翼支付大数据BI分析平台建设实践

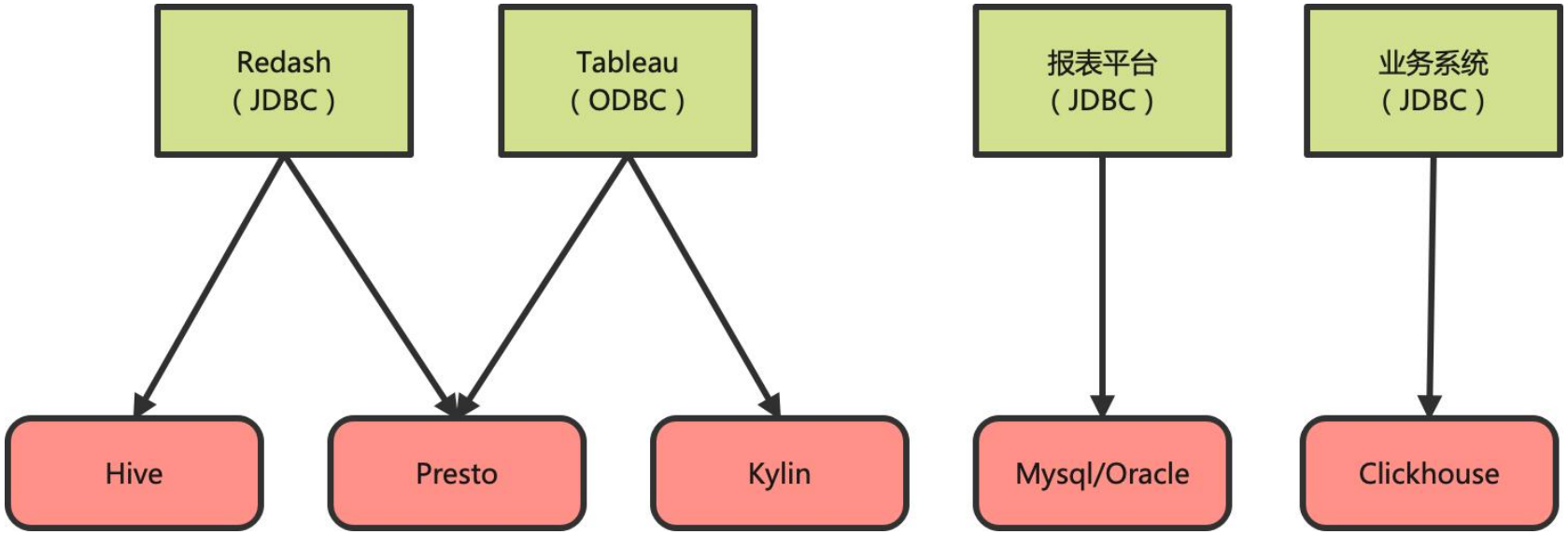
演讲人：吴晓兵、唐晔

一	翼支付在金融大数据分析的应用
二	翼支付大数据 BI 分析平台架构
三	OLAP引擎技术实践
四	未来规划

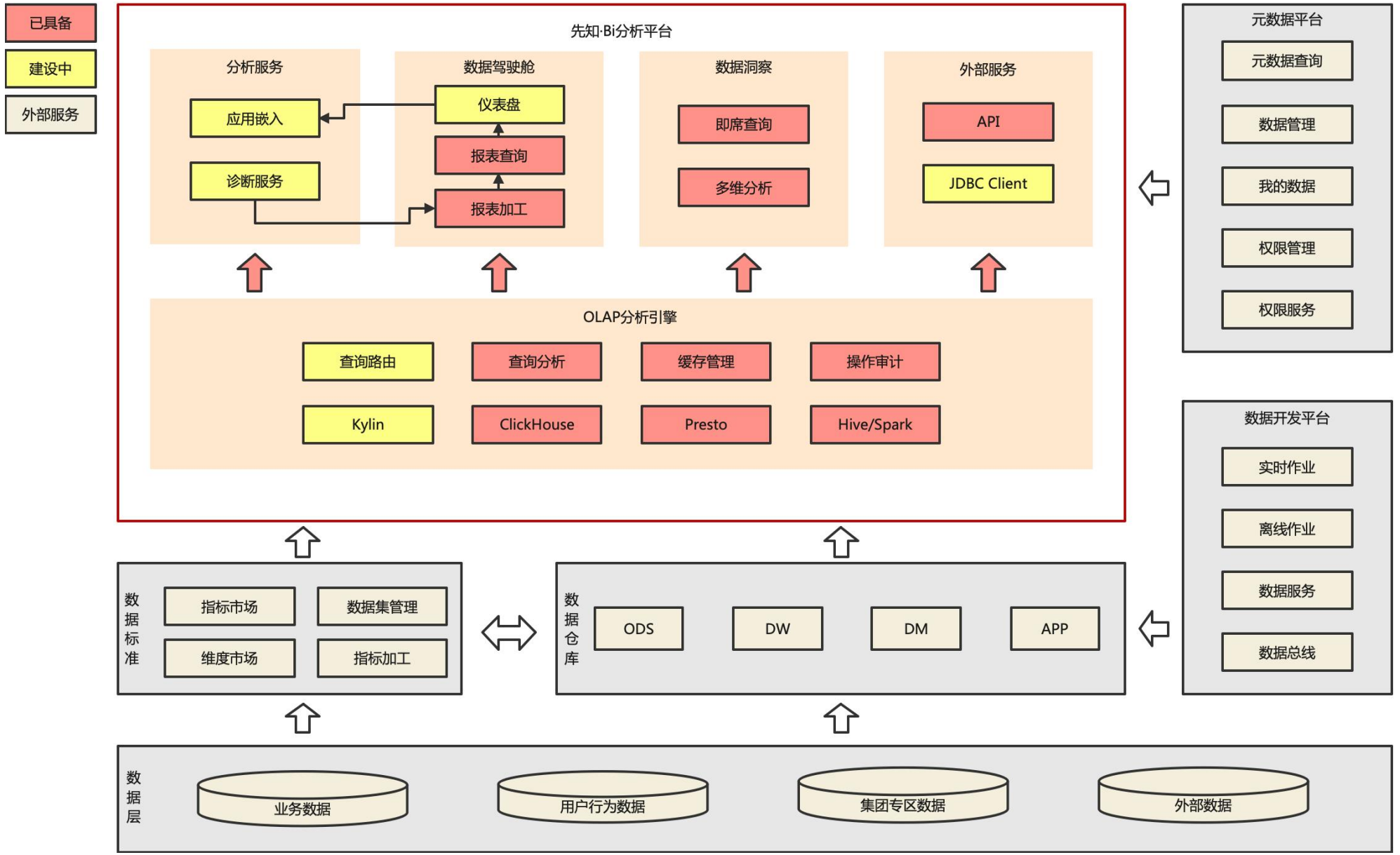
业务场景

- 业务场景
  - 数据探查
  - 数据可视化
  - 实时数据快速查询
  - 离线数据快速查询

- 存在问题：
  - 烟囱式架构
  - 查询性能及稳定性差
  - 自助式数据获取门槛过高
  - 数据权限管控混乱
  - 数据质量低下



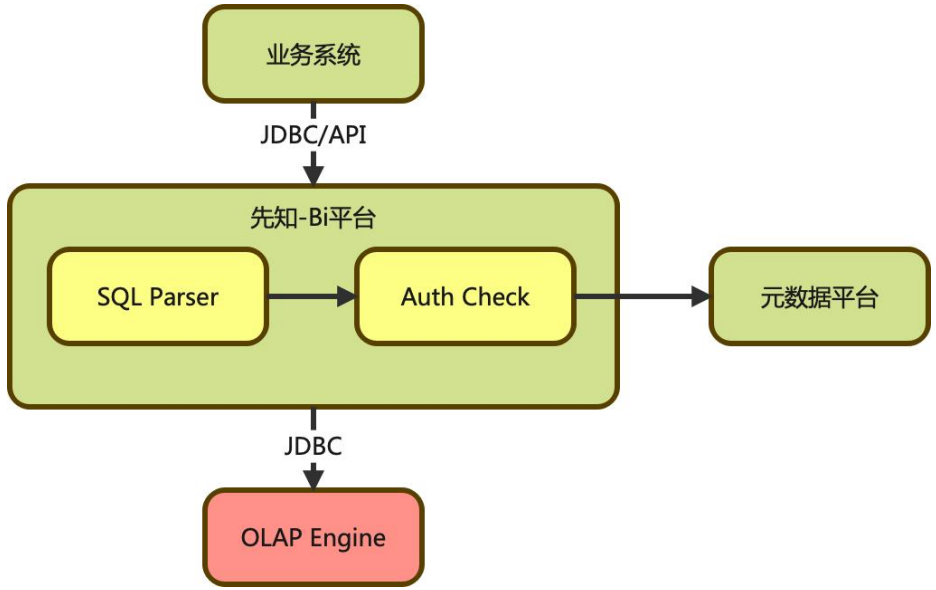
平台架构

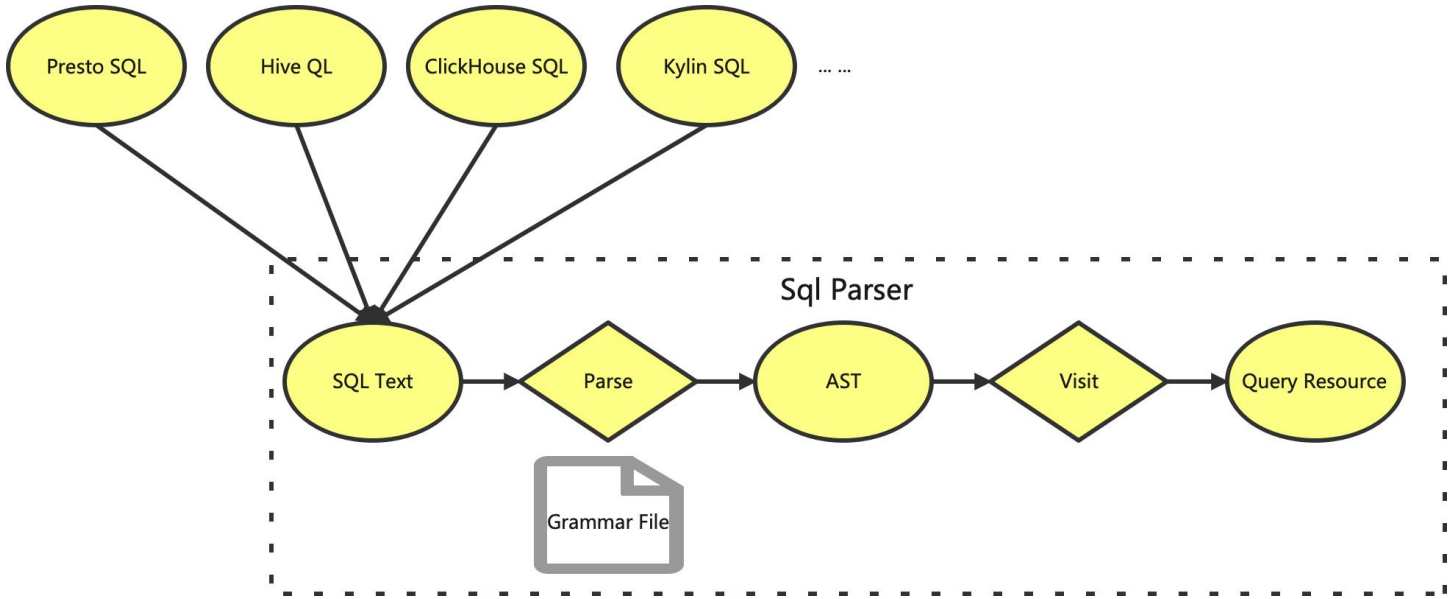

已具备
建设中
外部服务
先知·Bi分析平台
分析服务
应用嵌入
诊断服务
数据驾驶舱
仪表盘
报表查询
报表加工
数据洞察
即席查询
多维分析
外部服务
API
JDBC Client
OLAP分析引擎
查询路由
查询分析
缓存管理
操作审计
Kylin
ClickHouse
Presto
Hive/Spark
元数据平台
元数据查询
数据管理
我的数据
权限管理
权限服务
数据开发平台
实时作业
离线作业
数据服务
数据总线
数据标准
指标市场
数据集管理
维度市场
指标加工
数据仓库
ODS
DW
DM
APP
数据层
业务数据
用户行为数据
集团专区数据
外部数据

一	翼支付在金融大数据分析的应用
二	翼支付大数据 BI 分析平台架构
三	OLAP引擎技术实践
四	未来规划

多租户权限管控

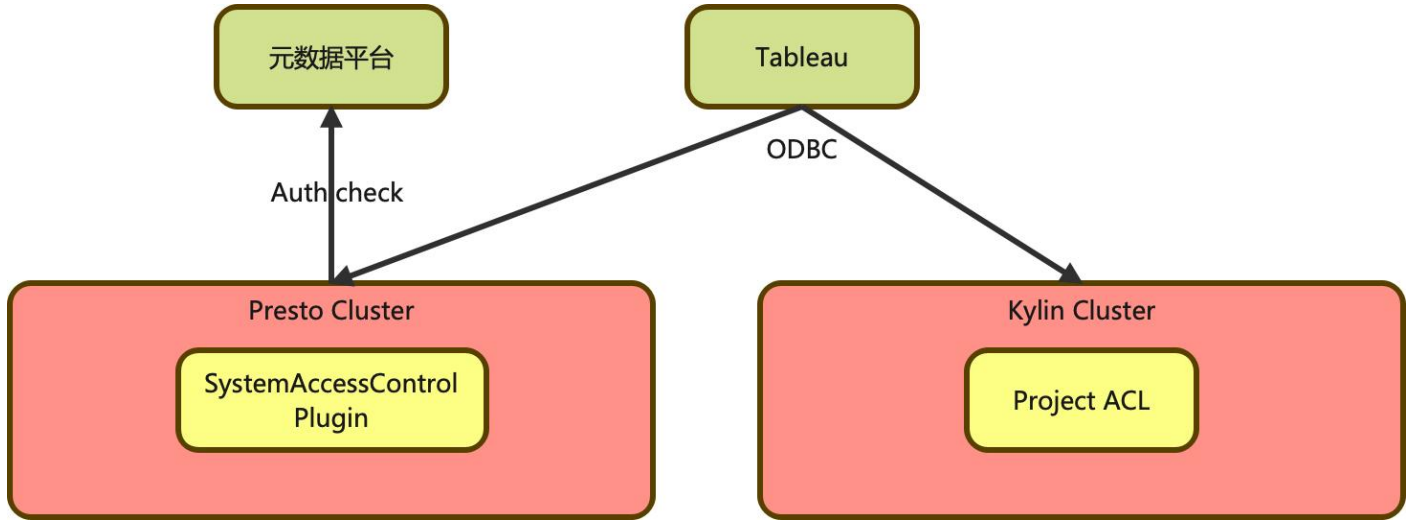
- 平台侧权限管控：
  - 参考presto-parser模块的代码用于SQL解析
  - 修改Presto的SqlBase.g4文件，添加对Hive/Spark语法的支持
  - 使用Antlr4的Visitor模式对生成的AST进行遍历，解析出SQL中的资源信息，随后与元数据平台交互来进行权限校验





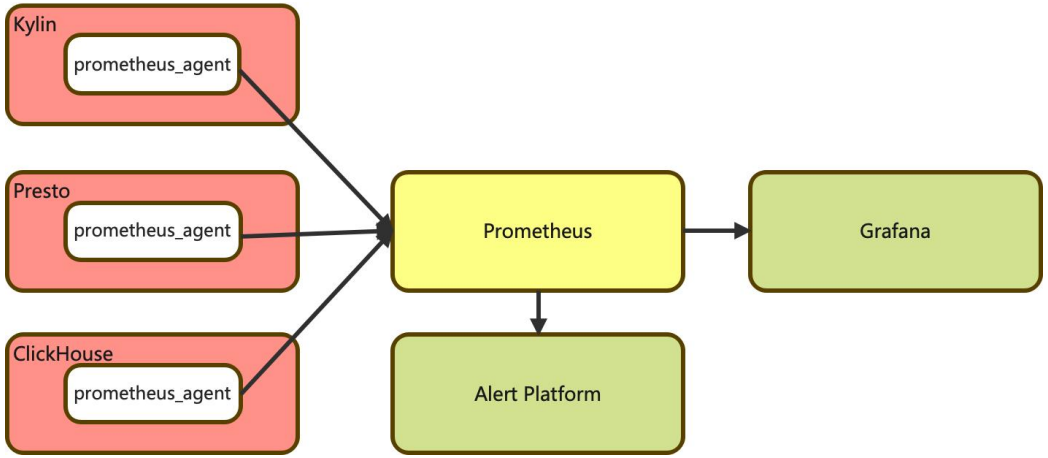
多租户权限管控

- Tableau权限管控：
  - 基于Presto SPI开发SystemAccessControl Plugin对接元数据平台进行权限管控
  - Kylin采用的Project ACL进行权限管控

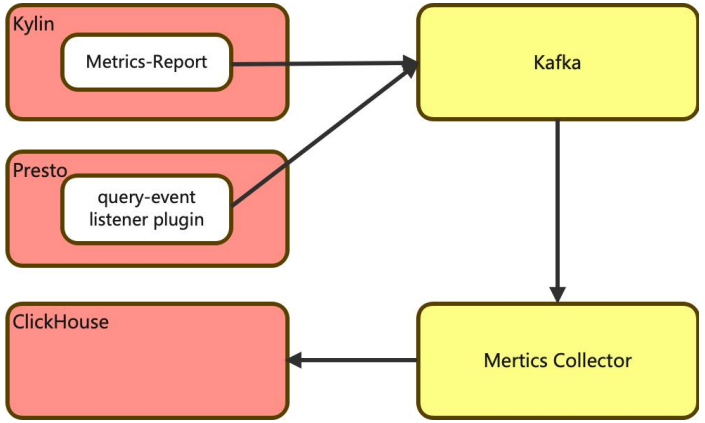


OLAP Metrics收集

- OLAP集群监控与告警：
  - 每种引擎都集成prometheus_agent向Prometheus汇报集群核心Metrics
  - Grafana绘制Cluster Dashboard
  - Alert Platform进行异常指标告警



- Query层级监控与分析：
  - 基于Presto SPI开发query-event-listener插件汇报单SQL Metrics
  - Kylin通过Metrics-Report模块向kafka发送cube查询的指标
  - CK本身会将查询信息落到系统表中
  - 最终所有的查询Metrics落CK，通过QueryId即可对单SQL进行性能瓶颈分析



CK join分布式表不走local join的问题

- 描述：在启用了local join(set distributed_product_mode=local)之后，直接join分布式表不走local join。

更具体描述可参考文章：https://mp.weixin.qq.com/s/FL4fK2W4DKVVlrPzmyOyfA

```
select xxx
from t inner join distrib_t
on xxx
```

```
select xxx
from t inner join
(
select xxx
from distrib_t
)on xxx
```

CK join分布式表不走local join的问题

- 相关代码：src/Interpreters/InJoinSubqueriesPreprocessor.cpp

```
191         ASTTableJoin * table_join = node.table_join->as<ASTTableJoin>();
192         if (table_join->locality != ASTTableJoin::Locality::Global)
193         {
194             if (auto & subquery = node.table_expression->as<ASTTableExpression>()->subquery)
195             {
196                 std::vector<ASTPtr> renamed;
197                 NonGlobalTableVisitor::Data table_data(data.getContext(), data.checker, renamed, nullptr, table_join);
198                 NonGlobalTableVisitor(table_data).visit(subquery);
199                 if (!renamed.empty()) //-V547
200                     data.renamed_tables.emplace_back(subquery, std::move(renamed));
201             }
202         }
```

V21.11及以前的代码

```
196             if (auto & subquery = table->subquery)
197             {
198                 std::vector<ASTPtr> renamed;
199                 NonGlobalTableVisitor::Data table_data(data.getContext(), data.checker, renamed, nullptr, table_join);
200                 NonGlobalTableVisitor(table_data).visit(subquery);
201                 if (!renamed.empty()) //-V547
202                     data.renamed_tables.emplace_back(subquery, std::move(renamed));
203             }
204             else if (table->database_and_table_name)
205             {
206                 auto tb = node.table_expression;
207                 std::vector<ASTPtr> renamed;
208                 NonGlobalTableVisitor::Data table_data{data.getContext(), data.checker, renamed, nullptr, table_join};
209                 NonGlobalTableVisitor(table_data).visit(tb);
210                 if (!renamed.empty()) //-V547
211                     data.renamed_tables.emplace_back(tb, std::move(renamed));
212             }
```

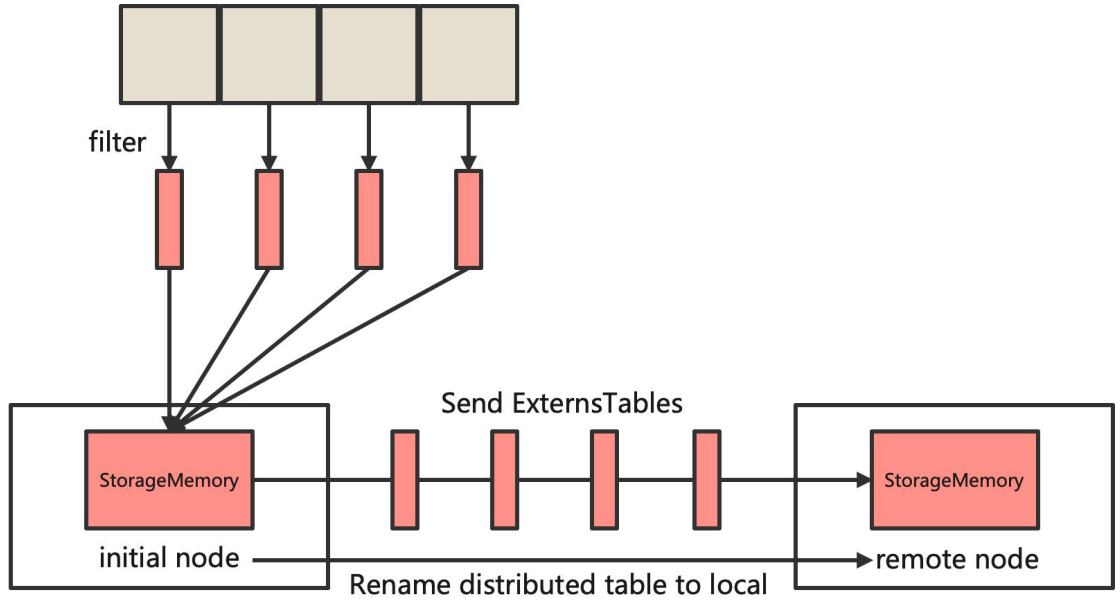
V21.12及以后的代码

CK Global IN/Not IN性能问题

- 描述：在使用global in/not in时，出现严重的性能问题，甚至查询超时。

具体分析可参考文章:https://mp.weixin.qq.com/s/Dv7N_td9t5RfhuymkQlPug

```
select xxx
from t where xxx global in (
select xxx
from t2 where xxx
)
```



Clickhouse mergetree引擎并行读取机制与本场景中t2表的数据分布导致读入了大量碎片block

StorageMemory的数据以block形式存储，粒度与写入时一致，不会进行合并

Send ExternsTables时按block逐个发送数据到remote node

CK Global IN/Not IN性能问题

- 相关代码：src/Interpreters/GlobalSubqueriesVisitor.h

```
150             auto external_table = external_storage_holder->getTable();
151             auto table_out = external_table->write({}, external_table->getInMemoryMetadataPtr(), getContext());
152             auto io = interpreter->execute();
153             PullingAsyncPipelineExecutor executor(io.pipeline);
154
155             table_out->writePrefix();
156             Block block;
157             while (executor.pull(block))
158             {
159                 if (block)
160                     table_out->write(block);
161                 block.clear();
162             }
163
164             table_out->writeSuffix();
```

V21.8及以前代码

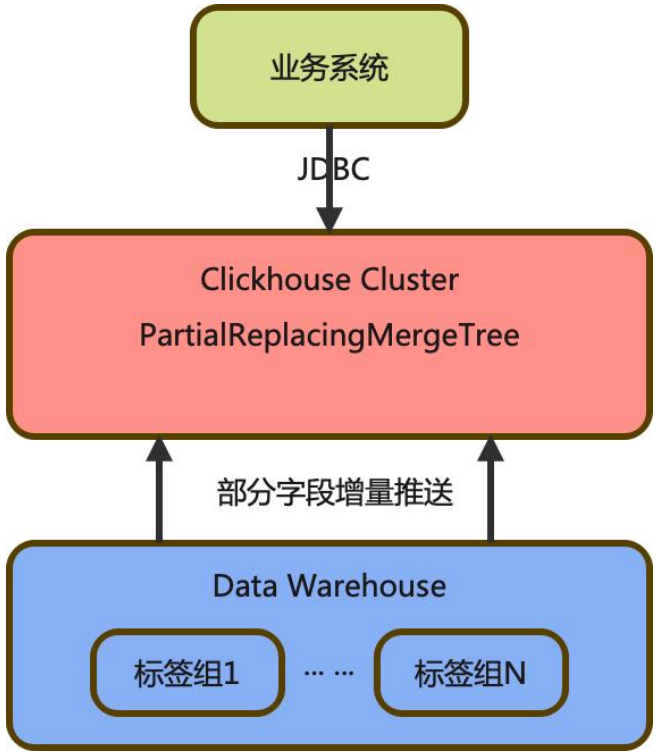
```
151             auto external_table = external_storage_holder->getTable();
152             auto table_out = external_table->write({}, external_table->getInMemoryMetadataPtr(), getContext());
153             auto io = interpreter->execute();
154             //squashing blocks
155             BlockInputStreamPtr data = std::make_shared<SquashingBlockInputStream>(
156                     io.getInputStream(), getContext()->getSettingsRef().min_insert_block_size_rows,
157                     getContext()->getSettingsRef().min_insert_block_size_bytes);
158             copyData(*data, *table_out);
```

使用SquashingBlockInputStream紧凑block

CK添加PartialReplacingMergeTree引擎

- 需求：标签数据以部分字段增量的方式直接推送clickhouse，去除对hbase的依赖。

- 意义：
  - 去除对hbase的依赖，缩短推数链路，合并出最新快照交由clickhouse完成，性能和稳定性会有所提高。
  - 每日定时人群包推送通过select xxx from t final的方式可提早服务就绪时间。



CK添加PartialReplacingMergeTree引擎

- 实现：基于主键(primary key)，按照指定顺序(order by)，根据用户指定的字段index(从1开始)进行部分字段更新，并支持删除功能。
- 代码：https://github.com/kindred77/ClickHouse/tree/21.3-partial-replacing-mergetree-engine

```
CREATE TABLE partial_rep_test
(
    `id` UInt64,
    `name` String,
    `age` UInt8,
    `col_idxes_arr` Array(UInt16),
    `version` DateTime
)
ENGINE = PartialReplacingMergeTree(col_idxes_arr)
PARTITION BY id
PRIMARY KEY id
ORDER BY (id, version);
```

id为主键，id相同的记录会被合并。

col_idxes_arr指定需要更新的字段序号列表，从1开始，单独一个元素0表示删除记录，没有任何元素表示覆盖所有字段。

数据按照id, version的顺序进行合并。

CK添加PartialReplacingMergeTree引擎

part1

id	name	age	col_idxes_arr	version
1	111	20	[]	2021-07-31 00:00:00
1	222	20	[2,5]	2021-07-31 00:00:01
2	test	25	[2,3,5]	2021-07-28 00:00:01
3	test3	21	[]	2021-07-28 00:00:01

part2

id	name	age	col_idxes_arr	version
1	333	30	[2,3,5]	2021-07-31 00:00:02
2	test2	30	[]	2021-07-28 00:00:00
2	test3	35	[3,5]	2021-07-28 00:00:02
3	x	x	[0]	2021-07-28 00:00:02

MergeSorting

id	name	age	col_idxes_arr	version
1	111	20	[]	2021-07-31 00:00:00
1	222	20	[2,5]	2021-07-31 00:00:01
1	333	30	[2,3,5]	2021-07-31 00:00:02
2	test2	30	[]	2021-07-28 00:00:00
2	test	25	[2,3,5]	2021-07-28 00:00:01
2	test3	35	[3,5]	2021-07-28 00:00:02
3	test3	21	[]	2021-07-28 00:00:01
3	x	x	[0]	2021-07-28 00:00:02

Apply merge Algorithm

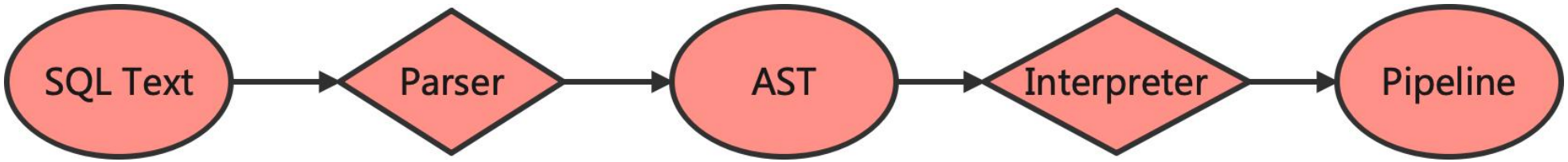
id	name	age	col_idxes_arr	version
1	111	20	[]	2021-07-31 00:00:00
1	222	20	[2,5]	2021-07-31 00:00:01
1	333	30	[2,3,5]	2021-07-31 00:00:02
2	test2	30	[]	2021-07-28 00:00:00
2	test	25	[2,3,5]	2021-07-28 00:00:01
2	test3	35	[3,5]	2021-07-28 00:00:02
3	test3	21	[]	2021-07-28 00:00:01
3	x	x	[0]	2021-07-28 00:00:02

InsertRow

id	name	age	col_idxes_arr	version
1	333	30	[]	2021-07-31 00:00:02
2	test	35	[]	2021-07-28 00:00:02

CK执行计划以及优化器改造

- clickhouse最大的短板在执行计划及优化器模块

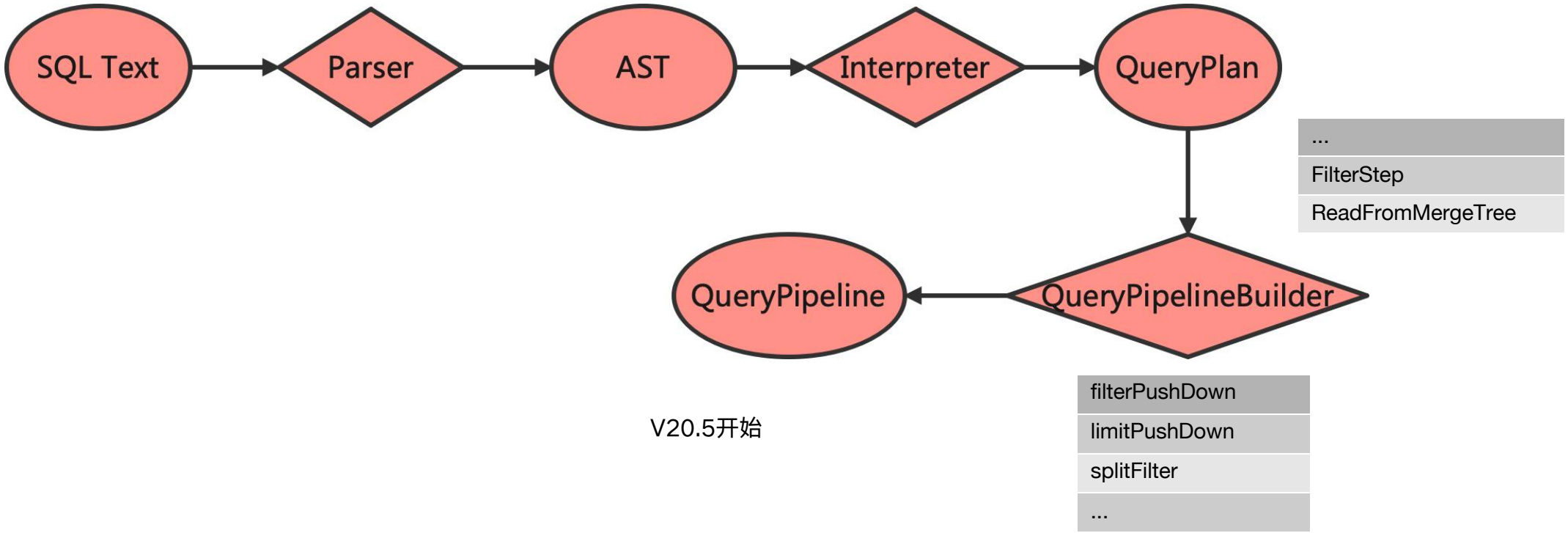


V20.4及以前

...
FilterBlockInputStream
MergeTreeSelectBlockInputStream

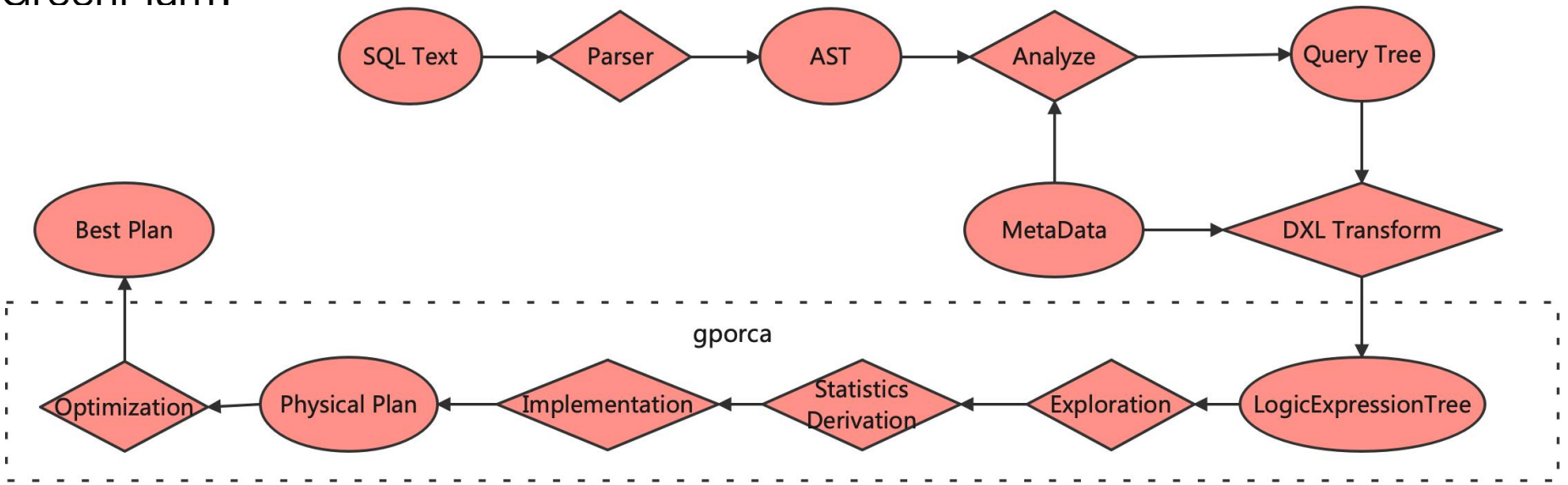
CK执行计划以及优化器改造

- 从V20.5逐步到最新的版本，初步执行计划及优化器模块雏形
- 不过QueryPlan的优化是基于简单的几条优化规则进行，暂时还是不具备CBO特征

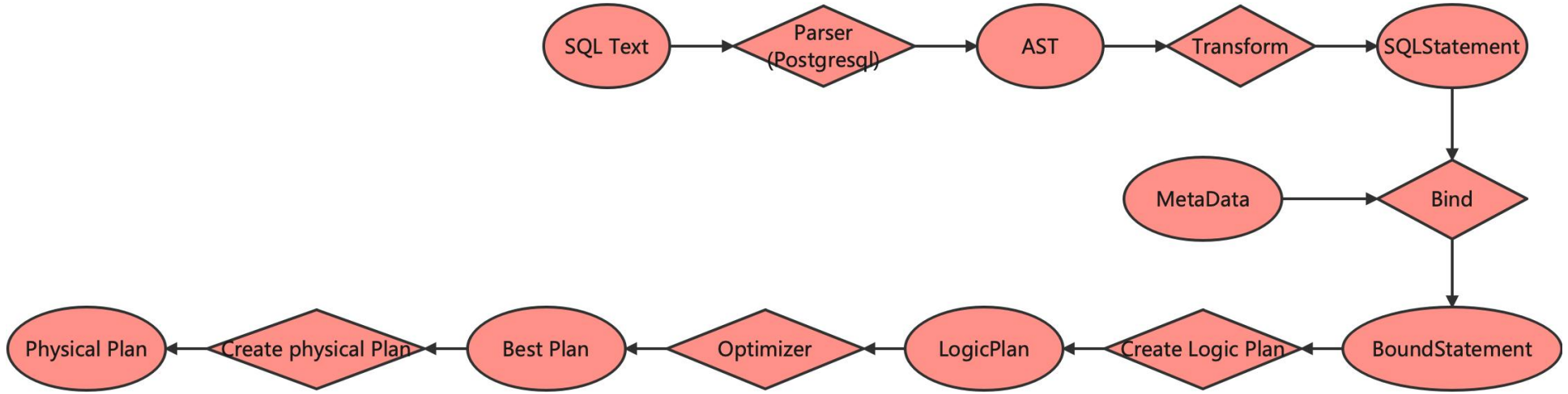


CK执行计划以及优化器改造

- GreenPlum：

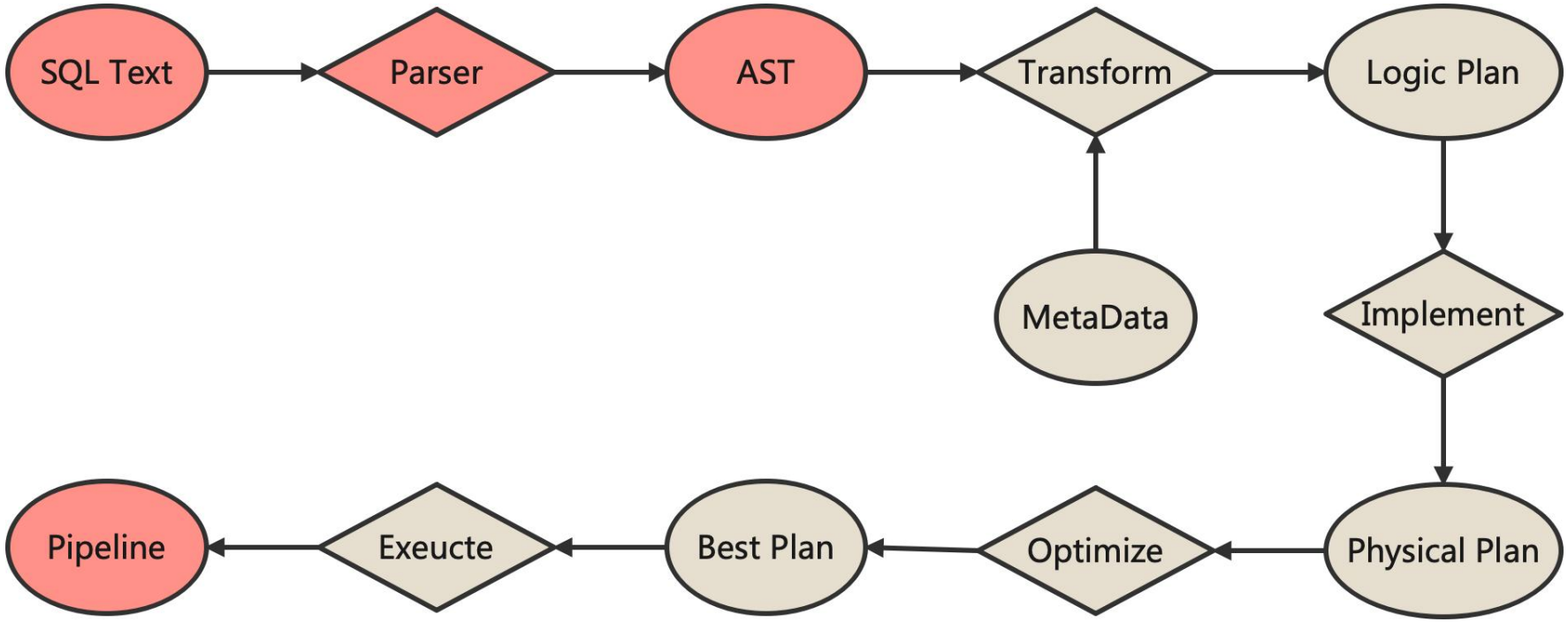


- DuckDB：



CK执行计划以及优化器改造

- 意义：
  - clickhouse具备性能强悍的算子，配合完善的优化器将更能发挥性能优势，能解决更多问题。
  - 进一步提升我们对clickhouse的掌控度。



一	翼支付在金融大数据分析的应用
二	翼支付大数据 BI 分析平台架构
三	OLAP引擎技术实践
四	未来规划

未来规划

- 平台和OLAP引擎侧全面云原生化
- 对用户屏蔽不同SQL的语法差异，实现不同SQL语法的互转
- 基于规则的智能路由引擎

中国电信 CHINA TELECOM

翼支付

Thank you !