ShardingSphere 联邦查询引擎演进与实战

端正强

duanzhengqiang@apache.org

目录

ShardingSphere 简介

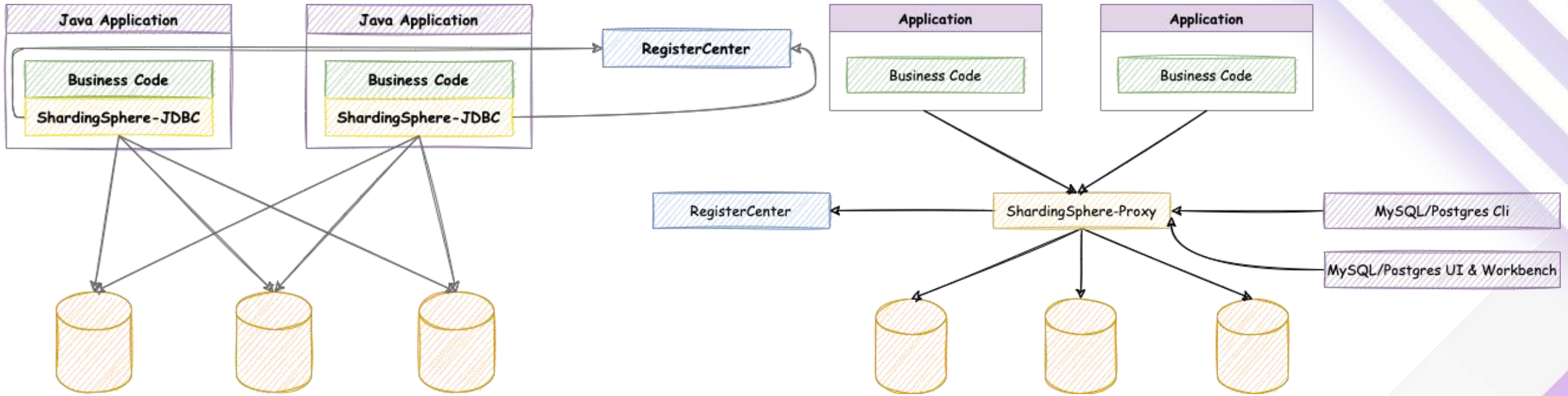
ShardingSphere 简介

ShardingSphere 是一款开源的「数据服务能力增强引擎」
提供 *数据分片、分布式事务、数据安全* 等能力。

ShardingSphere 遵循 Database Plus 理念
旨在构建「异构数据库」上层的「服务标准」和「生态」。

ShardingSphere

ShardingSphere 接入端



- ShardingSphere 由 JDBC 和 Proxy 2 个接入端组成，既支持「独立部署」，也可「混合部署」；
- JDBC 和 Proxy 接入端均提供「标准化的增量功能」：可适用于 *Java 同构、异构语言* 等各种应用场景；

ShardingSphere 混合部署

- ShardingSphere- JDBC 采用无中心化架构，与应用程序共享资源，适用于 Java 开发的高性能的「轻量级 OLTP 应用」
- ShardingSphere-Proxy 提供静态入口以及异构语言的支持，独立于应用程序部署。适用于 OLAP 应用以及对「分片数据库」进行管理和运维 的场景。

ShardingSphere 社区

THE APACHE SOFTWARE FOUNDATION

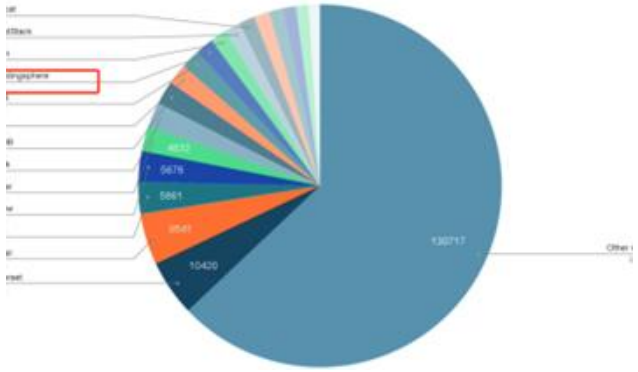
全球最顶级的开源软件基金会

管理超过两亿行代码

成功孵化 300+ 顶级开源项目

+

Projects by Number of Commits, 2021



- 17,000+ Stars
- 13,000+ Pull Requests
- 6,000+ Forks
- 400+ Contributors

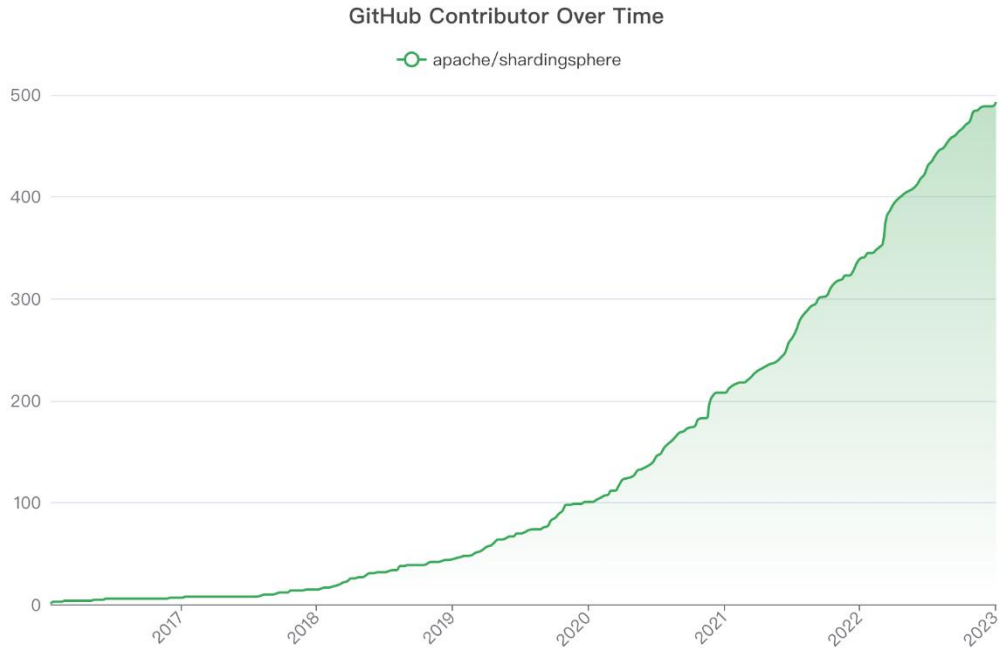
基金会认可 Apache 软件基金会顶级项目

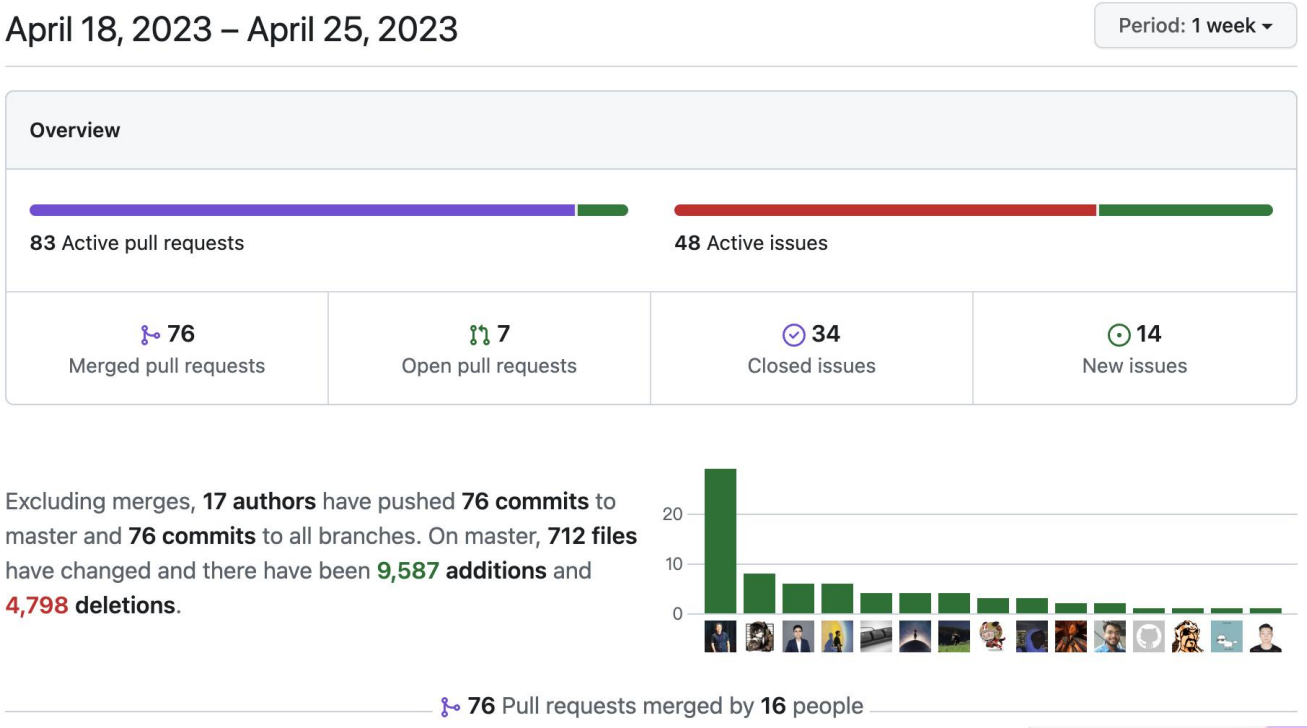
社区活跃度 2021 年度 Apache 基金会年度报告代码提交量位列前十

学术界认可 数据库顶会 **ICDE 2022 发表论文**《A Holistic and Pluggable Platform for Data Sharding》

出版**《Apache ShardingSphere 的权威指南》**，有力阐述了如何在多模型数据库之上构建开放生态

ShardingSphere 社区



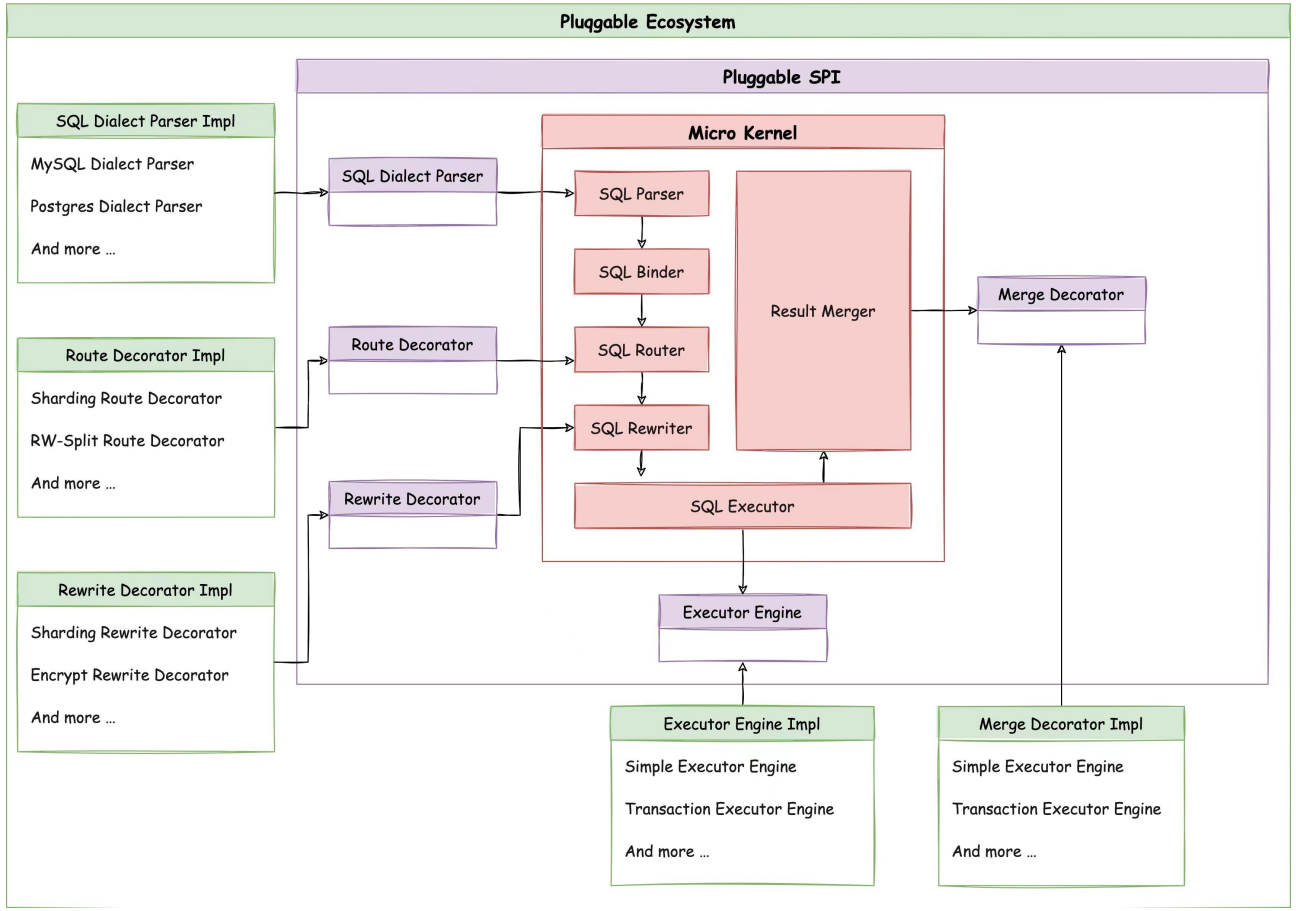


- 社区优于代码
- 尊重和信任合作伙伴
- 保持友好和开放

- 测试自动化
- 创建自动渠道
- 公开和远程的工作方式

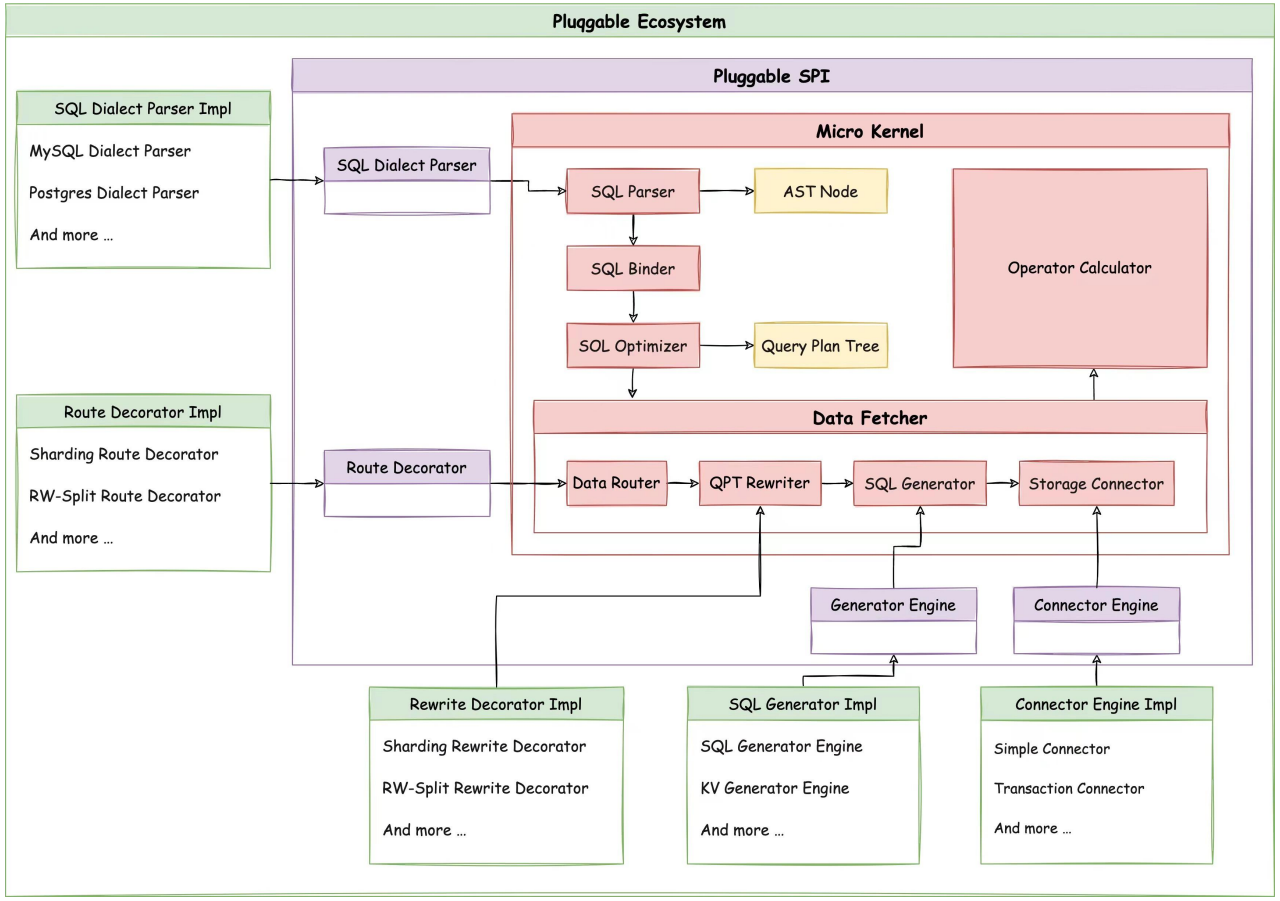
ShardingSphere 联邦查询引擎入门

为什么需要联邦查询引擎



- ShardingSphere 内核采用的是计算下推模型，通过 SQL 解析、SQL 路由、SQL 改写、SQL 执行和 SQL 归并几个阶段；
- 计算下推模型只能支持完美 Sharding 场景下的 SQL 执行，对于跨库 JOIN、子查询、复杂聚合查询等语句无法支持；
- 用户业务查询 SQL 复杂多变，通过业务 SQL 改造接入 ShardingSphere 成本较大；

联邦查询引擎原理



- SQL Optimizer：对跨库的关联查询及子查询进行 RBO 和 CBO 优化，得到最优执行计划；
- Data Fetcher：根据最优执行计划生成的 SQL 从存储节点获取数据，Data Fetcher 内部包含了对生成 SQL 的路由、改写和执行；
- Operator Calculator：根据最优执行计划以及从存储节点获取的数据进行算子计算，得到最终查询结果。

联邦查询引擎第一版

[New feature] Federated SQL query and Query optimization are going to sail out. #8284

Edit | New issue

Open | 24 of 33 tasks | **tristaZero** opened this issue on Nov 22, 2020 · 50 comments · Fixed by #11079

tristaZero commented on Nov 22, 2020 • edited | Member

Hi community,

As you know, ShardingSphere has made a lot of efforts on SQL parser and **provided a great independent SQL parser** to help users parse SQL.

Based on this substantial work, we **plan to do query optimization** to optimize the input SQLs from users and produce an optimized SQL query plan to improve query efficiency. Plus, **the federated SQL query feature** (Like join query from different instances) is another essential highlight for our next release. :)

We will leverage Apache Calcite, an excellent framework to implement two of the features. Currently, three main work focus are presented here.

- [x] The investigation of **RBO** (Rule-Based Optimization) and **CBO** (Cost-Based optimization) ,i.e., Hep planner engine and Volcano planner engine. (**Mechanism, usage, pros and cons**).
- [x] The investigation of Calcite adaptor, especially `TranslatableTable` API.
- [x] How to transform the parsed result of ShardingSphere to the algebra of Calcite
- [x] Call some of the optimization rules to process the relational expression
- [x] Implement Calcite adaptor to join SQLs from different instances
- [x] Combine sharding tables with Calcite adaptors.

Actually, there are plenty of works to do on this issue. We are **in the investigation phase** now and will seek contributors for this issue later. If you are interested in this one, please give it a watch. 😉

Assignees

guimingyue
strongduanmu
junwen12221
wgy8283335
tristaZero

Labels

in: kernel | type: new feature

Projects

None yet

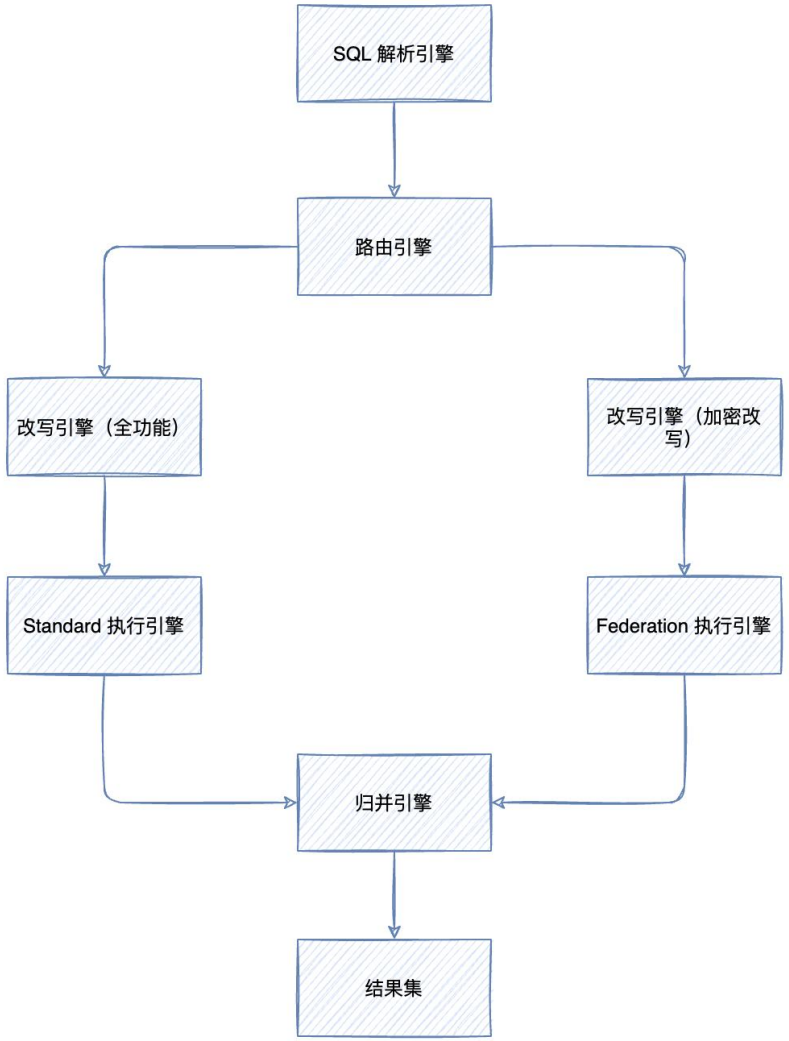
Milestone

No milestone

Development

Successfully merging a pull request may close this issue.

联邦查询引擎第一版



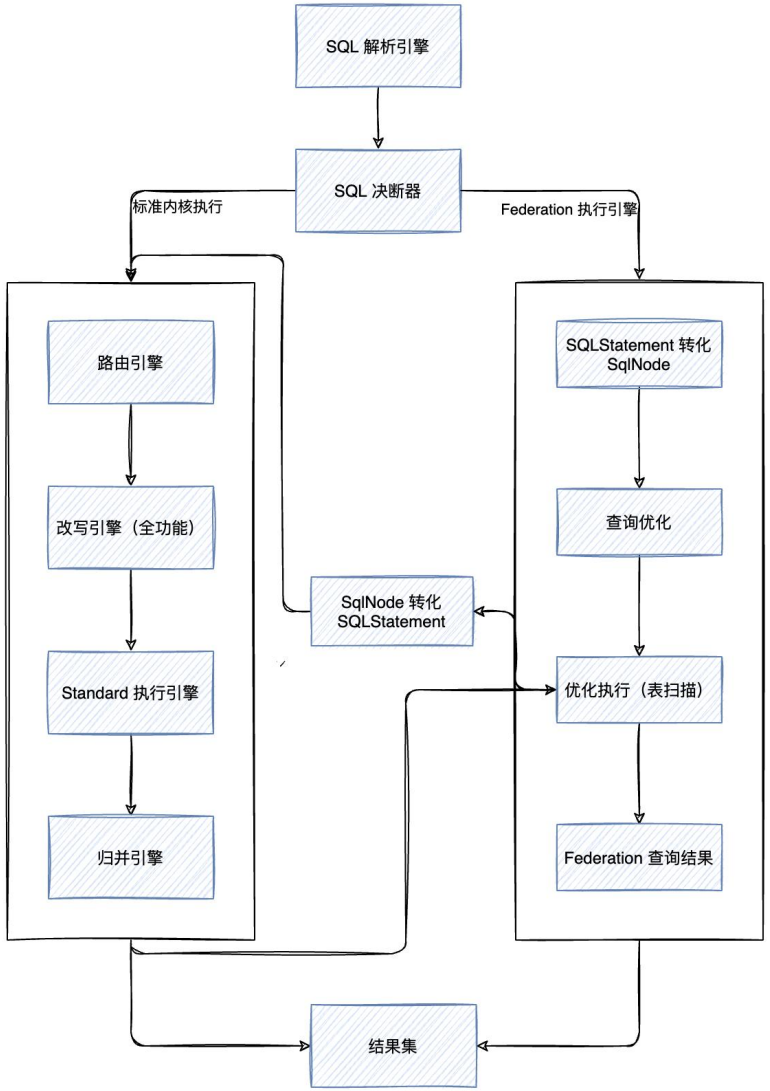
实现思路：

- 使用 Calcite JDBC 实现基础联邦查询，支持跨库查询、子查询、聚合查询等复杂 SQL；
- 使用真实元数据注册到 Calcite，Calcite 需要提前路由、改写为真实 SQL 再交由联邦查询引擎处理；
- SQL 生成器根据 ProjectableFilterableTable 接口参数生成 SQL 并执行；
- 通过归并引擎将真实结果集归并为逻辑结果集返回给用户。

存在的问题：

- 默认使用 Calcite 内置优化规则，没有集成优化模块 ShardingSphereOptimizer；
- 存在着两次解析问题（Calcite 解析和 ShardingSphere 解析）；
- 下推 SQL 生成后全路由，会导致在所有真实表中执行查询，当真实表数量非常多时，会导致查询效率降低；
- 两套元数据，维护成本高；

联邦查询引擎第二版



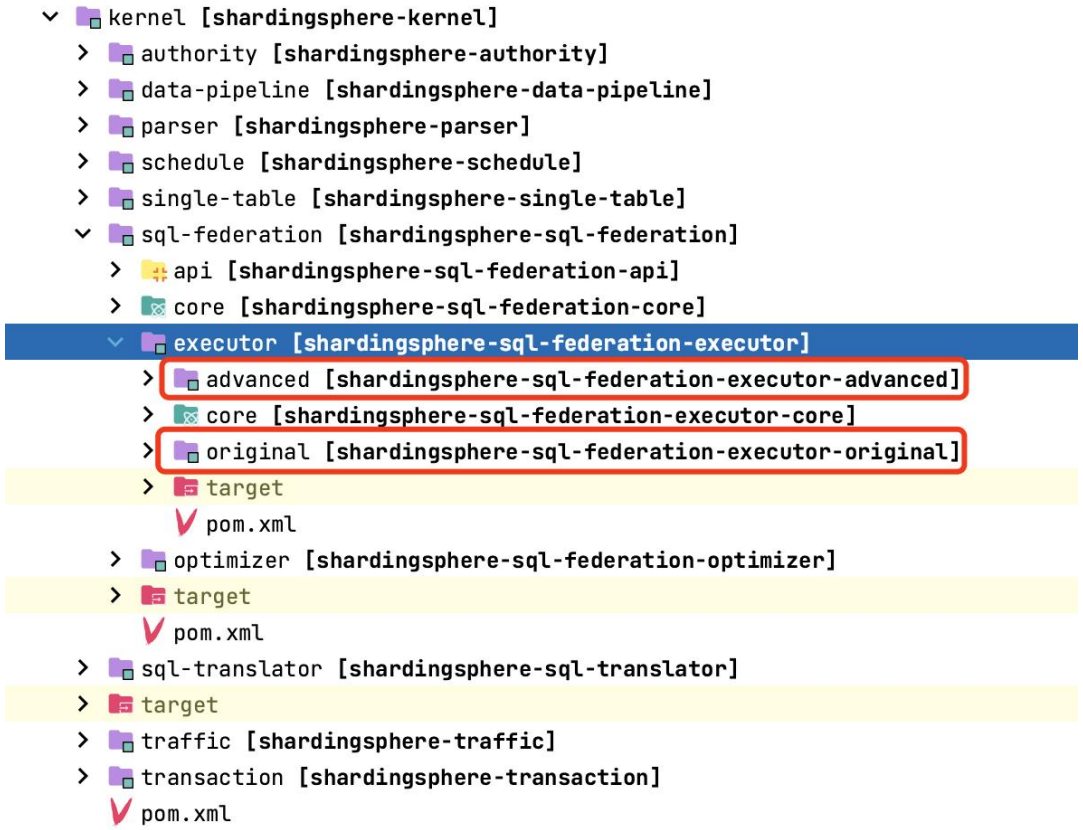
实现思路：

- 参考 Calcite JDBC，自行实现联邦查询引擎执行流程：
  - ShardingSphere SQL 解析；
  - SQLStatement 转换 SqlNode；
  - SqlNode 转换 RelNode 并校验；
  - 基于优化规则进行逻辑优化；
  - 基于统计信息，代价模型进行物理优化（未实现）；
  - 物理计划执行（未实现自定义算子）；
- 统一内核逻辑元数据和真实元数据，使用逻辑元数据注册到 Calcite，逻辑 SQL 通过决断器后交由 Calcite 执行；
- 基于优化之后的物理计划生成 SQL。

存在的问题：

- 存在 SQLStatement 和 SqlNode 的相互转换，影响效率和兼容性；
- 缺乏统计信息和代价模型，暂不支持物理优化；
- 物理计划的执行依赖 Calcite 内置的 EnumerableRel 算子，性能一般且无法灵活控制；
- 缺乏内存保护机制，需要避免加载数据过多导致 OOM；

联邦查询引擎第二版



联邦查询引擎改造：

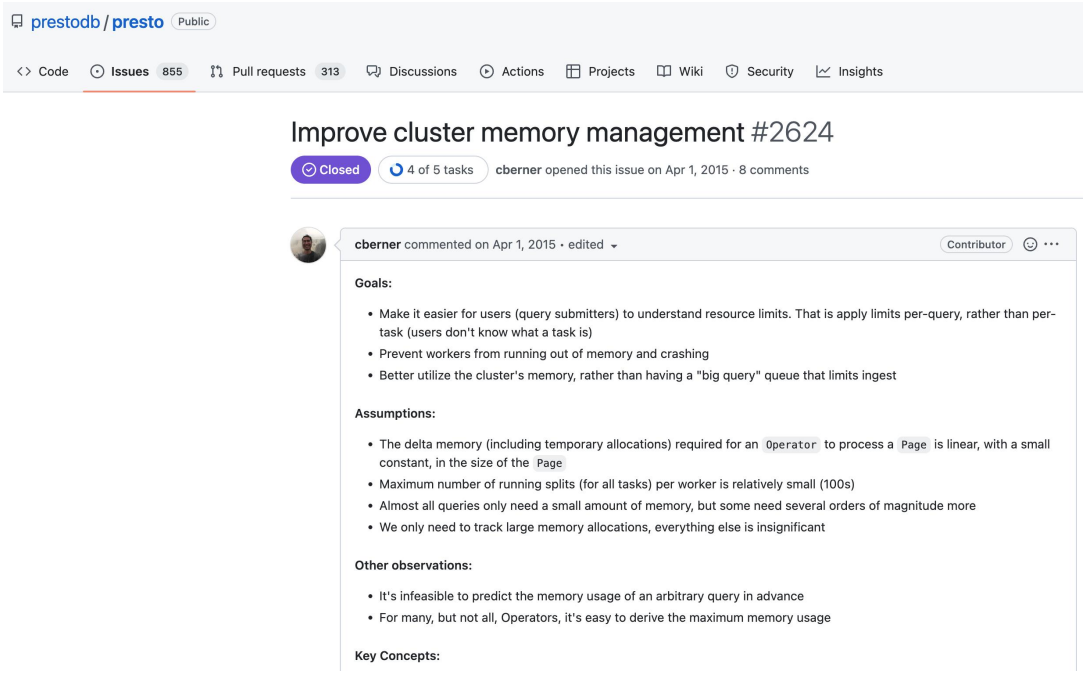
- SQL Federation 由 infra 模块抽取到 kernel 模块，作为内置的功能；
- ShardingSphere 元数据和联邦查询引擎完全解耦；
- 通过 SQLFederationExecutor SPI 接口，将 Calcite JDBC 实现和自定义实现进行隔离，用户通过 sql-federation-type: ADVANCED 配置使用；

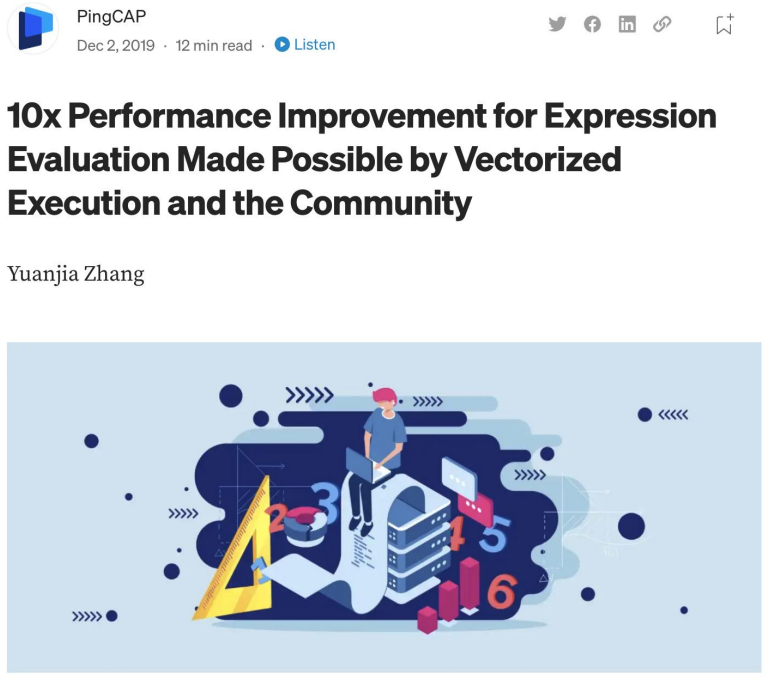
联邦查询引擎功能增强：

- 跨库关联查询，包括：INNER JOIN、 CROSS JOIN、 LEFT JOIN、 RIGHT JOIN、 FULL JOIN 和 NATURAL [XXX] JOIN；
- 子查询，包括： Projection 子查询， From 子查询和 Where 子查询；
- 支持任意组合的 MAX、MIN、SUM、AVG 和 COUNT 聚合函数；
- 支持 UNION/INTERSECT/EXCEPT/MINUS 查询；
- 支持逻辑视图查询；

联邦查询引擎第三版

- 统一 ShardingSphere 解析引擎和 Calcite 解析引擎，统一以 SqlNode 抽象语法树作为解析结果；
- 基于 ShardingSphere 系统库，收集 Calcite 优化所需的统计信息；
- 设计 ShardingSphere 分布式数据库代价模型，cost = row_count + cpu + io + 网络传输；
- 自定义部分物理算子，实现并发执行引擎，提供执行性能；
- 设计联邦查询引擎内存管理器，有效管控执行过程中的内存占用；
- 设计向量化执行引擎，进一步提升联邦查询引擎性能；





联邦查询引擎适配 openGauss

支持 openGauss 查询语法解析

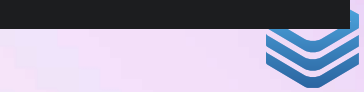
SQL参考 > SQL语言结构和语法 > SQL语法 > SELECT

- 查询数据

```
[ WITH [ RECURSIVE ] with_query [, ...] ]
SELECT [/*+ plan_hint */] [ ALL | DISTINCT [ ON
{ * | {expression [ [ AS ] output_name ]} [, ..
[ into_option ]
[ FROM from_item [, ...] ]
[ WHERE condition ]
[ [ START WITH condition ] CONNECT BY [NOCYCLE]
[ GROUP BY grouping_element [, ...] ]
[ HAVING condition [, ...] ]
[ WINDOW {window_name AS ( window_definition )}
[ { UNION | INTERSECT | EXCEPT | MINUS } [ ALL
[ ORDER BY {expression [ [ ASC | DESC | USING c
[ LIMIT { [offset,] count | ALL } ]
[ OFFSET start [ ROW | ROWS ] ]
[ FETCH { FIRST | NEXT } [ count ] { ROW | ROWS
[ into_option ]
[ {FOR { UPDATE | NO KEY UPDATE | SHARE | KEY S
[into_option];
```

DMLStatement.g4

```
101
102 select
103     : selectNoParens | selectWithParens
104     ;
105
106 selectWithParens
107     : LP_ selectNoParens RP_ | LP_ selectWithParens RP_
108     ;
109
110 selectNoParens   Duan, 2021/10/29, 18:52 • add independent openGauss parsing engine implementation (#13351)
111     : selectClauseN
112     | selectClauseN sortClause
113     | selectClauseN sortClause? forLockingClause selectLimit?
114     | selectClauseN sortClause? selectLimit forLockingClause?
115     | withClause selectClauseN
116     | withClause selectClauseN sortClause
117     | withClause selectClauseN sortClause? forLockingClause selectLimit?
118     | withClause selectClauseN sortClause? selectLimit forLockingClause?
119     ;
120
121 selectClauseN
122     : simpleSelect
123     | selectWithParens
124     | selectClauseN INTERSECT allOrDistinct? selectClauseN
125     | selectClauseN (UNION | EXCEPT | MINUS) allOrDistinct? selectClauseN
126     ;
```

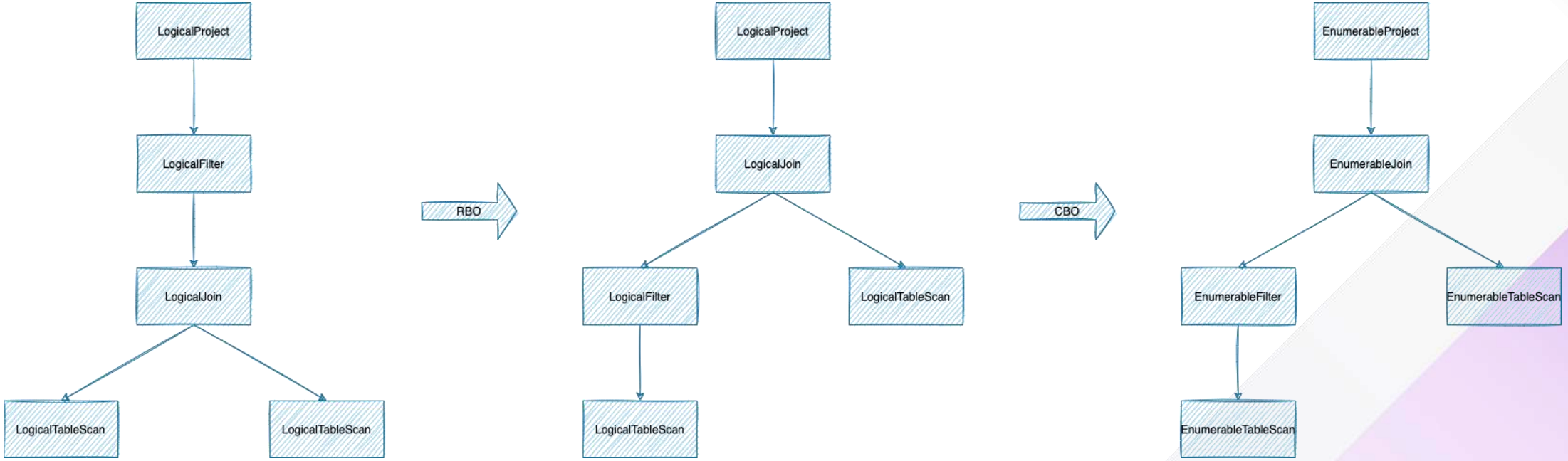


支持 openGauss 复杂查询执行

联邦查询引擎支持复杂查询语句需要在计算层实现全部查询相关的算子，以下面的 SQL 为例：

```
SELECT * FROM t_order o INNER JOIN t_order_item i ON o.order_id = i.order_id WHERE o.order_id = 3;
```

t_order 和 t_order_item 存储在不同的数据库实例上，SQL Federation 会基于 RBO 进行谓词下推、列裁剪等优化，CBO 优化目前仍在开发中，目前只是基于 Calcite VolcanoPlanner 优化器实现逻辑关系代数到物理关系代数的转换。



支持 openGauss 复杂查询执行

由于算子的实现工作量较大，目前联邦查询引擎支持了如下查询语句，仍然有一些复杂的 SELECT 语句不支持。

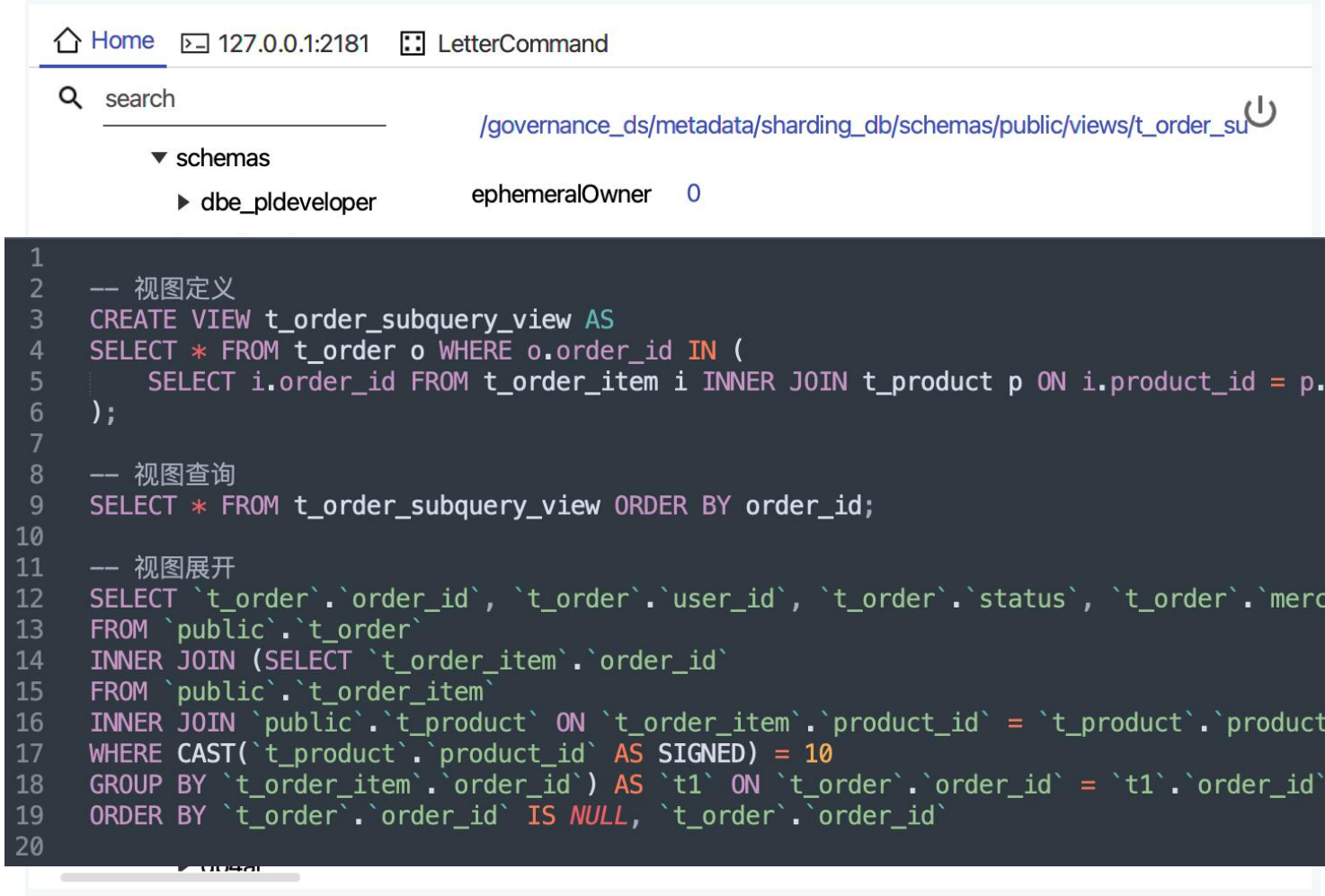
支持以下 SELECT 语句查询：

- 支持 SELECT GROUP BY HAVING 分组聚合查询及 MAX、MIN、SUM、AVG 和 COUNT 聚合函数；
- 支持 UNION | INTERSECT | EXCEPT | MINUS 查询；
- 支持 SELECT ORDER BY LIMIT 语句；
- 支持跨库 JOIN 查询，包括 INNER JOIN、CROSS JOIN、LEFT JOIN、RIGHT JOIN 和 FULL JOIN；
- 支持复杂子查询，包括 Projection、From 及 Where 中包含的子查询；
- 支持以上 SELECT 语句定义的视图查询；

不支持以下 SELECT 语句：

- 不支持 WITH 语句，例如：WITH temp_t(name,isdba) AS (SELECT usename,usesuper FROM pg_user) SELECT * FROM temp_t;
- 不支持 plan_hint 子句；
- 不支持 START WITH .. CONNECT BY 子句；
- 不支持 WINDOW 子句；
- 不支持 FOR UPDATE，FOR NO KEY UPDATE，FOR SHARE 或 FOR KEY SHARE 加锁语句；

支持 openGauss 跨库视图执行




```
1
2   -- 视图定义
3   CREATE VIEW t_order_subquery_view AS
4   SELECT * FROM t_order o WHERE o.order_id IN (
5       SELECT i.order_id FROM t_order_item i INNER JOIN t_product p ON i.product_id = p.product_id WHERE p.product_id = 10
6   );
7
8   -- 视图查询
9   SELECT * FROM t_order_subquery_view ORDER BY order_id;
10
11  -- 视图展开
12  SELECT `t_order`.`order_id`, `t_order`.`user_id`, `t_order`.`status`, `t_order`.`merchant_id`, `t_order`.`remark`, `t_order`.`creation_date`
13  FROM `public`.`t_order`
14  INNER JOIN (SELECT `t_order_item`.`order_id`
15  FROM `public`.`t_order_item`
16  INNER JOIN `public`.`t_product` ON `t_order_item`.`product_id` = `t_product`.`product_id`
17  WHERE CAST(`t_product`.`product_id` AS SIGNED) = 10
18  GROUP BY `t_order_item`.`order_id`) AS `t1` ON `t_order`.`order_id` = `t1`.`order_id`
19  ORDER BY `t_order`.`order_id` IS NULL, `t_order`.`order_id`
20
```

支持 openGauss CREATE VIEW、ALTER VIEW、DROP VIEW 等跨库视图语句。

ShardingSphere 自行实现了逻辑视图，将视图定义存储在治理中心，通过联邦查

联邦查询引擎实战

开启联邦查询引擎

sql-federation-type: ADVANCED

proxy-frontend-database-protocol-type: openGauss

DMLStatement.g4 | server.yaml | props.en.md

```yaml
props:
  system-log-level: INFO
  max-connections-size-per-query: 1
  kernel-executor-size: 16  # Infinite by default.
  proxy-frontend-flush-threshold: 128  # The default value is 128.
  proxy-hint-enabled: false
  # sql-show is the same as props in logger ShardingSphere-SQL, and its priority is lower than logging rule
  sql-show: true
  check-table-metadata-enabled: false
    # Proxy backend query fetch size. A larger value may increase the memory usage of ShardingSphere Proxy.
    # The default value is -1, which means set the minimum value for different JDBC drivers.
  proxy-backend-query-fetch-size: -1
  proxy-frontend-executor-size: 0 # Proxy frontend executor size. The default value is 0, which means let Netty decide.
  proxy-frontend-max-connections: 0 # Less than or equal to 0 means no limitation.
    # Available sql federation type: NONE (default), ORIGINAL, ADVANCED
  sql-federation-type: ADVANCED
  proxy-default-port: 3307 # Proxy default port.
  proxy-netty-backlog: 1024 # Proxy netty backlog.
  cdc-server-port: 33071 # CDC server port
  proxy-frontend-database-protocol-type: openGauss   You, Moments ago • Uncommitted changes
```

联邦查询引擎 & openGauss 实战 Case

技术交流

项目地址： https://shardingsphere.apache.org

GitHub 地址： https://github.com/apache/shardingsphere

开发者邮件列表： dev-subscribe@shardingsphere.apache.org

中文社区： https://community.sphere-ex.com

加入交流群

技术干货

Thank you!

openGauss

https://opengauss.org