

KYLIN-5431 Enhance the aggregation push down capability, and support using aggregate indexes to associate snapshots to answer queries

问题描述

Problem description

无子查询join模式下，原本只能支持tableIndex索引+snapshot联合回答，需要优化成可以支持聚合索引+snapshot联合回答

In the subquery join mode, it can only support tableIndex index + snapshot joint answer, but it needs to be optimized to support aggregated index + snapshot joint answer

举例说明：

For example:

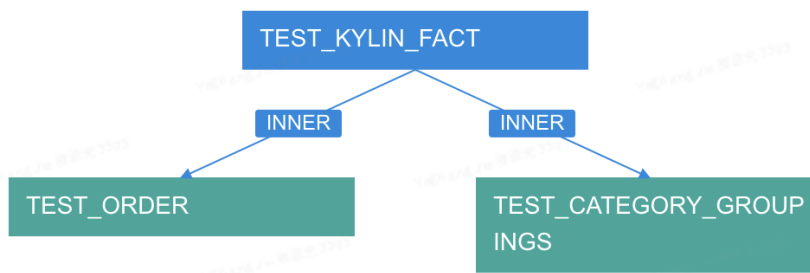
查询sql：

Query SQL :

```
1 select cast(3.1234 as decimal(8, 3)),SELLER_ID,sum(price)
2 from TEST_KYLIN_FACT
3 inner join TEST_ORDER on TEST_ORDER.ORDER_ID = TEST_KYLIN_FACT.ORDER_ID
4 inner join TEST_CATEGORY_GROUPINGS as tcg on TEST_KYLIN_FACT.LEAF_CATEG_ID = tcg
5 inner join KYLIN_CATEGORY_GROUPINGS as kcg on TEST_KYLIN_FACT.LEAF_CATEG_ID = kcg
6 group by SELLER_ID
7 LIMIT 500
```

modelA:

modelA:



snapshot表:

snapshot表:

KYLIN_CATEGORY_GROUPINGS

KYLIN_CATEGORY_GROUPINGS

查询的时候:

When inquiring:

```
1 select cast(3.1234 as decimal(8, 3)),SELLER_ID,sum(price) from TEST_KYLIN_FACT
2 inner join TEST_ORDER on TEST_ORDER.ORDER_ID = TEST_KYLIN_FACT.ORDER_ID
3 inner join TEST_CATEGORY_GROUPINGS on TEST_KYLIN_FACT.LEAF_CATEG_ID = TEST_CATEGORY_GROUPINGS.LEAF_CATEG_ID
4 left join KYLIN_CATEGORY_GROUPINGS on TEST_KYLIN_FACT.LEAF_CATEG_ID = KYLIN_CATEGORY_GROUPINGS.LEAF_CATEG_ID
5 group by SELLER_ID
6 LIMIT 500
```

查询 ID: c5542aa6-9611-6a52-d74c-31ca8d896204

查询耗时: 0s

查询对象: model_03

索引 ID: 20000000001

快照: DEFAULT.KYLIN_CATEGORY_GROUPINGS

查询扫描记录数: 144

查询扫描字节数: 58733

查询结果记录数: 500

是否击中缓存: true

缓存类型: EHCACHE

背景

background

Kylin已经支持了agg pushdown的行为，该行为是借鉴与calcite的AggregateJoinTransposeRule，因此agg pushdown这部分逻辑我们可以借用，但是aggpushdown会有scope，因为本issue也会存在下压的scope。

Kylin supported the behavior of agg pushdown, which is borrowed from the AggregateJoinTransposeRule of calcite, so we can borrow the logic of agg pushdown, but

aggpushdown will have scope, because this issue will also have the scope of pressure.

QueryContext Cutter包含2个动作:

QueryContext Cutter contains 2 actions:

1 cut context

2 select Realization

实现方案

implementation plan

因为aggpushdown本身不会改变查询结果，只会影响查询会不会命中最优结果。因此本次采用策略是扩大aggpushdown的scope，从而达到当前场景下的sql可以用聚合索引+snapshot回答。

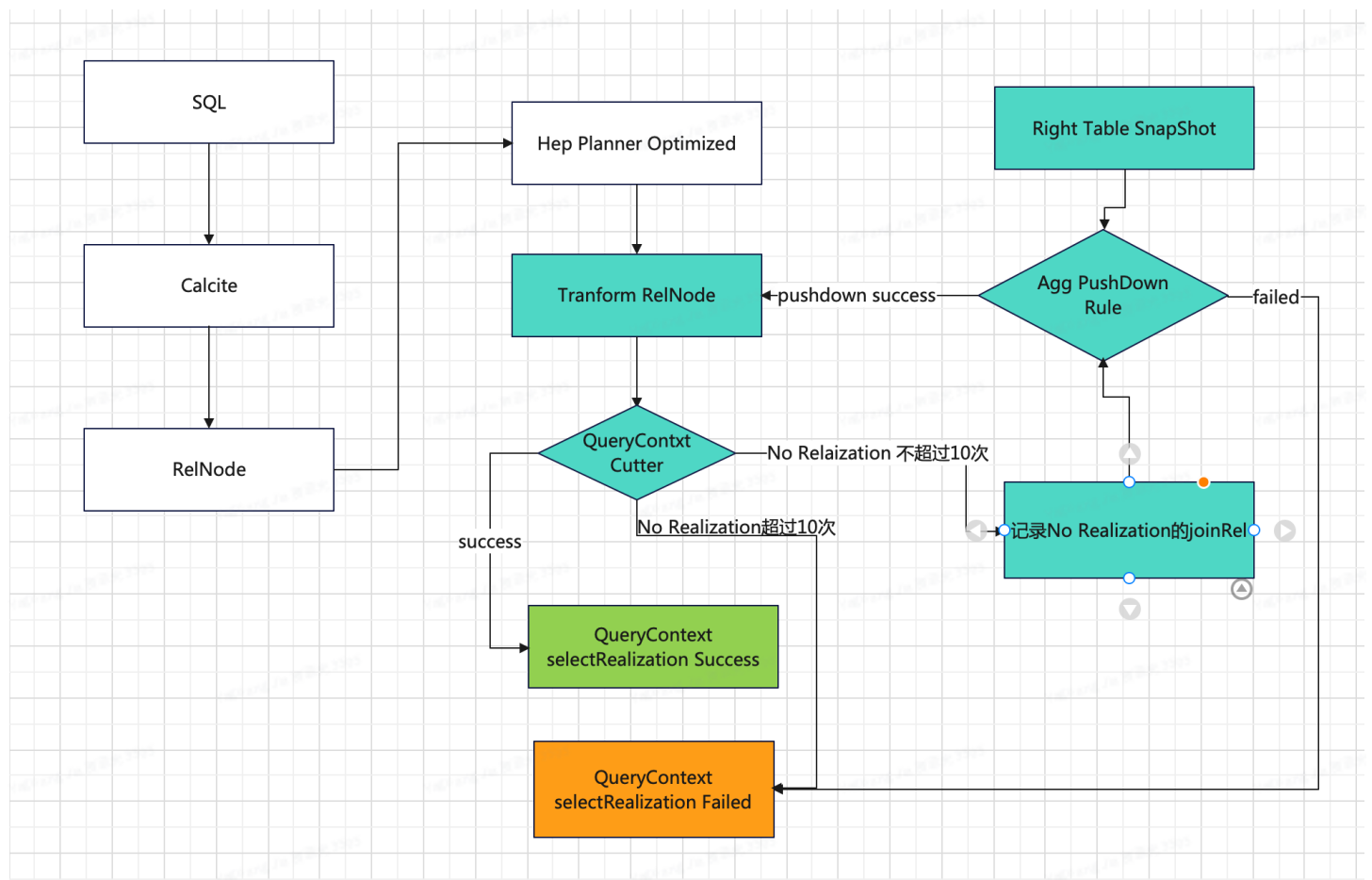
Because aggpushdown itself does not change the query result, it only affects whether the query will hit the optimal result. Therefore, the strategy adopted this time is to expand the scope of aggpushdown, so that SQL in the current scenario can be answered with aggregated index + snapshot.

增加配置: `kylin.query.enhanced-agg-pushdown-enabled=false` (默认不开启, 项目级别配置)

Add configuration: `kylin.query.enhanced -agg-pushdown-enabled = false` (not enabled by default, project-level configuration)

实现逻辑:

Implementation logic:



核心：当QueryContext Cutter不能匹配realization回答的时候记录当前joinRel，同时right能够命中snapshot，则进行agg pushdown。

Core: When QueryContext Cutter cannot match the realization answer, record the current joinRel, and right can hit snapshot, then agg pushdown is performed.

具体操作：

Specific operation:

- 1 在transform的QueryConext Cutter的时候，根据No Realization记录没有命中的joinRel。
- 1 When transforming QueryConext Cutter, joinRel that did not hit was recorded according to No Realization.
- 2 在transform整个不能在QueryConext Cutter回答的时候，对transform进行agg pushdown。
- 2 Agg pushdown on transform when transform cannot be answered in QueryConext Cutter.
- 3 agg pushdown沿用原来的代码，只是match的位置采用双重控制，即增加新的逻辑，如果满足则进行agg pushdown。
- 3 agg pushdown follows the original code, but the position of the match adopts dual control, that is, new logic is added, and if it is satisfied, agg pushdown is performed.

匹配逻辑：

Matching logic:

joinRel的left节点是不能找到realization。

joinRel's left node is unable to find realization.

joinRel的right节点是一个snapshot表。

The right node of joinRel is a snapshot table.

4 agg pushdown rule如果成功push down且不超过最大10次，则继续执行QueryConext Cutter。否则，跳出当前逻辑，报错No Realization。

4 agg pushdown rule If the push down is successful and does not exceed the maximum 10 times, continue to execute QueryConext Cutter. Otherwise, jump out of the current logic and report No Realization.

5 支持left join设计：aggPushDown采用的开源中的AggregateJoinTransposeRule方法，该方法仅支持inner join模式，里面对对inner join的之上agg下推到join之下进行改写：

5 Support left join design: AggregateJoinTransposeRule method in open source adopted by aggPushDown, this method only supports inner join mode, which pushes down the agg above inner join to join and rewrites it:

比如 针对inner join:

For example, for inner join:

```
1  #修改前:
2  KapOLAPToEnumerableConverter
3      KapAggregateRel(group-set=[[0]], groups=[null], EXPR$1=[SUM($1)], ctx=[])
4      KapProjectRel(SELLER_ID=[$7], PRICE=[$8], ctx=[])
5      KapJoinRel(condition=[=($4, $107)], joinType=[inner], ctx=[])
6      KapJoinRel(condition=[=($4, $71)], joinType=[inner], ctx=[])
7      KapJoinRel(condition=[=($1, $66)], joinType=[inner], ctx=[])
8      KapTableScan(table=[[DEFAULT, TEST_KYLIN_FACT]], ctx=[], fields=[[0,
9      KapTableScan(table=[[DEFAULT, TEST_ORDER]], ctx=[], fields=[[0, 1, 2
10     KapTableScan(table=[[DEFAULT, TEST_CATEGORY_GROUPINGS]], ctx=[], field
11     KapTableScan(table=[[DEFAULT, KYLIN_CATEGORY_GROUPINGS]], ctx=[], fields
12
13  #修改后
14  KapOLAPToEnumerableConverter
15      KapAggregateRel(group-set=[[1]], groups=[null], EXPR$1=[SUM($5)], ctx=[])
16      KapProjectRel(LEAF_CATEG_ID=[$0], SELLER_ID=[$1], EXPR$1=[$2], LEAF_CATEG_ID
17      KapJoinRel(condition=[=($0, $3)], joinType=[inner], ctx=[])
18      KapAggregateRel(group-set=[[4, 7]], groups=[null], EXPR$1=[SUM($8)], ctx
19      KapJoinRel(condition=[=($4, $71)], joinType=[inner], ctx=[])
20      KapJoinRel(condition=[=($1, $66)], joinType=[inner], ctx=[])
21      KapTableScan(table=[[DEFAULT, TEST_KYLIN_FACT]], ctx=[], fields=[[
22      KapTableScan(table=[[DEFAULT, TEST_ORDER]], ctx=[], fields=[[0, 1,
23      KapTableScan(table=[[DEFAULT, TEST_CATEGORY_GROUPINGS]], ctx=[], fie
24      KapAggregateRel(group-set=[[0]], groups=[null], agg#0=[COUNT()], ctx=[])
25      KapTableScan(table=[[DEFAULT, KYLIN_CATEGORY_GROUPINGS]], ctx=[], fiel
```

那么，在left join的时候，需要处理KapAggregateRel(group-set=[[0]], groups=[null], agg#0=[COUNT()], ctx=[])中的这个COUNT()结果，因为这个在后面的project会重新计算一次，所以，为了匹配left join模式，会针对\$f5=[CAST(*(\$2, \$4)):DECIMAL(19, 4)]进行改写，改成\$f5=[CAST(*(\$2, COALESCE(\$4, 1))):DECIMAL(19, 4)]

Then, when left join, the COUNT () result in KapAggregateRel (group-set = [[0]], groups = [null], agg #0 = [COUNT ()], ctx = []) needs to be processed, because this project will be recalculated once later, so in order to match the left join pattern, it will be rewritten for \$f5 = [CAST (* (2 dollars, 4 dollars)): DECIMAL (19,4)] to \$f5 = [CAST (* (2 dollars, COALESCE (4 dollars, 1)): DECIMAL (19,4)]

举例：

example:

```
1  #修改前:
2  KapOLAPToEnumerableConverter
3    KapAggregateRel(group-set=[[0]], groups=[null], EXPR$1=[SUM($1)], ctx=[])
4      KapProjectRel(SELLER_ID=[$7], PRICE=[$8], ctx=[])
5        KapJoinRel(condition=[=($4, $107)], joinType=[left], ctx=[])
6          KapJoinRel(condition=[=($4, $71)], joinType=[inner], ctx=[])
7            KapJoinRel(condition=[=($1, $66)], joinType=[inner], ctx=[])
8              KapTableScan(table=[[DEFAULT, TEST_KYLIN_FACT]], ctx=[], fields=[[0,
9                KapTableScan(table=[[DEFAULT, TEST_ORDER]], ctx=[], fields=[[0, 1, 2
10                 KapTableScan(table=[[DEFAULT, TEST_CATEGORY_GROUPINGS]], ctx=[], field
11                 KapTableScan(table=[[DEFAULT, KYLIN_CATEGORY_GROUPINGS]], ctx=[], fields
12
13  #修改后:
14  KapOLAPToEnumerableConverter
15    KapAggregateRel(group-set=[[1]], groups=[null], EXPR$1=[SUM($5)], ctx=[])
16      KapProjectRel(LEAF_CATEG_ID=[$0], SELLER_ID=[$1], EXPR$1=[$2], LEAF_CATEG_ID
17        KapJoinRel(condition=[=($0, $3)], joinType=[left], ctx=[])
18      KapAggregateRel(group-set=[[4, 7]], groups=[null], EXPR$1=[SUM($8)], ctx
19        KapJoinRel(condition=[=($4, $71)], joinType=[inner], ctx=[])
20          KapJoinRel(condition=[=($1, $66)], joinType=[inner], ctx=[])
21            KapTableScan(table=[[DEFAULT, TEST_KYLIN_FACT]], ctx=[], fields=[[
22              KapTableScan(table=[[DEFAULT, TEST_ORDER]], ctx=[], fields=[[0, 1,
23                KapTableScan(table=[[DEFAULT, TEST_CATEGORY_GROUPINGS]], ctx=[], fie
24            KapAggregateRel(group-set=[[0]], groups=[null], agg#0=[COUNT()], ctx=[])
25            KapTableScan(table=[[DEFAULT, KYLIN_CATEGORY_GROUPINGS]], ctx=[], fiel
26
```

目前测试下来agg-pushdown代码存在一些bug，会顺带一起fix：

At present, there are some bugs in the agg-pushdown code tested, which will be fixed together:

1 count distinct 单列 的agg pushdown存在问题，查询结果不对

1 count distinct single column, there is a problem with agg pushdown, the query result is incorrect

2 agg-project merge代码存在问题（研发），导致不能下推

2 There is a problem with the agg-project merge code (R & D), which makes it impossible to push down

3 sum expression + count distinct在开启开关之后，代码逻辑存在问题，查询结果不对

3 sum expression + count distinct After the switch is turned on, there is a problem with the code logic, and the query result is incorrect

4 count函数下推的时候，代码逻辑存在问题，查询结果不对

4 When the count function is pushed down, there is a problem with the code logic and the query result is incorrect

已知限制：

Known limitations:

1 join的类型是inner join、left join

1 join的类型是inner join、left join

2 snapshot表需要放在join的末尾

2 The snapshot table needs to be placed at the end of the join

3 原来的agg-pushdown的scope会影响到当前的scope

3 The scope of the original agg-pushdown will affect the current scope

测试效果：

Test effect:

可以命中聚合索引 + snapshot联合查询：

Aggregate index + snapshot joint query can be hit:

KYLIGENCE Enterprise

🏠 首页

🔍 查询 ▾

分析

📜 查询历史

📦 数据资产 ▾

数据源

模型

快照

🖥 监控 ▾

批数据任务

⚙ 设置

数据源

🔍 搜索数据库名或表名

▼ 数据源: Hive

- 📁 DEFAULT (Default)

查询编辑器 query1 ×

1 select cast(3.1234 as decimal(8, 3)),SELLER_ID,sum(price) from TEST_KYLIN_FACT

2 inner join TEST_ORDER on TEST_ORDER.ORDER_ID = TEST_KYLIN_FACT.ORDER_ID

3 inner join TEST_CATEGORY_GROUPINGS as tcg on TEST_KYLIN_FACT.LEAF_CATEG_ID = tcg.LEAF_CATEG_ID

4 inner join KYLIN_CATEGORY_GROUPINGS as kgc on TEST_KYLIN_FACT.LEAF_CATEG_ID = kgc.LEAF_CATEG_ID

5 group by SELLER_ID

6 LIMIT 500

🔗

保存

查询结果

查询 ID: 03000d1e-aabc-cd41-21a3-ce86b1aafc40

查询对象: model_03

索引 ID: 1

快照: DEFAULT.KYLIN_CATEGORY_GROUPINGS

查询节点: 10.198.48.150:7070

查询耗时: 79.53s

查询扫描记录数: 144

查询结果记录数: 500

数据 可视化

🔍 筛选...

导出 CSV

EXPR\$0	SELLER_ID	EXPR\$2
3.1234	10000496	43.3900
3.1234	10000470	496.0600
3.1234	10000446	846.8300
3.1234	10000467	3547.0700
3.1234	10000716	1508.6100