Apache Calcite

Julian Hyde | 2023/05/02

Julian Hyde

Staff Engineer, Google

Member, Apache Software Foundation

Author, of Apache Calcite, Morel, Mondrian, SQLstream

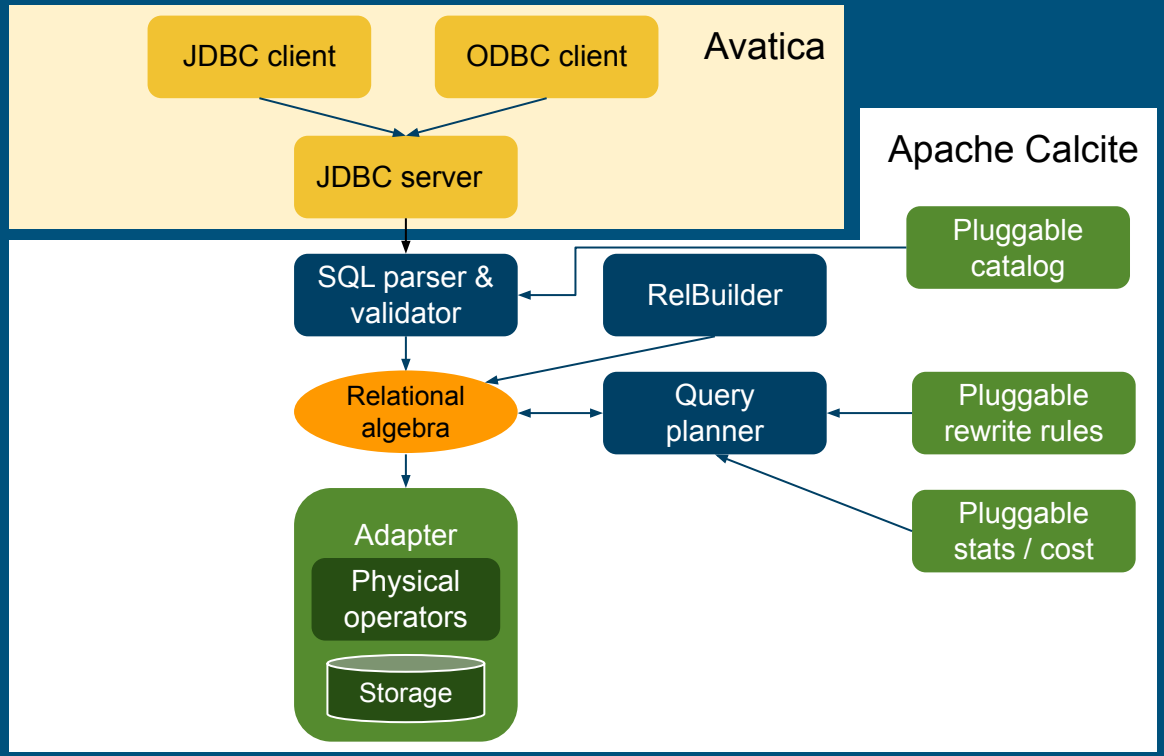
THE APACHE SOFTWARE FOUNDATION

APACHE calcite™

Google Cloud

1. What is Calcite?

Apache Calcite



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

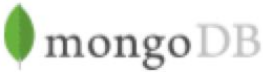
External – Data storage, algorithms and catalog

Optional – SQL parser, JDBC & ODBC drivers

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

Used by

Connects to



main ▾ | 36 branches | 97 tags | Go to file | Add file ▾ | Code ▾

chunweilei [CALCITE-5162] RelMdUniqueKeys can return mor... ··· ✓ f278efb 2 days ago | 4,926 commits

.github/workflows	[CALCITE-5140] Spark, Piglet tests fail in GitHub CI with Ope...	20 days ago
.idea	Enable commit message inspection on IntelliJ IDEA	2 years ago
babel	[CALCITE-4980] Babel parser support MySQL NULL-safe eq...	4 months ago
bom	[CALCITE-5008] Ignore synthetic and static methods in Met...	3 months ago
buildSrc	[CALCITE-4829] Bump Gradle to 7.2 and test with Java 17 at...	8 months ago
cassandra	[CALCITE-5011] CassandraAdapterDataTypesTest fails with i...	3 months ago
core	[CALCITE-5162] RelMdUniqueKeys can return more precise ...	2 days ago
druid	[CALCITE-5131] Remove redundant type cast	21 days ago
elasticsearch	[CALCITE-4992] Resource leak in Elasticsearch adapter	2 months ago
example	Fix typo in filterable-model.yaml	4 months ago
file	[CALCITE-4841] Support decimal column type in CSV file for...	6 months ago
geode	[CALCITE-1794] Expressions with numeric comparisons are ...	3 months ago
gradle/wrapper	Update Gradle to 7.3	7 months ago
innodb	[CALCITE-4858] Use Log4j2 instead of unsupported Log4j (...	7 months ago
kafka	[CALCITE-4858] Use Log4j2 instead of unsupported Log4j (...	7 months ago
linq4j	[CALCITE-3673] ListTransientTable should not leave tables i...	3 months ago
mongodb	[CALCITE-4858] Use Log4j2 instead of unsupported Log4j (...	7 months ago
pig	[CALCITE-4830] Remove remaining uses of ImmutableBean...	8 months ago
piglet	[CALCITE-5131] Remove redundant type cast	21 days ago
plus	[CALCITE-5037] Upgrade HSQLDB to 2.5.2	2 months ago

About

Apache Calcite

calcite.apache.org/

java | sql | big-data | hadoop | geospatial | calcite

Readme
Apache-2.0 license
Code of conduct
3.1k stars
164 watching
1.7k forks

Releases

97 tags

Create a new release

Packages

No packages published
Publish your first package

Contributors 283

+ 272 contributors

dev@calcite.apache.org, past month:

Showing **1 through 10** of **87** topics

dev | issues | commits | private

4wei	**Apply contributor permission** Hi all, I want to contribute to calcite, could anyone please give me the contributor permission…	2	1	2022-05-31 13:16 -07:00
Gavin Ray	**Implementing JSON_TABLE, even hackily as a UDF?** I'm interested in implementing JSON_TABLE functionality for Calcite This opens up some nea…	2	3	2022-05-31 12:47 -07:00
Yanjing Wang	**[DISCUSS] not equal operator vs less than combining greater than** Hi community, I have this sql: select * from "emps" where "name" <> '' and "name" <> '3' I th…	3	8	2022-05-31 04:54 -07:00
Ruben Q L	**[DISCUSS] Towards Calcite 1.31.0** Hello, It has been more than two months since our last release [1], and I think we should mak…	7	7	2022-05-30 10:57 -07:00
itxiangkui (Jira)	**[jira] [Created] (CALCITE-5173) ProjectableFilterableTable may provoke DefaultEnumer…** itxiangkui created CALCITE-5173: ---------- Summary: ProjectableFilter…	1	0	2022-05-30 01:03 -07:00
luws (Jira)	**[jira] [Created] (CALCITE-5172) calcite-elasticsearch query failed with No applicable co…** luws created CALCITE-5172: ---------- Summary: calcite-elasticsearch query…	1	0	2022-05-30 00:47 -07:00
Luke Segars	**Questions for properly handling Postgres system functions** Hello, I'm a relatively new Calcite user, and am working on configuring Calcite to handle queri…	3	2	2022-05-29 20:10 -07:00
Julian Hyde (Jira)	**[jira] [Created] (CALCITE-5171) NATURAL join and USING should fail if join columns are …** Julian Hyde created CALCITE-5171: ---------- Summary: NATURAL joi…	1	0	2022-05-29 11:57 -07:00
Roman Kondakov (…	**[jira] [Created] (CALCITE-5170) Assertion error on range distribution creation** Roman Kondakov created CALCITE-5170: ---------- Summary: Asse…	1	0	2022-05-29 02:32 -07:00
Benchao Li (Jira)	**[jira] [Created] (CALCITE-5169) 'xx < 1 OR xx > 1' cannot be simplified to 'xx <> 1'** Benchao Li created CALCITE-5169: ---------- Summary: 'xx < 1 OR xx …	1	0	2022-05-29 00:33 -07:00

Project goals

Make it easier to write a simple DBMS

Advance the state of the art for complex DBMS by pooling resources

Bring database approaches to new areas (e.g. streaming)

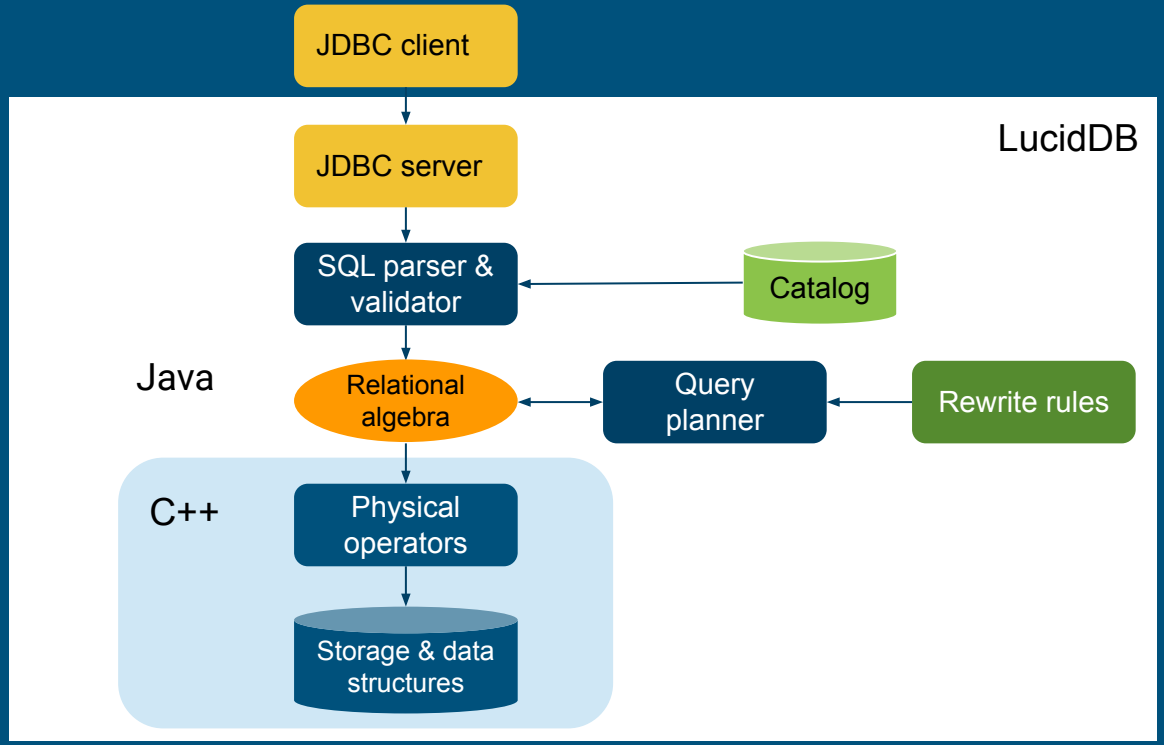
Allow create a DBMS by composing pieces (federation, etc.)

Customize by plugging into framework, evolving framework when necessary

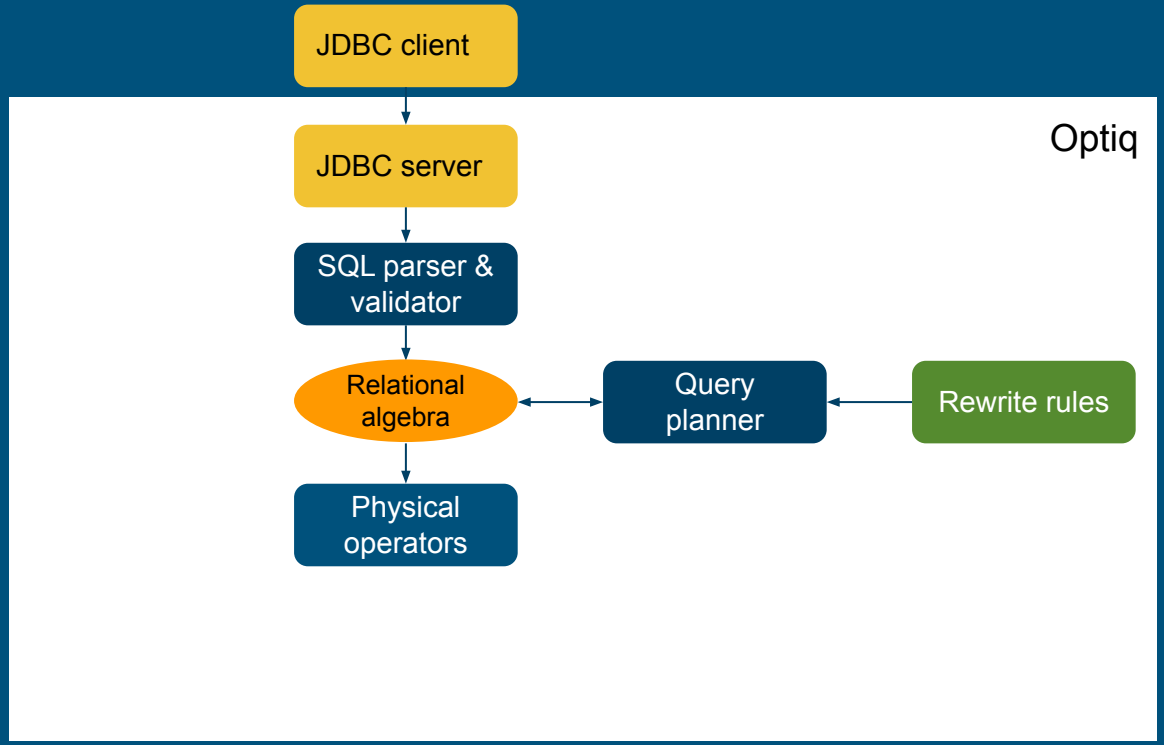
Apache license & governance

2. Evolution

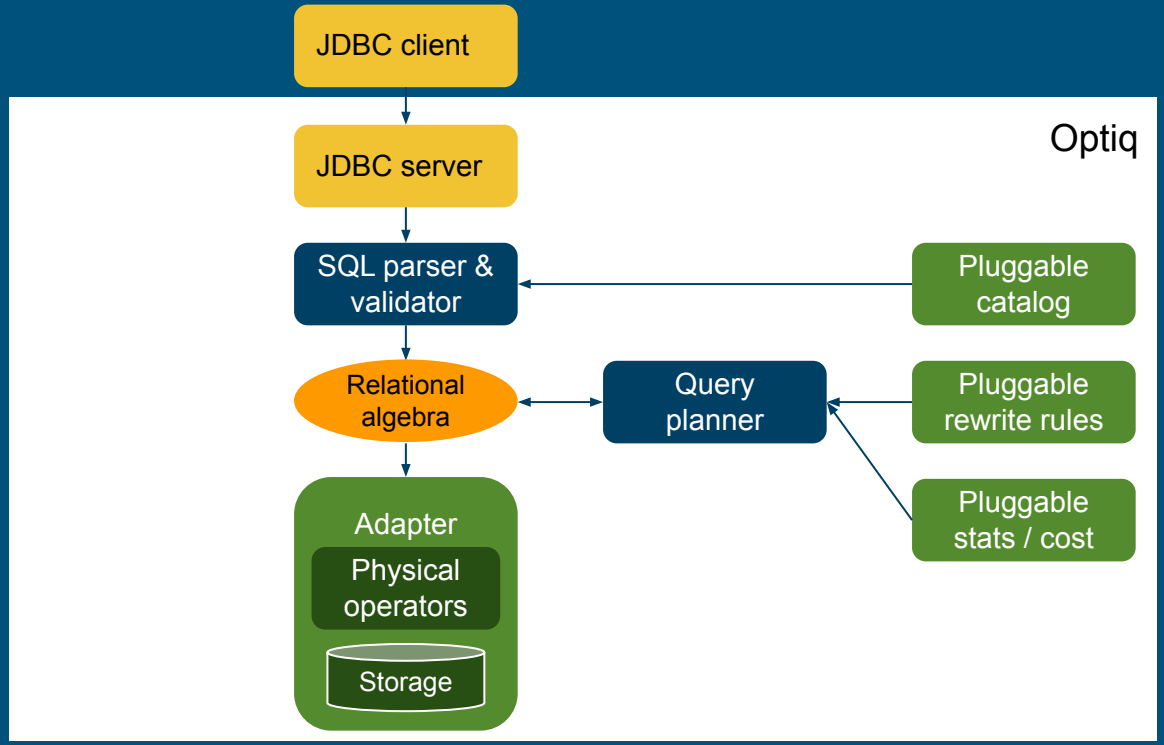
Calcite evolution - origins as an SMP DB



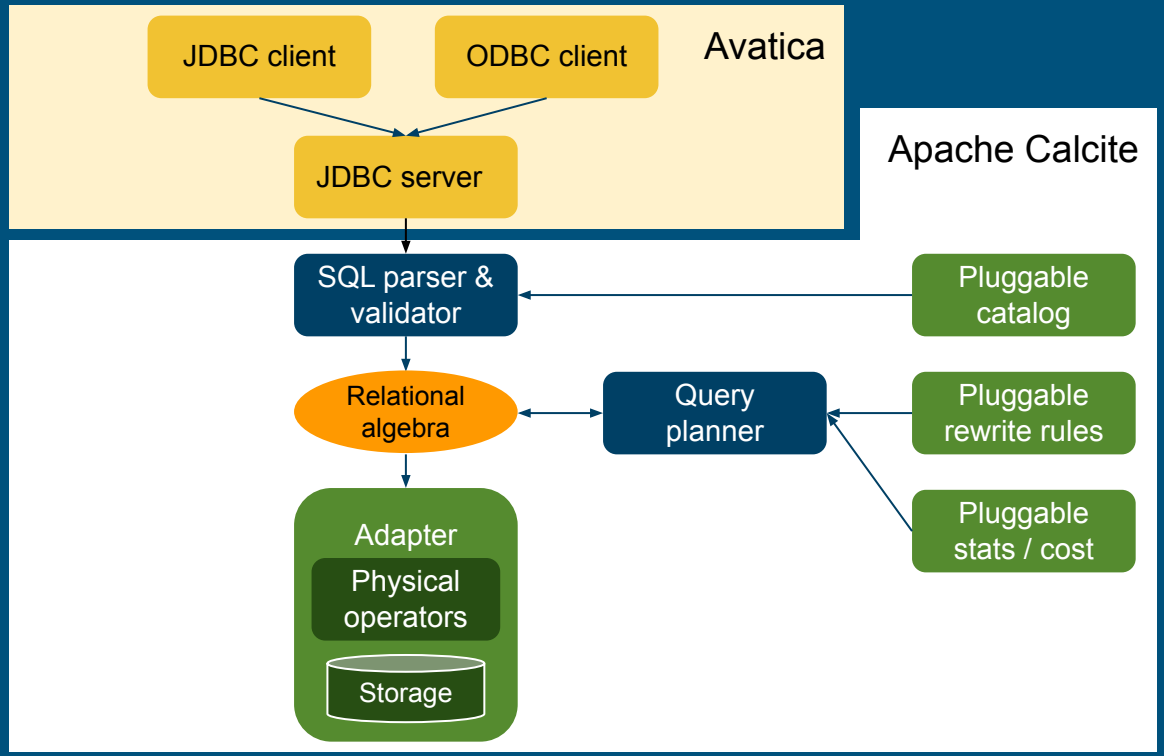
Calcite evolution - pluggable components



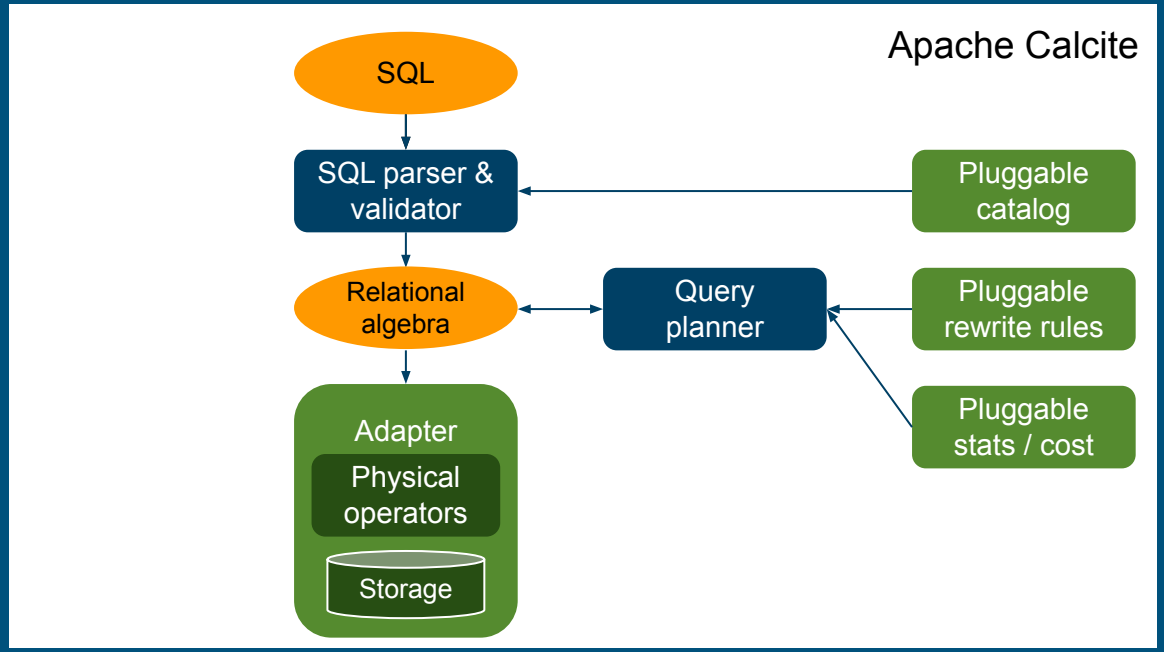
Calcite evolution - pluggable components



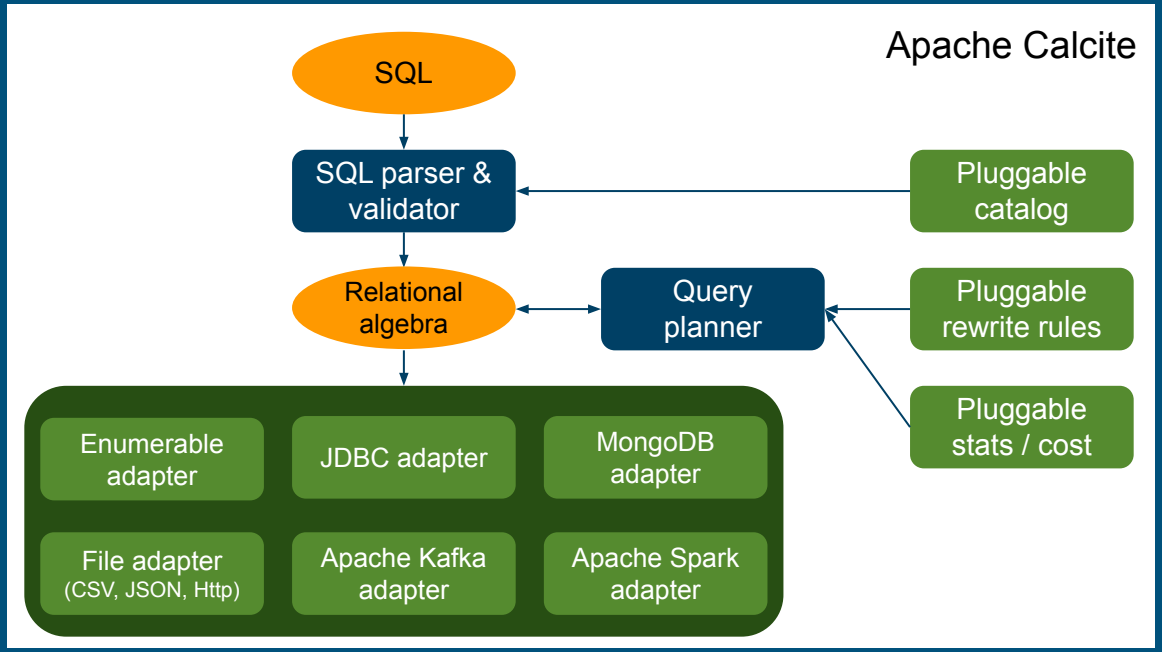
Calcite evolution - separate JDBC stack



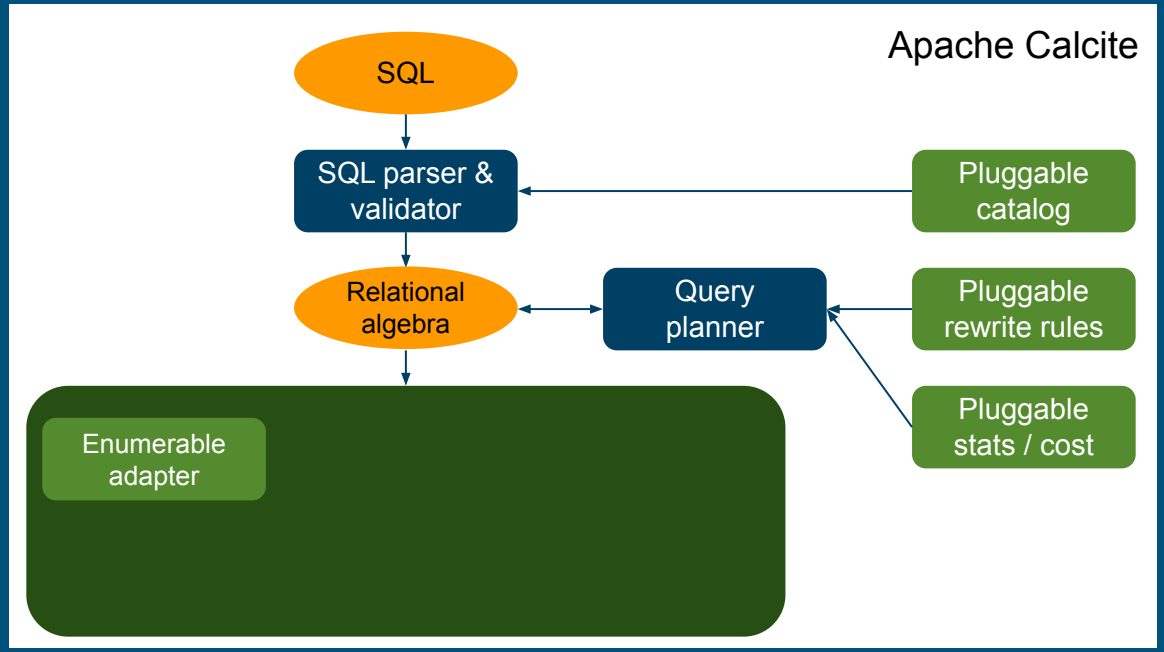
Calcite evolution - federation via adapters



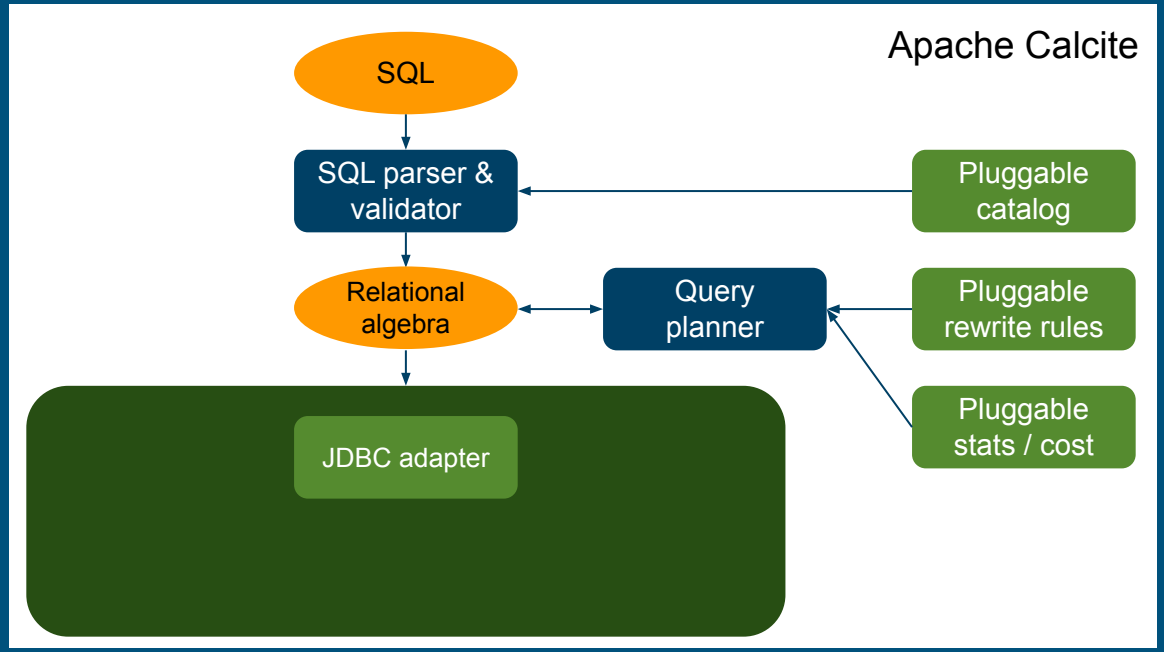
Calcite evolution - federation via adapters



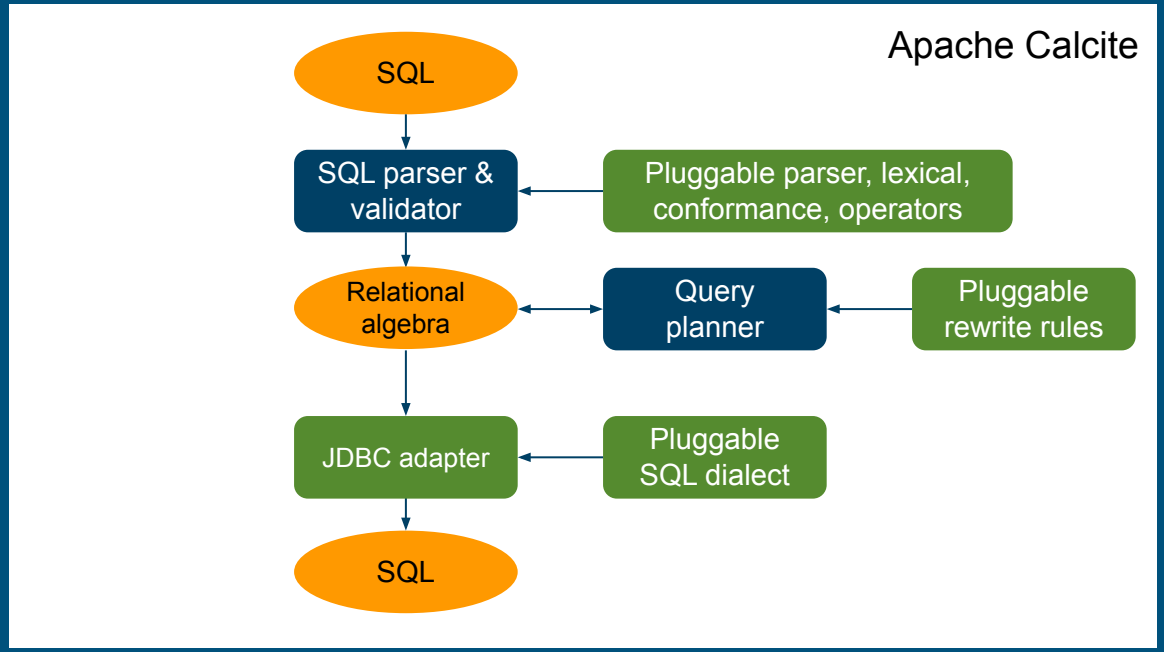
Calcite evolution - federation via adapters



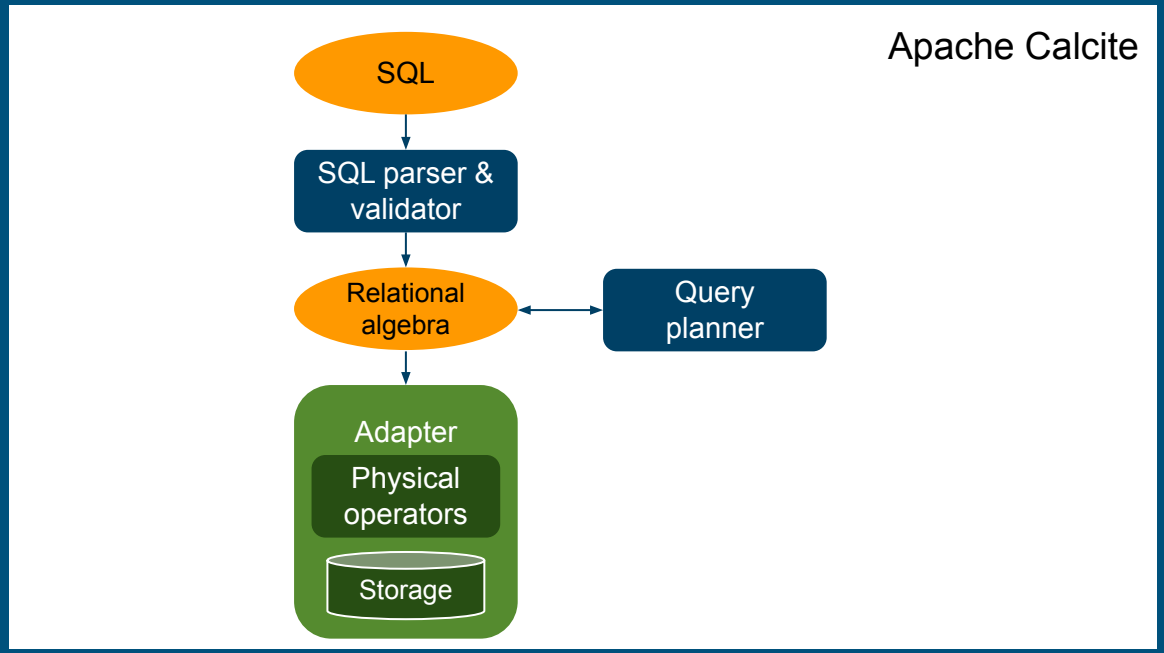
Calcite evolution - federation via adapters



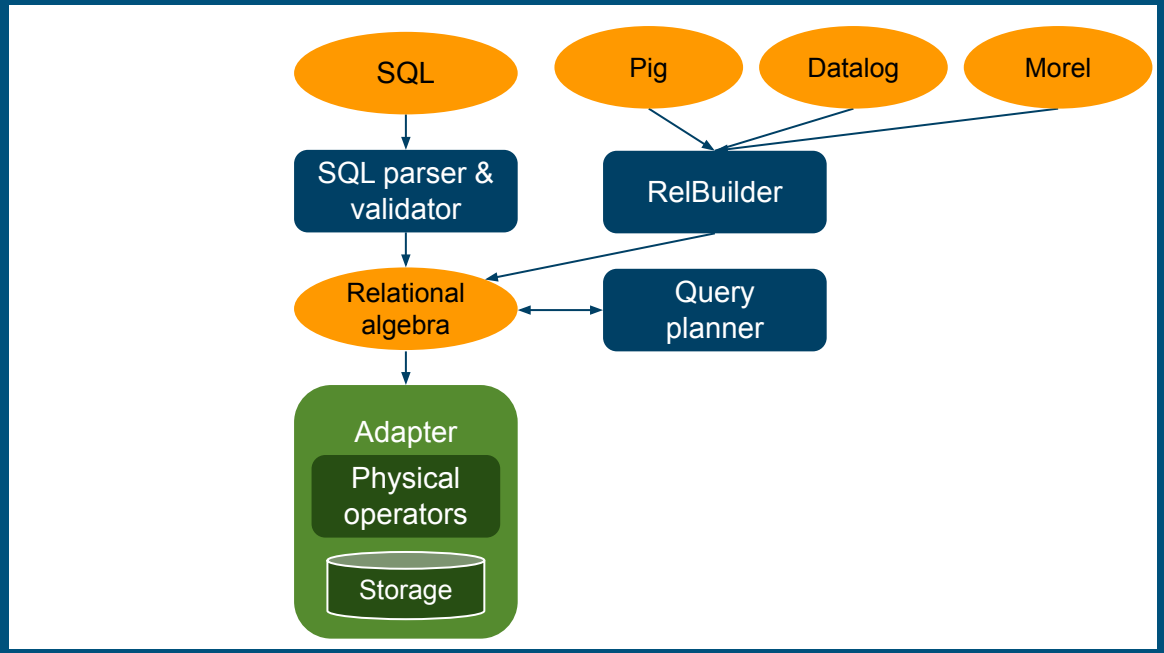
Calcite evolution - SQL dialects



Calcite evolution - other front-end languages



Calcite evolution - other front-end languages



3. Relational algebra

Relational algebra

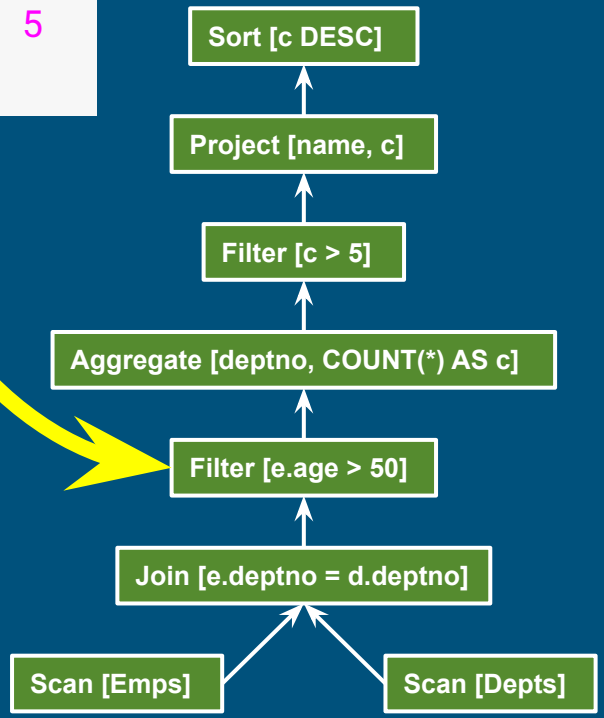
Based on set theory, plus operators: Project, Filter, Aggregate, Union, Join, Sort

Requires: declarative language (SQL), query planner

Original goal: data independence

Enables: query optimization, new algorithms and data structures

```
SELECT d.name, COUNT(*) AS c
FROM Emps AS e
JOIN Depts AS d USING (deptno)
WHERE e.age > 50
GROUP BY d.deptno
HAVING COUNT(*) > 5
ORDER BY c DESC
```



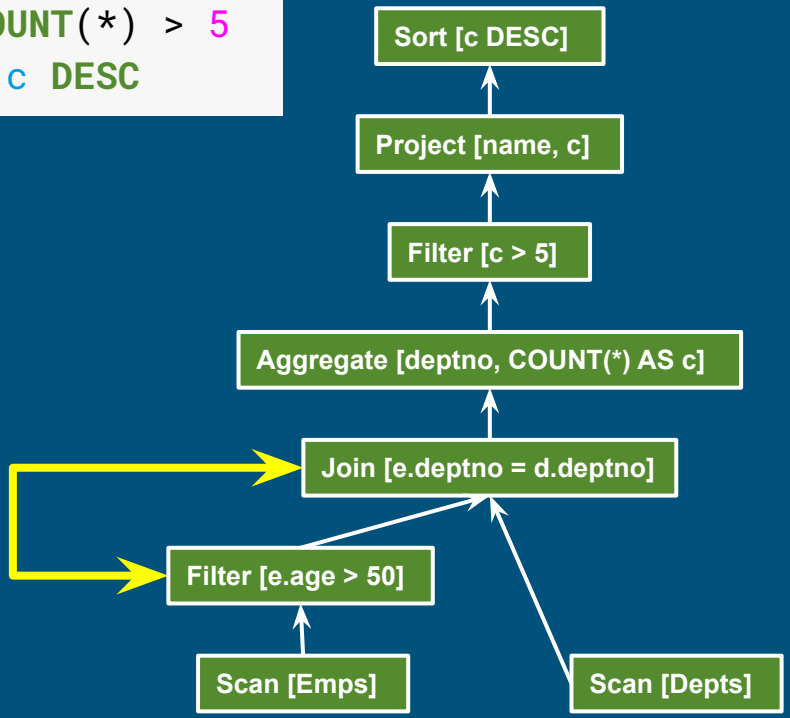
Algebraic rewrite

Optimize by applying rewrite rules that preserve semantics

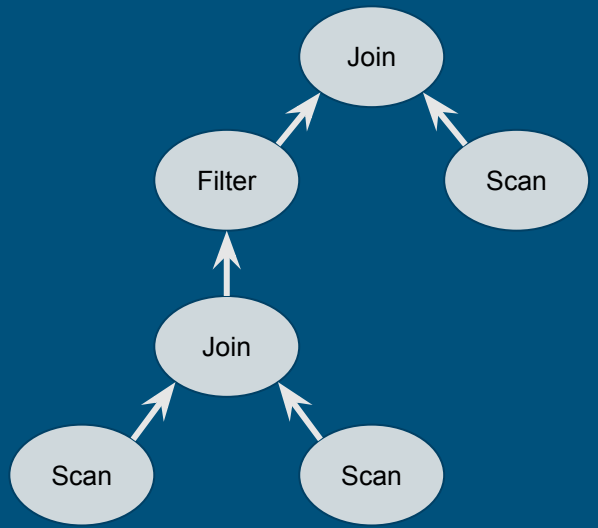
Hopefully the result is less expensive; but it's OK if it's not (planner keeps "before" and "after")

Planner uses dynamic programming, seeking the lowest total cost

```
SELECT d.name, COUNT(*) AS c
FROM (SELECT * FROM Emps
      WHERE e.age > 50) AS e
JOIN Depts AS d USING (deptno)
GROUP BY d.deptno
HAVING COUNT(*) > 5
ORDER BY c DESC
```



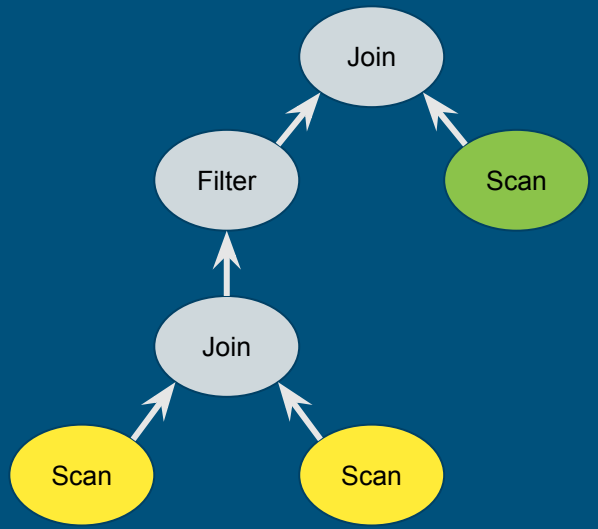
Calling convention



Initially all nodes belong to “logical” calling convention

Logical calling convention cannot be implemented, so has infinite cost

Calling convention

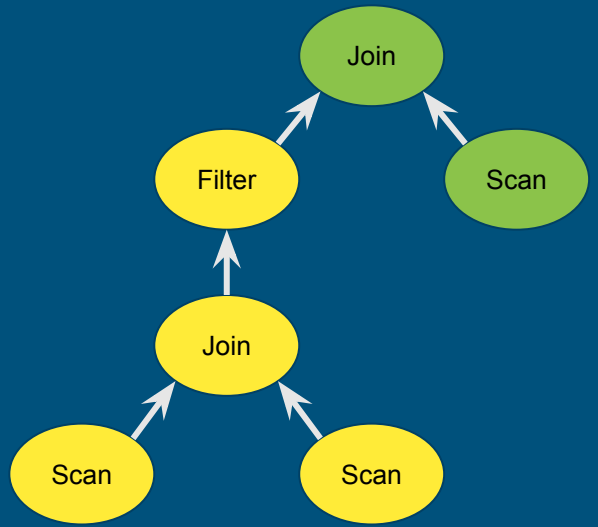


Tables can’t be moved so there is only one choice of calling convention for each table

Examples:

- Enumerable
- Druid
- Drill
- HBase
- JDBC

Calling convention



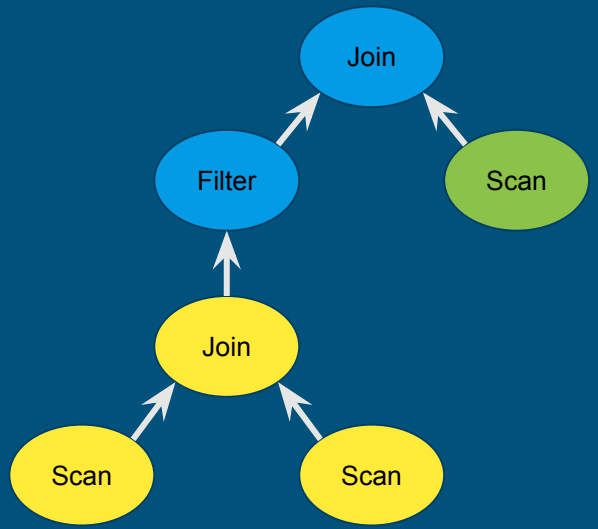
Rules fire to convert nodes to particular calling conventions

The calling convention propagates through the tree

LogicalFilter(YellowJoin) → YellowFilter(YellowJoin)

Because this is Volcano, each node can have multiple conventions

Calling convention



We also consider “engines” -- calling conventions that do not have a storage format.

Examples are Drill, Spark, Presto.

To implement, we generate program that calls out to query1 and query2.

Adapter

Implement `SchemaFactory` interface

Connect to a data source using parameters

Extract schema - return a list of tables

Push down processing to the data source:

- A set of planner rules
- Calling convention (optional)
- Query model & query generator (optional)

```
"schemas": [
 {
   "name": "HR",
   "type": "custom",
   "factory":
"org.apache.calcite.adapter.file.FileSchemaFactory",
   "operand": {
     "directory": "hr-csv"
   }
 }
]
```

```
$ ls -l hr-csv
-rw-r--r--  1 jhyde  staff   62 Mar 29 12:57 DEPTS.csv
-rw-r--r--  1 jhyde  staff  262 Mar 29 12:57 EMPS.csv.gz
$ ./sqlline  -u jdbc:calcite:model=hr.json -n scott -p tiger
sqlline> select count(*) as c from emp;
'C'
'5'
1 row selected (0.135 seconds)
```

Algebra builder

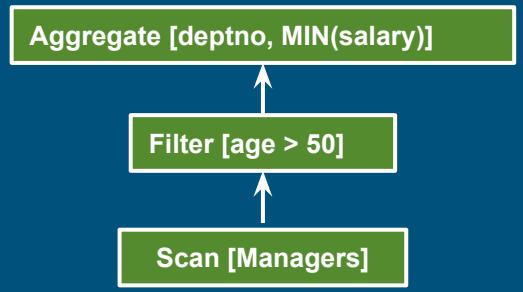
```
final FrameworkConfig config;
final RelBuilder builder = RelBuilder.create(config);
final RelNode node = builder
    .scan("EMPS")
    .aggregate(
        builder.groupKey("DEPTNO"), builder.count(false, "C"),
        builder.sum(false, "S", builder.field("SAL")))
    .filter(
        builder.call(SqlStdOperatorTable.GREATER_THAN,
            builder.field("C"), builder.literal(10)))
    .build();
System.out.println(RelOptUtil.toString(node));
```

```
SELECT deptno,
  COUNT(*) AS c,
  SUM(sal) AS s
FROM Emps
HAVING COUNT(*) > 10
```

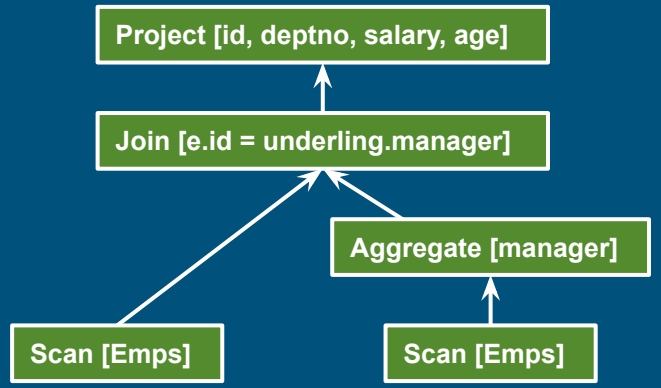
```
LogicalFilter(condition=[>($1, 10)])
    LogicalAggregate(group=[{7}], C=[COUNT()], S=[SUM($5)])
        LogicalTableScan(table=[[EMPS]])
```

Views

```
SELECT deptno, MIN(salary)
FROM Managers
WHERE age > 50
GROUP BY deptno
```



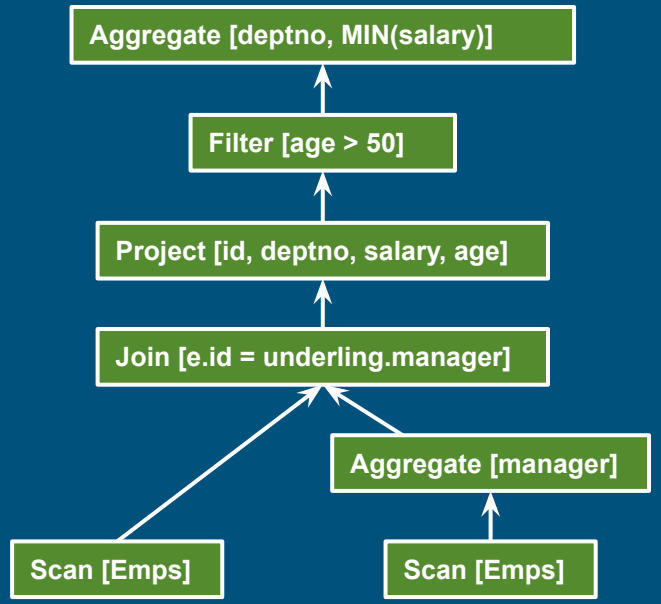

```
CREATE VIEW Managers AS
SELECT *
FROM Emps AS e
WHERE EXISTS (
  SELECT *
  FROM Emps AS underling
  WHERE underling.manager = e.id)
```



View query (after expansion)

```
SELECT deptno, MIN(salary)
FROM Managers
WHERE age > 50
GROUP BY deptno
```

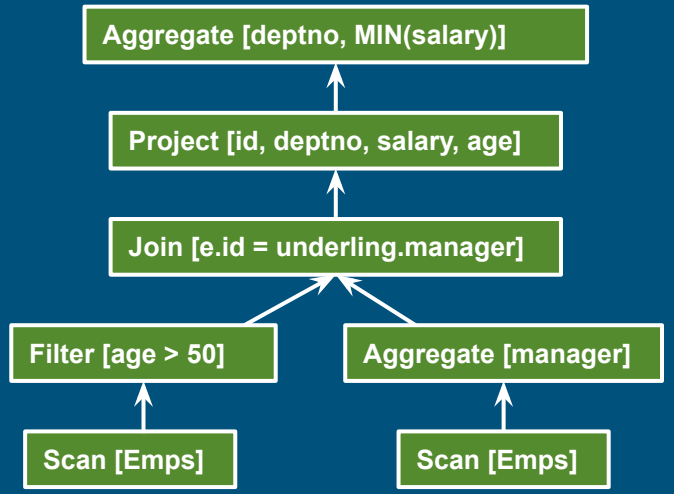
```
CREATE VIEW Managers AS
SELECT *
FROM Emps AS e
WHERE EXISTS (
  SELECT *
  FROM Emps AS underling
  WHERE underling.manager = e.id)
```



View query (after pushing down filter)

```
SELECT deptno, MIN(salary)
FROM Managers
WHERE age > 50
GROUP BY deptno
```

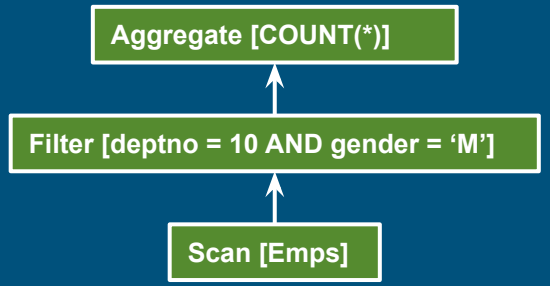
```
CREATE VIEW Managers AS
SELECT *
FROM Emps AS e
WHERE EXISTS (
  SELECT *
  FROM Emps AS underling
  WHERE underling.manager = e.id)
```



Materialized view

```
CREATE MATERIALIZED VIEW
  EmpSummary AS
SELECT deptno, gender,
  COUNT(*) AS c, SUM(sal) AS s
FROM Emps
GROUP BY deptno, gender
```

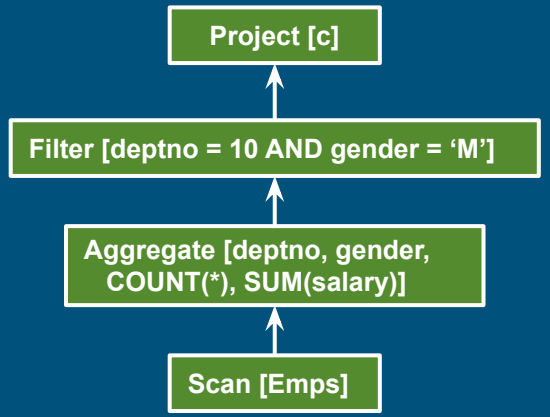

```
SELECT COUNT(*) AS c
FROM Emps
WHERE deptno = 10
AND gender = 'M'
```



Materialized view: rewrite query to match

```
CREATE MATERIALIZED VIEW
  EmpSummary AS
SELECT deptno, gender,
  COUNT(*) AS c, SUM(sal) AS s
FROM Emps
GROUP BY deptno, gender
```

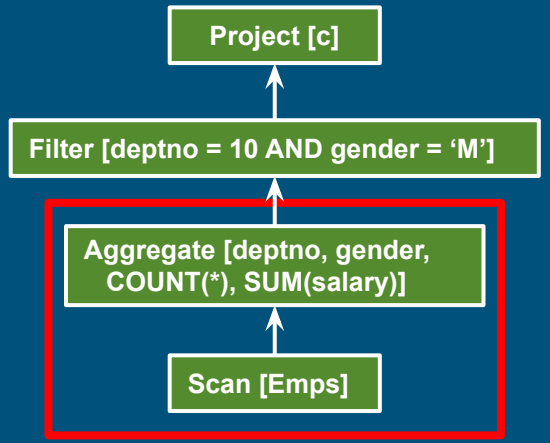

```
SELECT COUNT(*) AS c
FROM Emps
WHERE deptno = 10
AND gender = 'M'
```



Materialized view: rewrite query to match

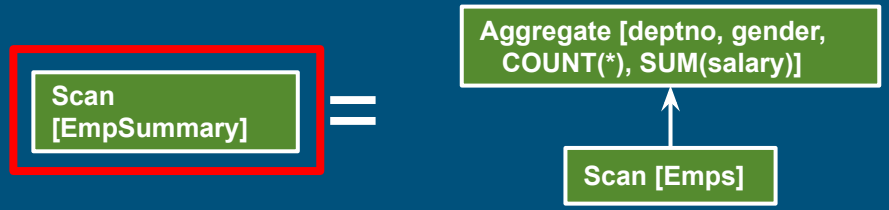
```
CREATE MATERIALIZED VIEW
  EmpSummary AS
SELECT deptno, gender,
  COUNT(*) AS c, SUM(sal) AS s
FROM Emps
GROUP BY deptno, gender
```


```
SELECT COUNT(*) AS c
FROM Emps
WHERE deptno = 10
AND gender = 'M'
```

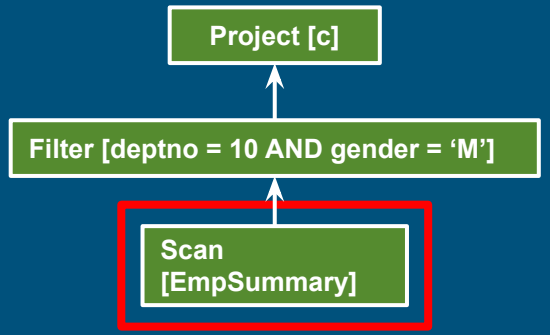


Materialized view: substitute table scan

```
CREATE MATERIALIZED VIEW
  EmpSummary AS
SELECT deptno, gender,
  COUNT(*) AS c, SUM(sal) AS s
FROM Emps
GROUP BY deptno, gender
```



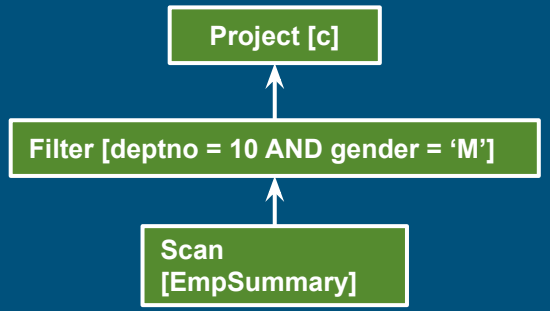

```
SELECT COUNT(*) AS c
FROM Emps
WHERE deptno = 10
AND gender = 'M'
```



Materialized view: substitute table scan

```
CREATE MATERIALIZED VIEW
  EmpSummary AS
SELECT deptno, gender,
  COUNT(*) AS c, SUM(sal) AS s
FROM Emps
GROUP BY deptno, gender
```


```
SELECT c
FROM EmpSummary
WHERE deptno = 10
AND gender = 'M'
```



Summary

Apache Calcite is a toolkit for **constructing your own DBMS**

Core is **Relational Algebra**

Optimize queries by applying **transformation rules** and choosing the best based upon **statistics**

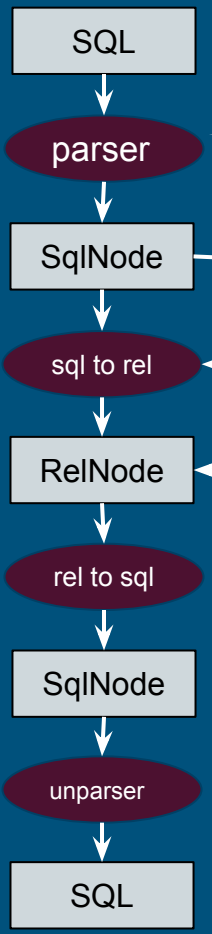
Calling conventions allow you to represent hybrid queries

Materialized views allow you optimize your data by providing sorted, aggregate copies

Thank you!
Questions?

https://calcite.apache.org
@ApacheCalcite
@julianhyde

Custom dialects



Sub-class parser (core, ddl, server, babel)

Parser lexical config (e.g. identifier quoting, casing, literal quoting)

Conformance, e.g. `isGroupByOrdinal()`

validator

Operator table (e.g. `fun=oracle,postgresql`)

typing

Type system (e.g. default precision of `TIMESTAMP`)

Desugaring (e.g. `NVL` → `CASE`, `USING` → `ON`)

planner

Relational transformations (e.g. decorrelation, push-down filters, join ordering, predicate-based expression simplification, materialized view selection)

Dialect-specific relational transformations (e.g. `ORDER BY x NULLS LAST` → `ORDER BY x IS NULL, x`)

Dialect (e.g. `SqlDialect.unparserIdentifier(id)`)