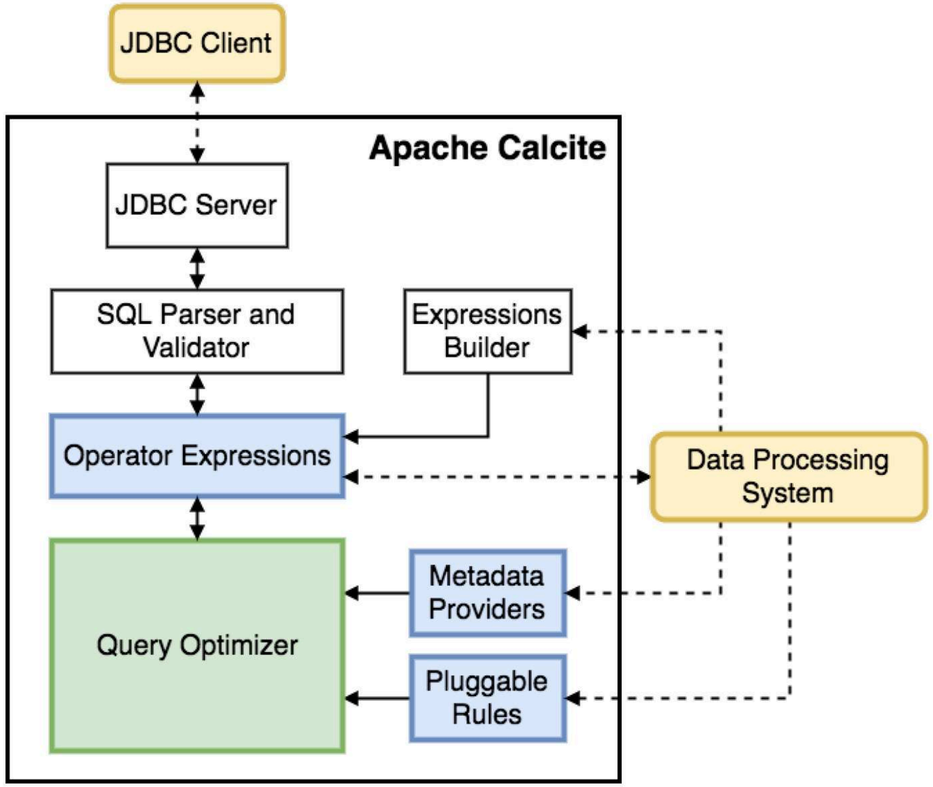
Calcite 基础介绍 | 核心组件

Core – Operator expressions (relational algebra) and planner (based on Volcano/Cascades)

External – Data storage, algorithms and catalog

Optional – SQL parser, JDBC & ODBC drivers

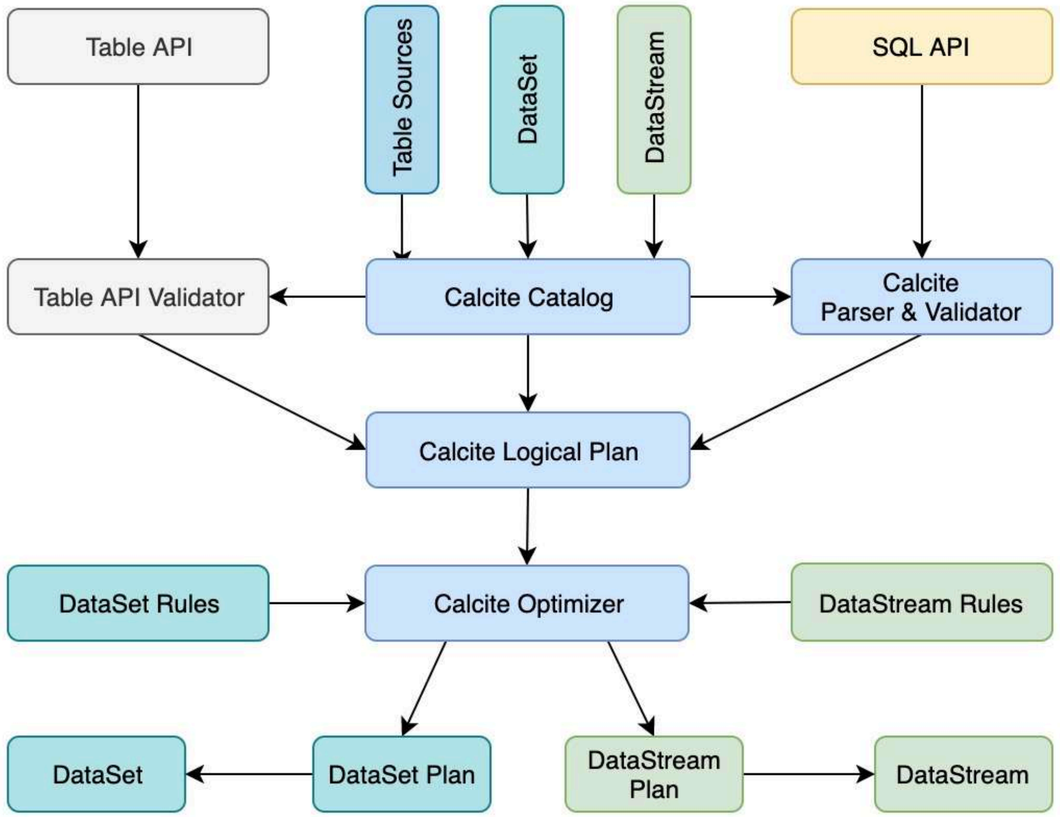
Extensible – Planner rewrite rules, statistics, cost model, algebra, UDFs



Calcite 基础介绍 | Flink 用到的组件

System	Query Language	JDBC Driver	SQL Parser and Validator	Relational Algebra	Execution Engine
Apache Drill	SQL + extensions	✓	✓	✓	Native
Apache Hive	SQL + extensions			✓	Apache Tez, Apache Spark
Apache Solr	SQL	✓	✓	✓	Native, Enumerable, Apache Lucene
Apache Phoenix	SQL	✓	✓	✓	Apache HBase
Apache Kylin	SQL	✓	✓		Enumerable, Apache HBase
Apache Apex	Streaming SQL	✓	✓	✓	Native
Apache Flink	Streaming SQL	✓	✓	✓	Native
Apache Samza	Streaming SQL	✓	✓	✓	Native
Apache Storm	Streaming SQL	✓	✓	✓	Native
MapD [32]	SQL		✓	✓	Native
Lingual [30]	SQL		✓	✓	Cascading
Qubole Quark [42]	SQL	✓	✓	✓	Apache Hive, Presto

Calcite 基础介绍 | Flink SQL 体系结构



Flink SQL 集成 Calcite

The integretion of Flink SQL and Apache Calcite

02

Flink SQL 集成 Calcite | 语法扩展

DDL

CREATE TABLE – FLIP-6962

Computed Column – FLIP-70 CALCITE-3379

Time Attribute – FLIP-66

Type Parse Extension – CALCITE-3213

```
CREATE TABLE t (
 a int,
 b varchar(20),
 c as to_timestamp(b), -- computed column
 -- watermark strategy
 watermark for `c` as myfunc(c, 1) - interval '5' second
) WITH (
 'k1' = 'v1'
 ...
);
```

Flink SQL 集成 Calcite | 语法扩展

Hive Dialect

PARTITION Support – FLIP-12742
FLINK-12867

Whole PARTITION Story – FLIP-63

```
CREATE TABLE t (
...
)
PARTITIONED BY (column1, column2)
WITH (
...
);
```

```
INSERT INTO t PARTITION(a = 2, b = 'abc') SELECT FROM …;
```

Flink SQL 集成 Calcite | Catalog

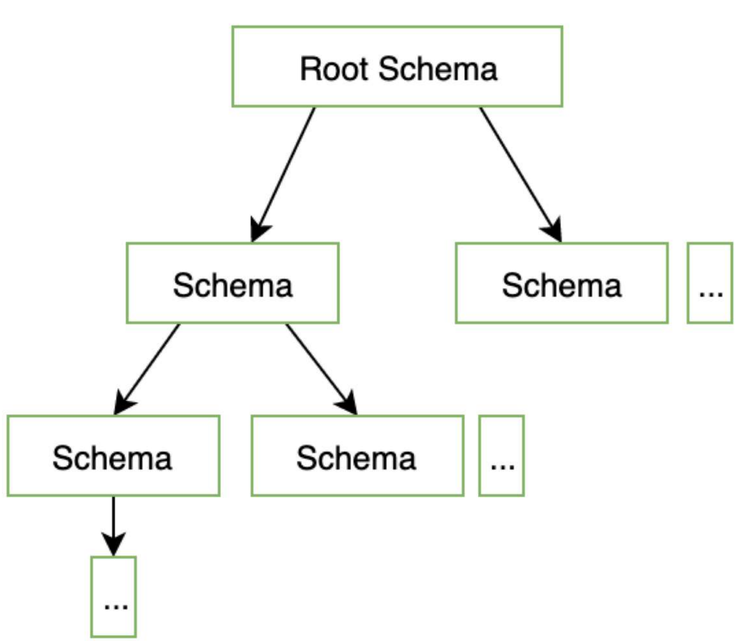
Why Catalog ?

Manage tables – Catalog -> DataBase -> Object(Table)

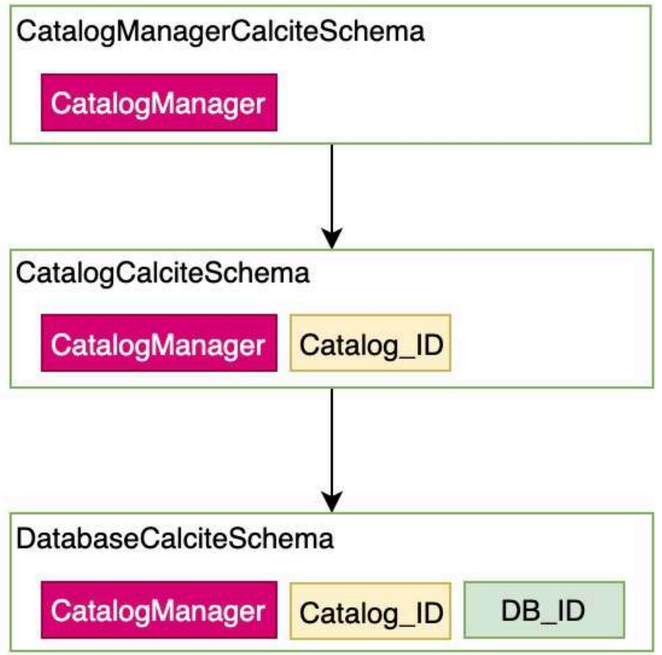
Table metadata persistence – Store table schema, the statistics

Bridge Alien Systems – Load meta from Hive metastore to read and write

Flink SQL 集成 Calcite | Catalog

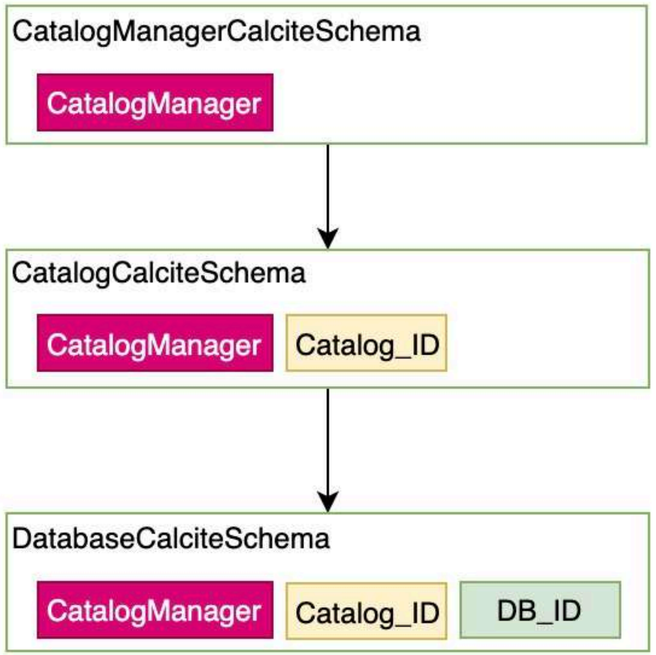


Calcite Schema

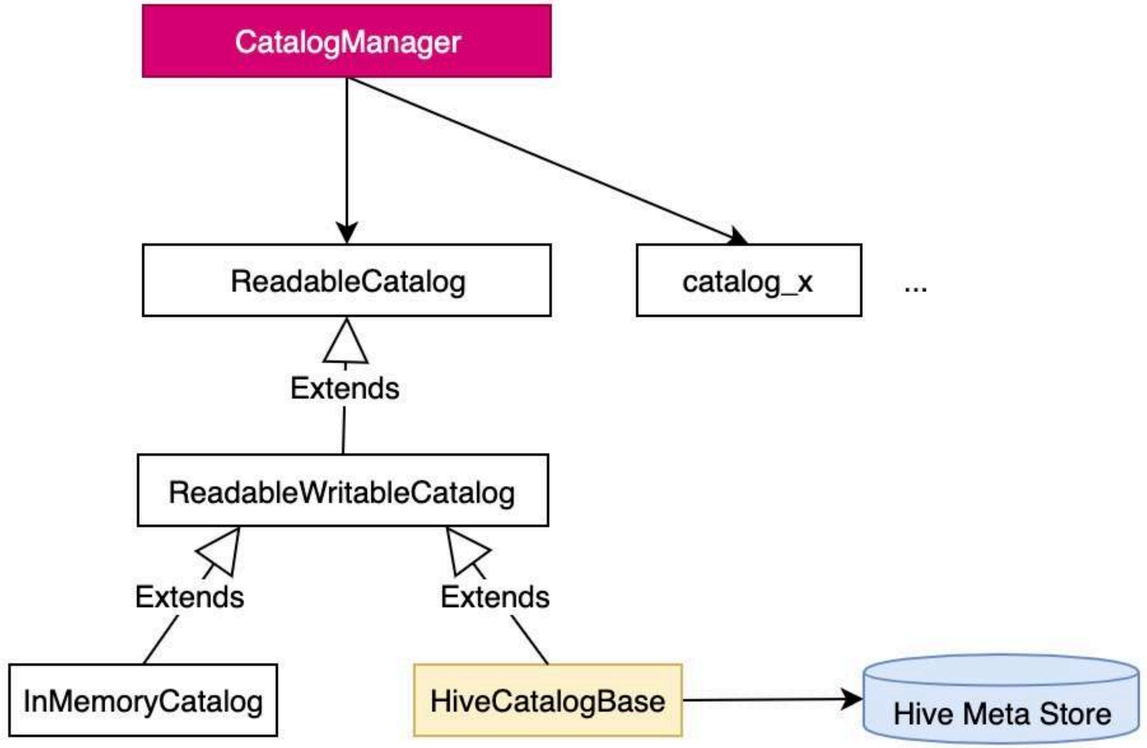


Flink Schema

Flink SQL 集成 Calcite | Catalog



Flink Schema



Flink Catalog

Flink SQL 集成 Calcite | Catalog

How To Use

```
// Register a catalog instance named "my_catalog" into the session.
tEnv.registerCatalog("my_catalog", catalog_instance);

// Use the catalog1 by specifying the name
tEnv.useCatalog("catalog1");

// Use the database
tEnv.useDataBase("database1");
```

Flink SQL 集成 Calcite | Metadata Extension

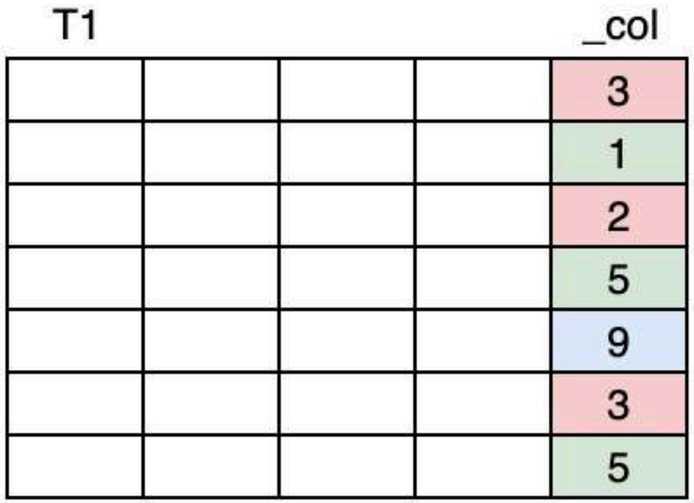
Column Interval – The column values interval, to make more accurate estimate of selectivity with predicates

Column Null Cnt – Column null values count, to filter out the null values of inner join inputs to reduce the skewed nulls

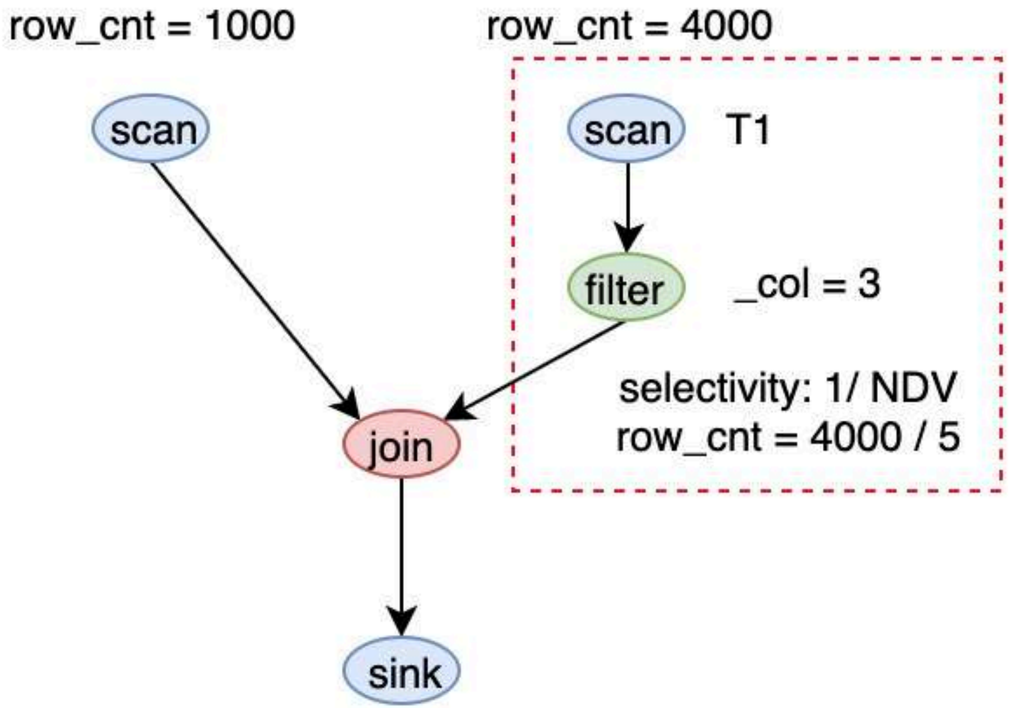
Minimum Unique Keys – Minimus unique keys permutation to reduce hash key computation burden

In CALCITE-3446, we also promote the extensibility of the RelMetadataQuery

Flink SQL 集成 Calcite | Metadata Extension



Number of Distinct Value = 5



Use Case of ColumnInterval

Flink SQL 集成 Calcite | Planner Rules

Multi Sink – Support DAG plan promotion

Sub plan reuse – Reuse the result of the pipline data sets

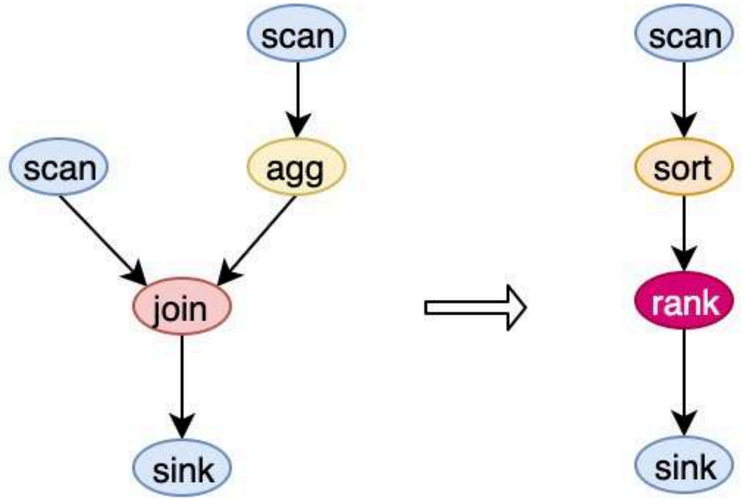
Self-join promotion – Use uniform Rank instead of agg and self join

Shuffle Reduction – Ajust the shuffle distribution atomatically to reduce data shuffle(Network IO)

Flink SQL 集成 Calcite | Planner Rules

T1		col_1	col_2	col_3	
				3	✓
				1	
				2	
				5	✓
				9	
				3	✓
				5	✓

Select col1, col2 from T1 where col3 = (select max(col3) from T1 T2 where T2.col1 = T1.col1 and T2.col2 = T1.col2)



self-join promotion

Flink SQL 集成 Calcite | Implicit Type Coercion

Why Implicit Type Coercion ?

Int > VARCHAR

INT = BOOLEAN

For convenience !

VARCHAR BETWEEN DATE and TIMESTAMP

Flink SQL 集成 Calcite | Implicit Type Coercion

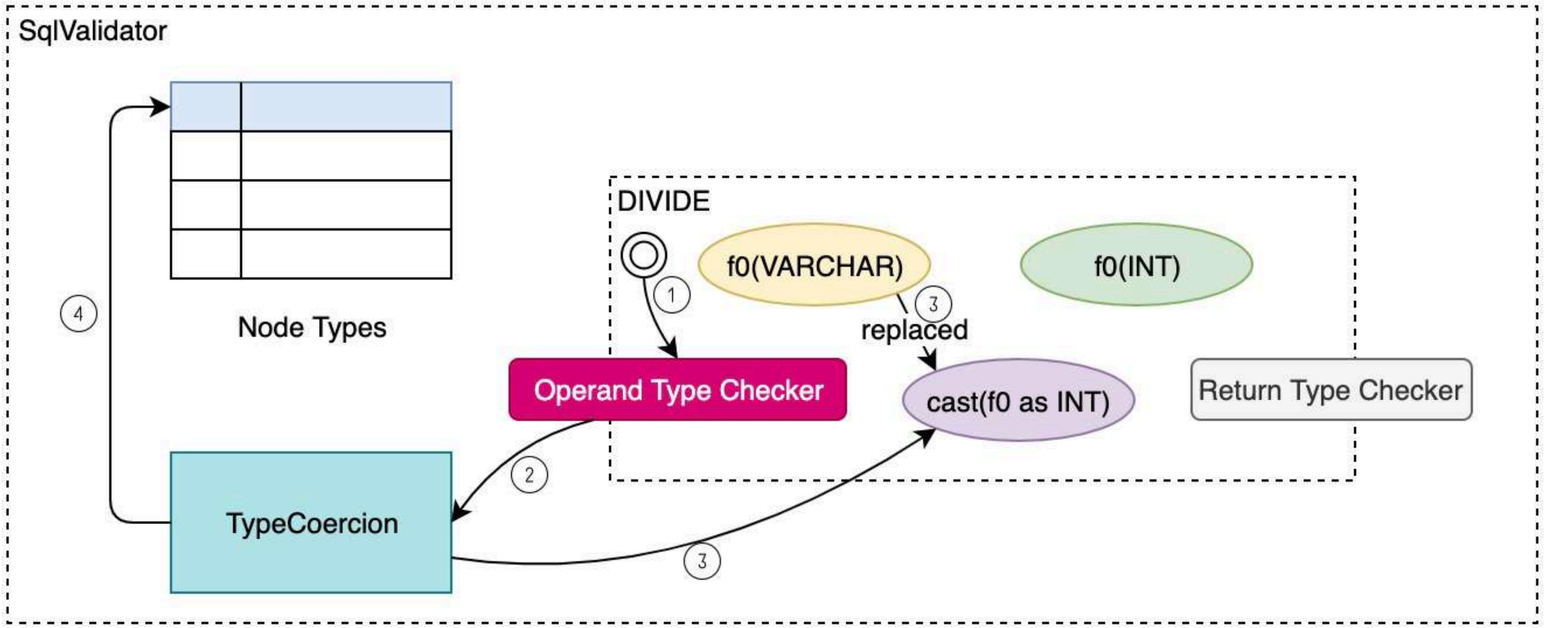
How To Use

```
CREATE TABLE t (
 f_data DATE,
 f_timestamp TIMESTAMP WITHOUT TIME ZONE
) WITH (
 ...
);

SELECT … FROM … WHERE 1 = true or '2019-11-03' BETWEEN f_date and
f_timestamp;
```

Flink SQL 集成 Calcite | Implicit Type Coercion

The Internal



1. SqlValidate fires the validation of the call
2. Operand type checker asks the TypeCoercion for the implicit type coercion
3. TypeCoercion wrap the operand with CAST and replace the operand with the new one
4. update node type cache

Flink SQL 集成 Calcite | SQL Hint

Why SQL Hint ?

Planner Enforcers – There's no perfect planner, so it makes sense to implement hints to allow user better control the execution. For instance: "never merge this subquery with others" (/*+ no_merge */).

Append meta data/statistics – Some statistics like “skew info of some shuffle keys" are somewhat dynamic for the query, config them with hints if our planning metadata is not that accurate.

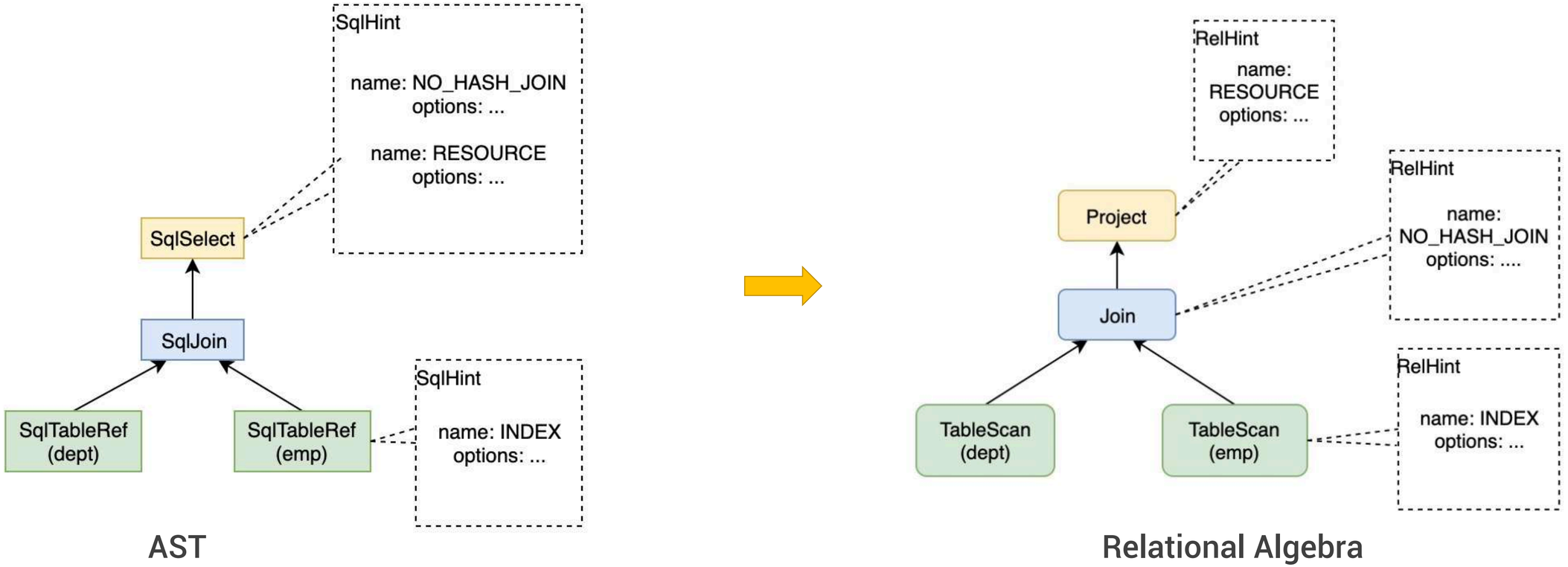
Operator resource constraints – It would be very flexible to profile the resource with hints per query (not the Job).

Flink SQL 集成 Calcite | SQL Hint

The Syntax

```
SELECT /*+ NO_HASH_JOIN, RESOURCE(mem='128mb', parallelism='24') */
FROM
  emp  /*+ INDEX(idx1, idx2) */
  JOIN
  dept
  ON emp.emono = dept.deptno
```

Flink SQL 集成 Calcite | SQL Hint



AST

Relational Algebra

Hints Propagation

展望未来

The Future Work

03

展望未来

Expression Reuse – Reuse the nested referenced expression witnin the projection list

Hign Level Algebra Expression – Recursion, Loop

Traits Propagation – CALCITE-2970


THANKS