# Open Source SQL - beyond parsers: ZetaSQL & Apache Calcite

Northwest Database Society Annual Meeting
2021/01/20

Mosha Pasumansky & Julian Hyde (Google)

# Apache Calcite goals

Make it easier to write a simple DBMS

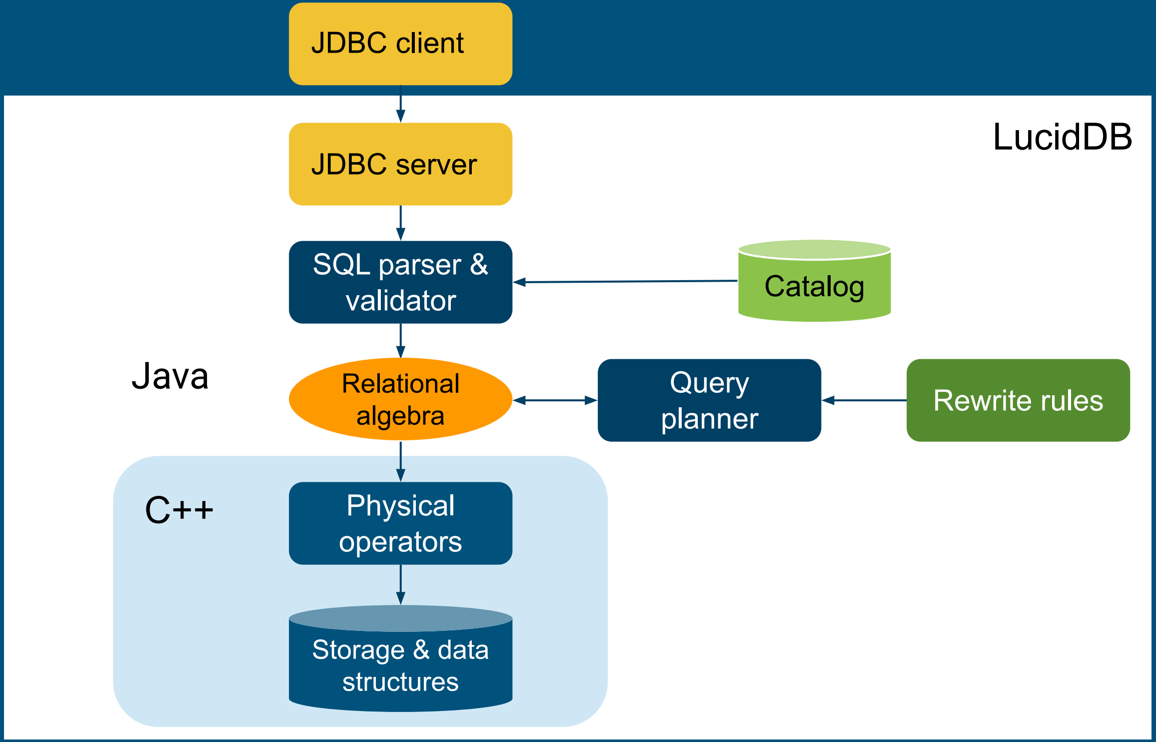
Advance the state of the art for complex DBMS

Bring database approaches to new areas (e.g. streaming, geospatial, federation, data science)

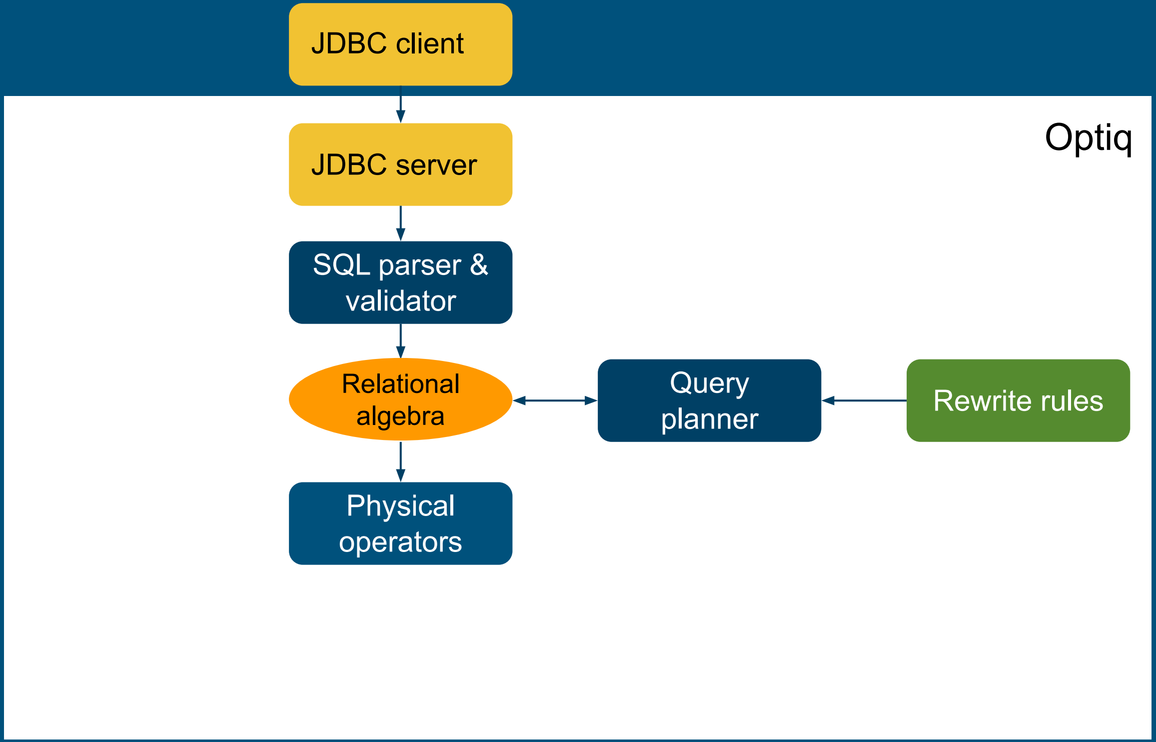
Composition + evolution (framework + open source)

Apache license & governance

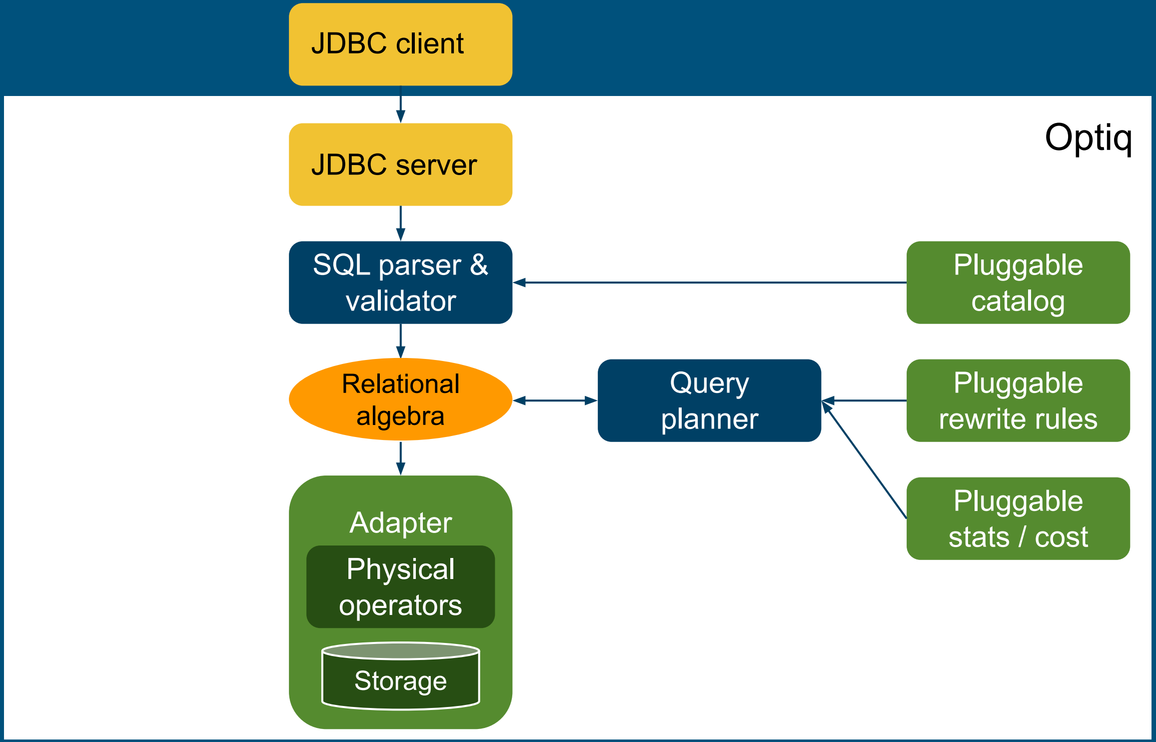
# Calcite evolution - origins as an SMP DB



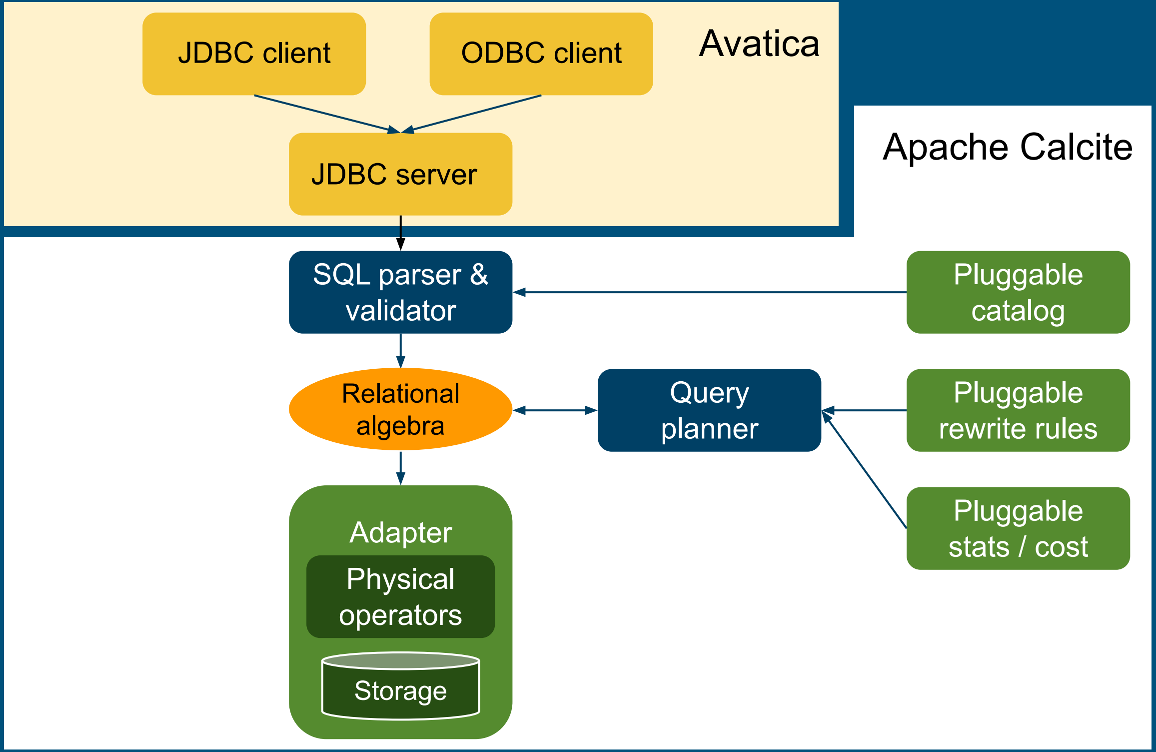
# Calcite evolution - pluggable components



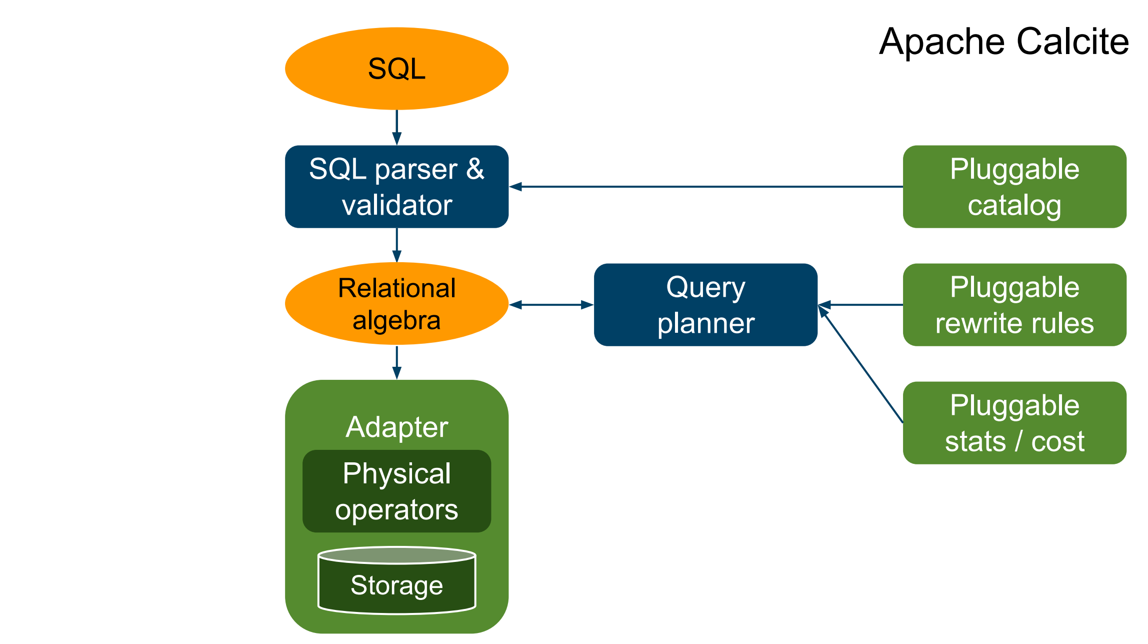
# Calcite evolution - pluggable components



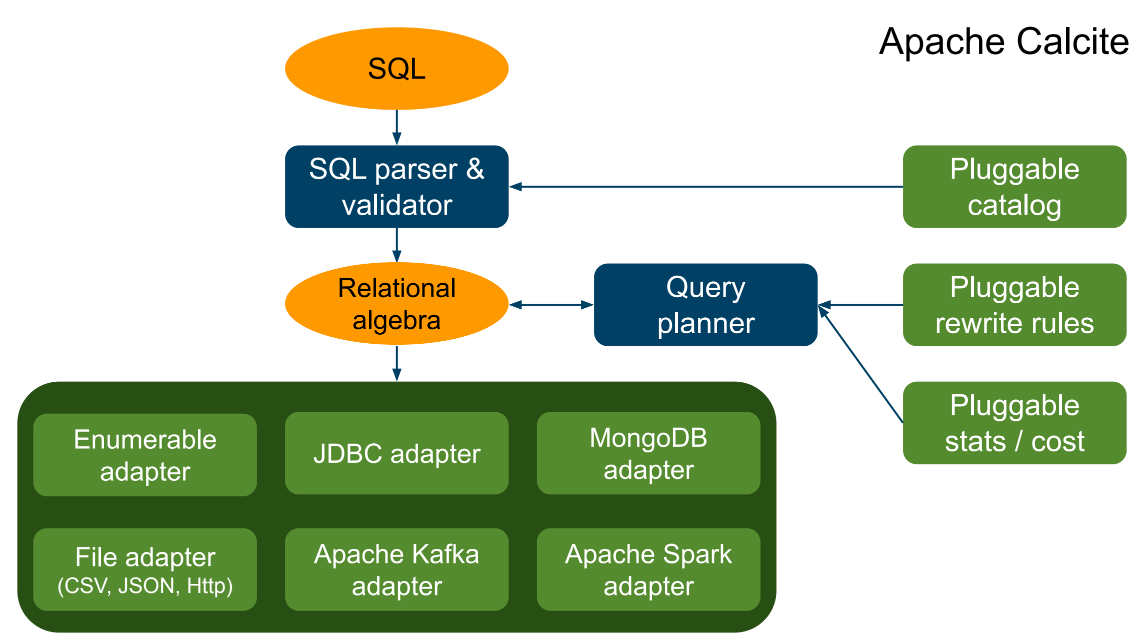
# Calcite evolution - separate JDBC stack



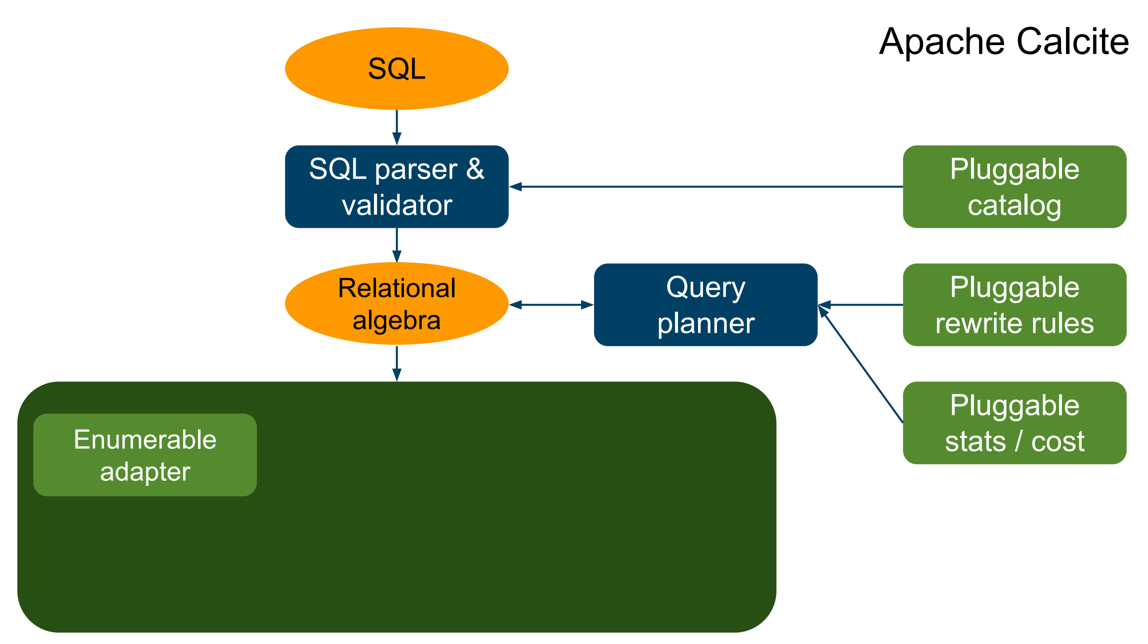
# Calcite evolution - federation via adapters



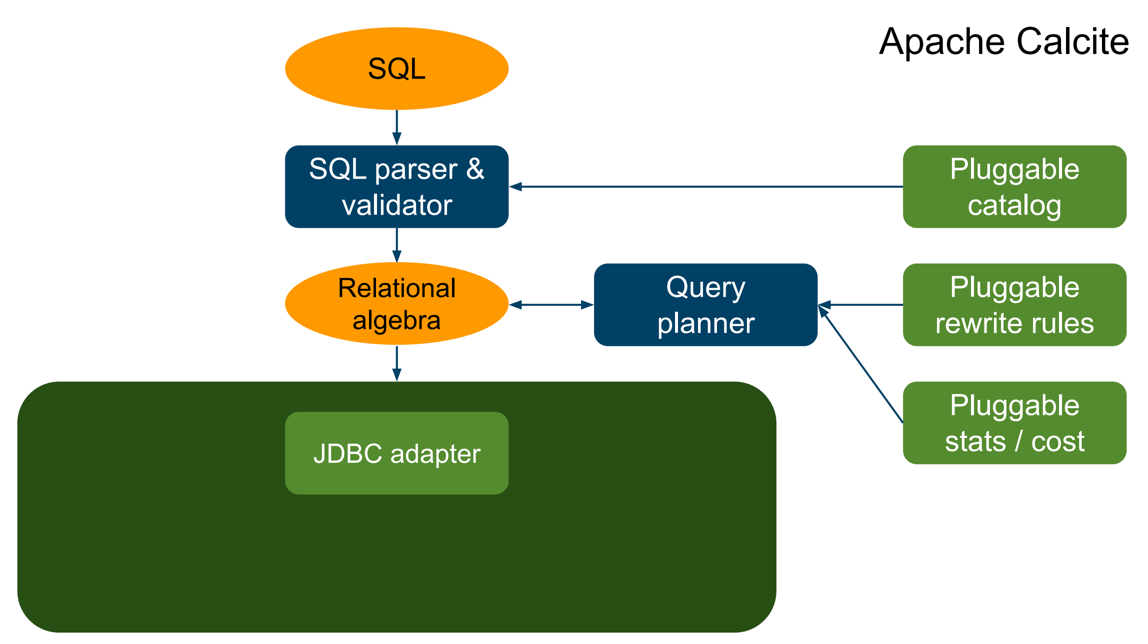
# Calcite evolution - federation via adapters



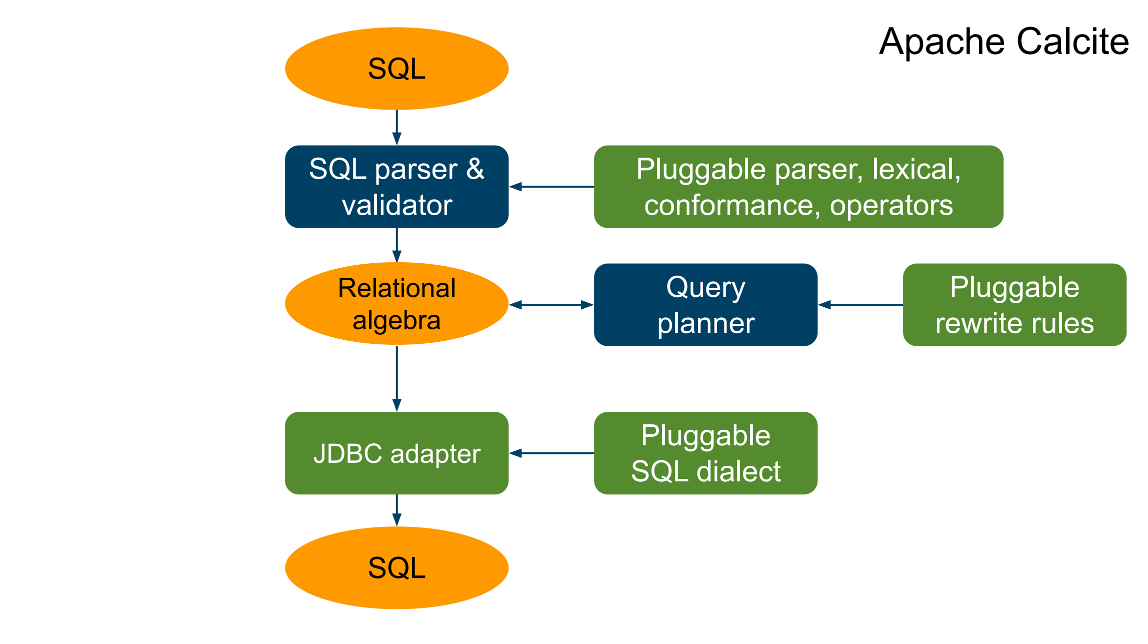
# Calcite evolution - federation via adapters



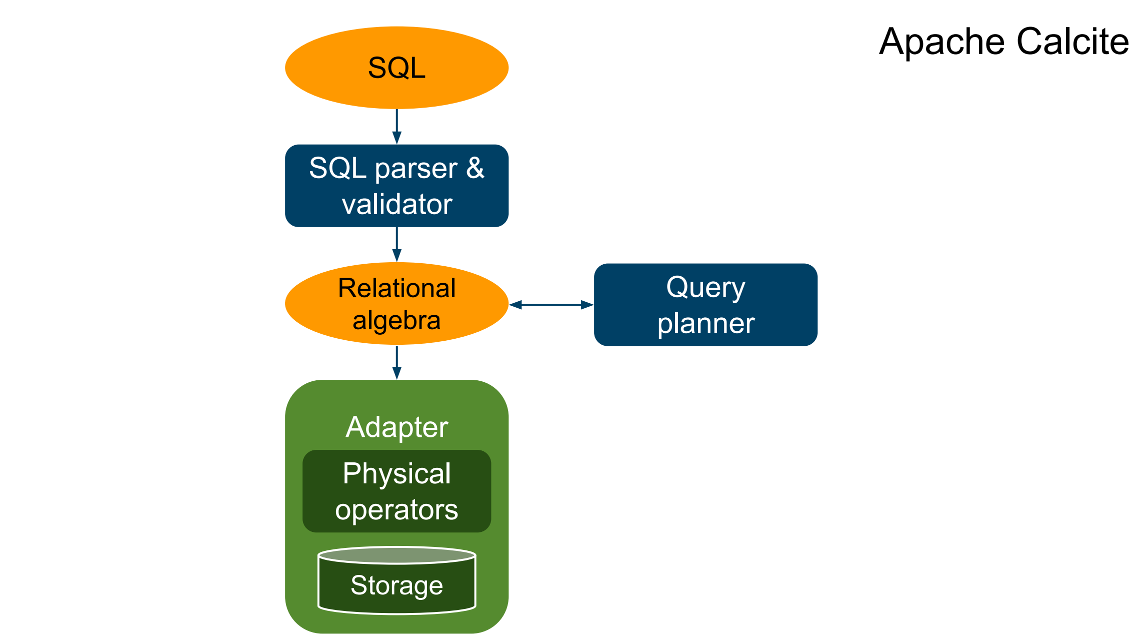
# Calcite evolution - federation via adapters



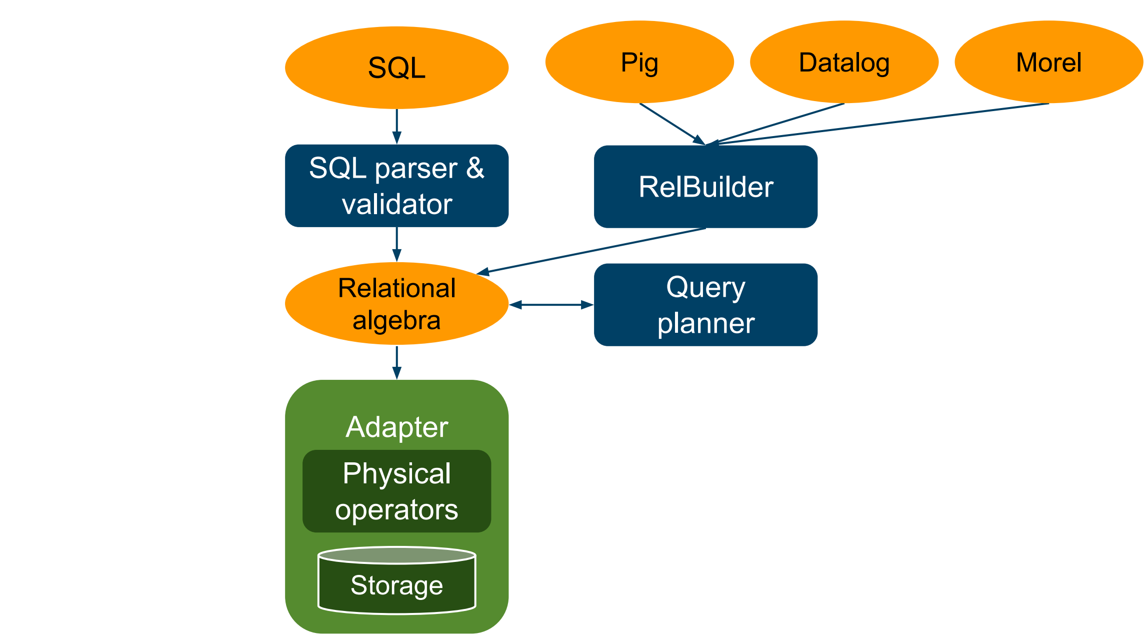
# Calcite evolution - SQL dialects



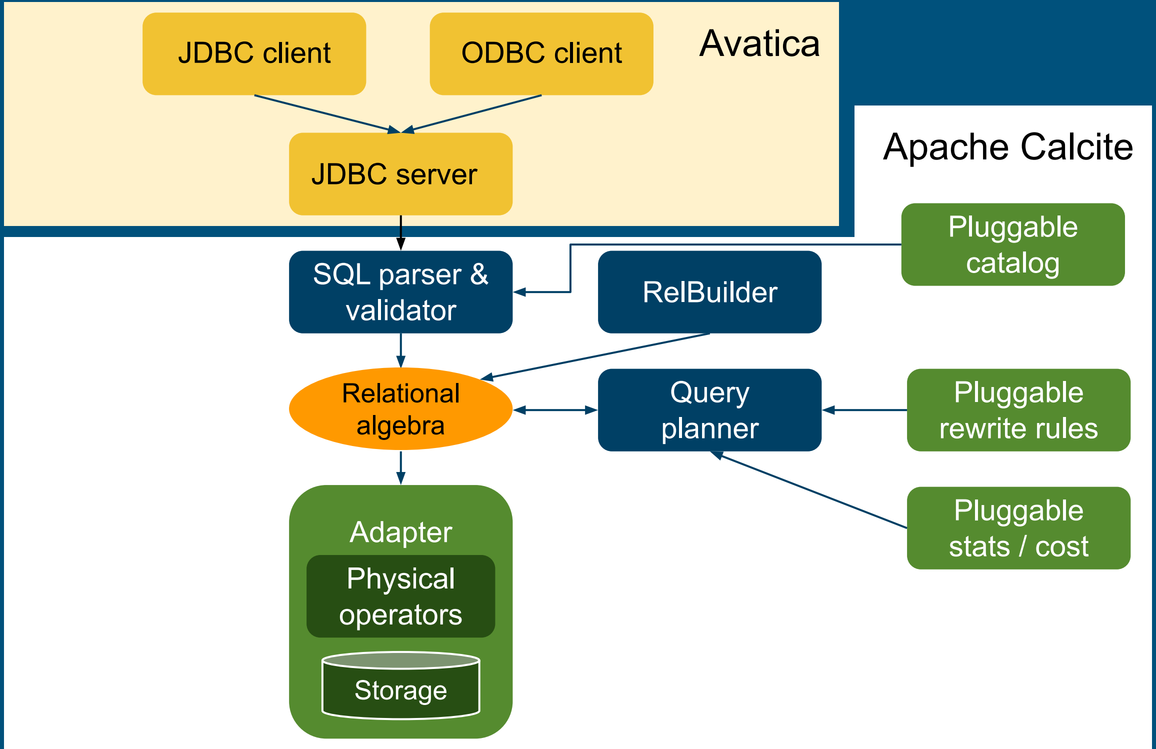
# Calcite evolution - other front-end languages



# Calcite evolution - other front-end languages



# Calcite architecture




**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

# One SQL to Rule Them All – an Efficient and Syntactically Idiomatic Approach to Management of Streams and Tables

An Industrial Paper


Edmon Begoli
Oak Ridge National Laboratory / Apache Calcite
Oak Ridge, Tennessee, USA
begoli@apache.org

Tyler Akidau
Google Inc. / Apache Beam
Seattle, WA, USA
takidau@apache.org

Fabian Hueske
Ververica / Apache Flink
Berlin, Germany
fhueske@apache.org

Julian Hyde
Looker Inc. / Apache Calcite
San Francisco, California, USA
jhyde@apache.org

Kathryn Knight
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
knightke@ornl.gov

Kenneth Knowles
Google Inc. / Apache Beam
Seattle, WA, USA
kenn@apache.org



## ABSTRACT

Real-time data analysis and management are increasingly critical for today's businesses. SQL is the de facto *lingua franca* for these endeavors, yet support for robust streaming analysis and management with SQL remains limited. Many approaches restrict semantics to a reduced subset of features and/or require a suite of non-standard constructs. Additionally, use of event timestamps to provide native support for analyzing events according to when they actually occurred is not pervasive, and often comes with important limitations.

We present a three-part proposal for integrating robust streaming into the SQL standard, namely: (1) time-varying relations as a foundation for classical tables as well as streaming data, (2) event time semantics, (3) a limited set of optional keyword extensions to control the materialization of time-varying query results. Motivated and illustrated using exam-



## CCS CONCEPTS

• **Information systems → Stream management**; **Query languages**;

## KEYWORDS

stream processing, data management, query processing

**ACM Reference Format:**

Edmon Begoli, Tyler Akidau, Fabian Hueske, Julian Hyde, Kathryn Knight, and Kenneth Knowles. 2019. One SQL to Rule Them All – an Efficient and Syntactically Idiomatic Approach to Management of Streams and Tables: An Industrial Paper. In *2019 International Conference on Management of Data (SIGMOD '19), June 30-July 5, 2019, Amsterdam, Netherlands.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3299869.3314040

# Tempura: A General Cost-Based Optimizer Framework for Incremental Data Processing

Zuozhi Wang[1], Kai Zeng[2], Botong Huang[2], Wei Chen[2], Xiaozong Cui[2], Bo Wang[2], Ji Liu[2], Liya Fan[2], Dachuan Qu[2], Zhenyu Hou[2], Tao Guan[2], Chen Li[1], Jingren Zhou[2]

[1]University of California, Irvine [2]Alibaba Group

[1] Irvine, United States [2] Hangzhou, China

[1]{zuozhiw, chenli}@ics.uci.edu, [2]{zengkai.zk, botong.huang, wickeychen.cw, xiaozong.cxz, yanyu.wb, niki.lj, liya.fly, dachuan.qdc, zhenyuhou.hzy, tony.guan, jingren.zhou}@alibaba-inc.com

## ABSTRACT

Incremental processing is widely-adopted in many applications, ranging from incremental view maintenance, stream computing, to recently emerging progressive data warehouse and intermittent query processing. Despite many algorithms developed on this topic, none of them can produce an incremental plan that always achieves the best performance, since the optimal plan is data dependent. In this paper, we develop a novel cost-based optimizer framework, called `Tempura`, for optimizing incremental data processing. We propose an incremental query planning model called TIP based on the concept of time-varying relations, which can formally model incremental processing in its most general form. We give a full specification of `Tempura`, which can not only unify various existing techniques to generate an optimal incremental plan, but also allow the developer to add their rewrite rules. We study how to explore the plan space and search for an optimal incremental plan. We evaluate `Tempura` in various incremental processing scenarios to show its effectiveness and efficiency.

**PVLDB Reference Format:**

Zuozhi Wang, Kai Zeng, Botong Huang, Wei Chen, Xiaozong Cui, Bo Wang, Ji Liu, Liya Fan, Dachuan Qu, Zhenyu Hou, Tao Guan, Chen Li, Jingren Zhou. Tempura: A General Cost-Based Optimizer Framework for Incremental Data Processing. PVLDB, 14(1): 14-27, 2021.

doi:10.14778/3421424.3421427

the adoption of the incremental processing model. Here are a few examples of emerging applications.

**Progressive Data Warehouse [45].** Enterprise data warehouses usually have a large amount of automated routine analysis jobs, which have a stringent schedule and deadline determined by various business logic. For example, at Alibaba, daily report queries are scheduled after 12 am when the previous day's data has been fully collected, and the results must be delivered by 6 am sharp before the bill-settlement time. These routine analysis jobs are predominately handled using batch processing, causing dreadful "rush hour" scheduling patterns. This approach puts pressure on resources during traffic hours, and leaves the resources over-provisioned and wasted during the off-traffic hours. Incremental processing can answer routine analysis jobs progressively as data gets ingested, and its scheduling flexibility can be used to smoothen the resource skew.

**Intermittent Query Processing [40].** Many modern applications require querying an incomplete dataset with the remaining data arriving in an intermittent yet predictable way. Intermittent query processing can leverage incremental processing to balance latency for maintaining standing queries and resource consumption by exploiting knowledge of data-arrival patterns. For instance, when querying dirty data, the data is usually first cleaned and then fed into a database. The data cleaning step can quickly spill the clean data but needs to conduct a time-consuming processing on the dirty data. Intermittent query processing can use incremental processing to quickly deliver informative but partial results to the

# Lessons learned

Decompose the database into components

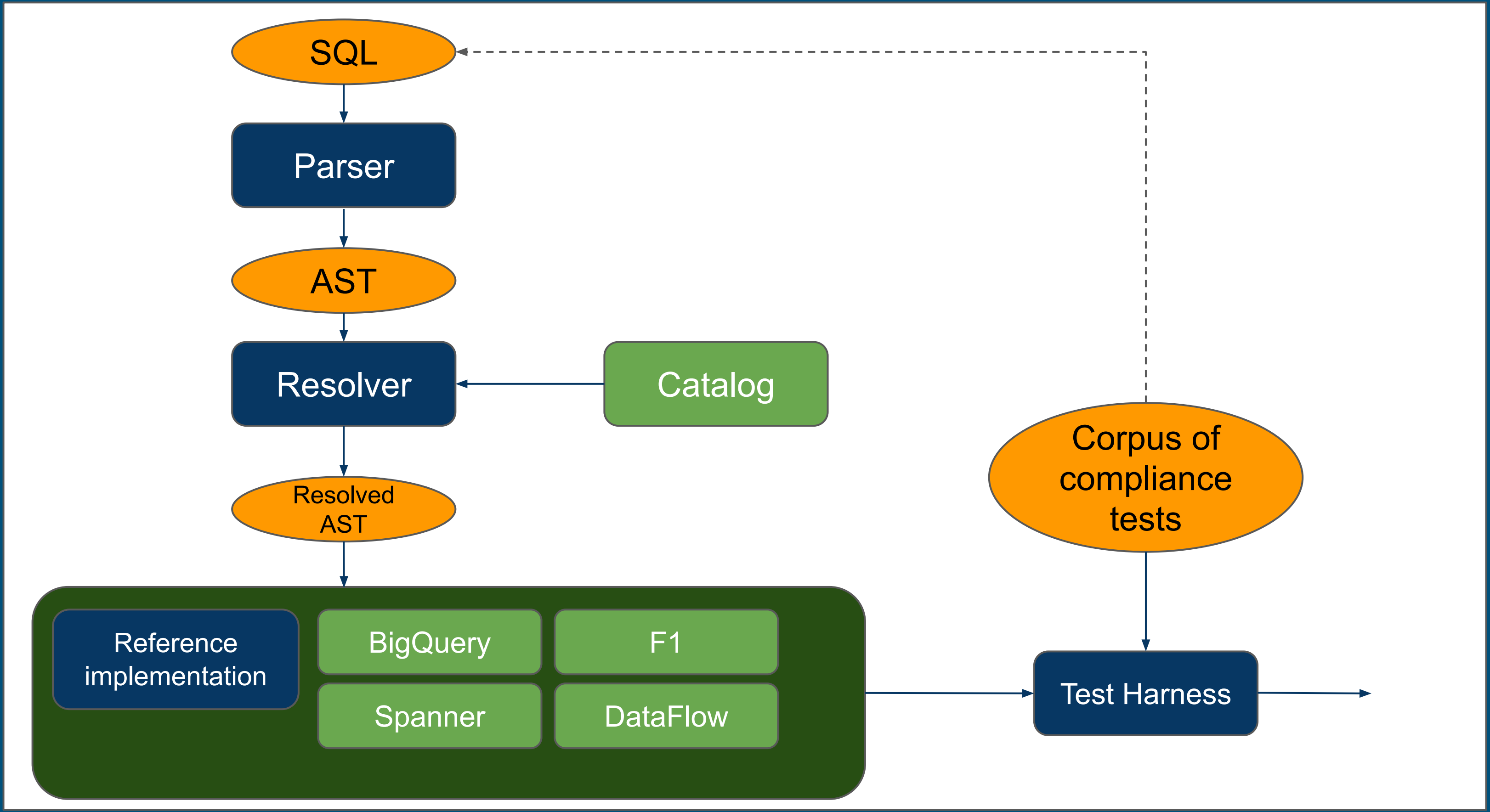
SQL is standard but also allows innovation

Relational algebra intermediate language

Calcite has many uses, including:

- Embedded within DBMS (e.g. Apache Hive, OmniSciDB)
- Lightweight DBMS
- Platform for research
- Sandbox for relational algebra
- Toolkit for translating between SQL dialects

# ZetaSQL


SQL
Parser
AST
Resolver
Catalog
Resolved AST
Corpus of compliance tests
Reference implementation
BigQuery
F1
Spanner
DataFlow
Test Harness

# Thank you!
# Questions?

#ZetaSQL
https://github.com/google/zetasql

@ApacheCalcite
https://calcite.apache.org