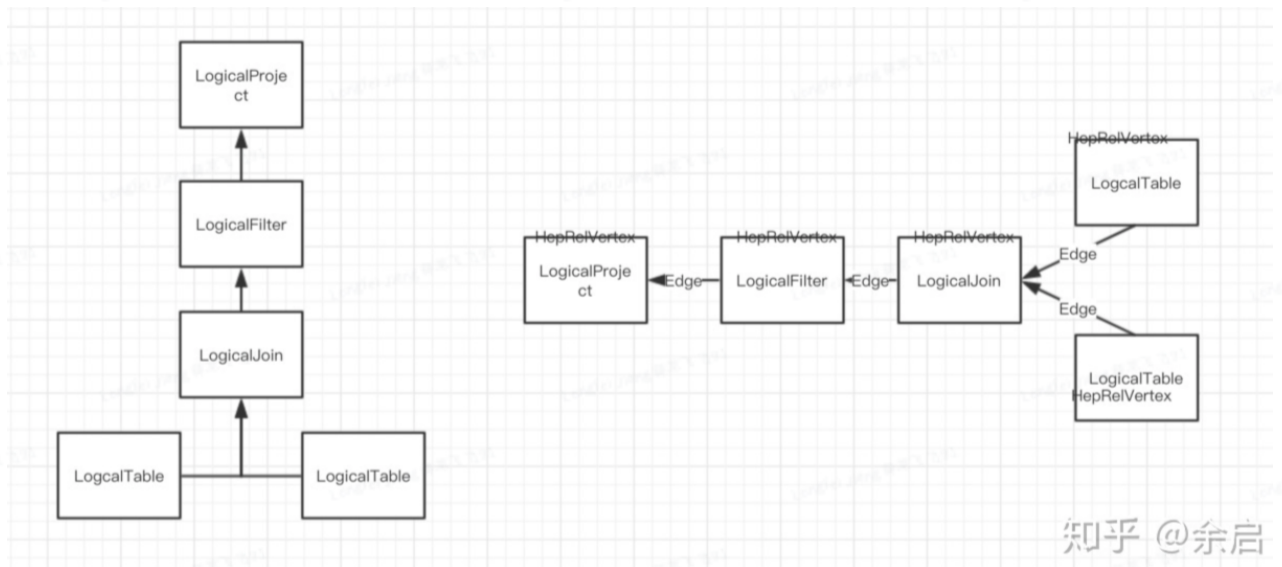


查询解析阶段发生OOM

Background

Calcite对SQL的解析划分成多步，第一步是将SQL解析为一个树状结构(AST)，第二步是基于这个树状结构，将其转换为一个有向无环图，该图中的各个节点由RelNode或RexNode组成。在生成有向无环图后，会利用Planner中加载的规则，对符合规则的图节点进行优化，将其替换为等价但效率更高的结构。



现在存在这样一个规则ProjectMergeRule，这个规则的作用是对两个相邻的Project节点进行合并，以减少存储的开销。Project节点对应SQL中的select，举个例子：

```
1 select
2   v + v
3 from
4 select
5   (case when SCORE > 60 then 1 else 0 end) v
6 from
7   STU_SCORE_RECORD;
8 转换为 ->
9 select
10  (case when SCORE > 60 then 1 else 0 end)
11  +
12  (case when SCORE > 60 then 1 else 0 end)
13 from
14  STU_SCORE_RECORD;
```

之前我们通过KapProjectMergeRule，针对case语句修过一版，修复方法是对重复的case语句进行简化合并，将"CASE(=(\$2, 0), null, CASE(=(\$2, 0), null, \$1))"转换为"CASE(=(\$2, 0), null, \$1)。

Root cause

VolcanoPlanner持有一个map对象mapDigestToRel，这个map存储了digest到RelNode的一个映射。ProjectMergeRule合并时，会通过RelOptRuleCall#transformTo调用VolcanoPlanner#registerImpl，这个方法会检查RelNode的digest，如果mapDigestToRel包含了相同的digest，则将Relnode放入到同一个RelSubset结构中。随着越来越多的合并，RelSubset中持有的RelNode也会越来越多，而RelNode中的digest会占用非常大的内存，最终导致OOM的出现。



从这里可以看到，产生了大量的RexCall节点。

Current object set: 601,167 instances of org.apache.calcite.rex.RexCall
2 selection steps, 19 MB shallow size, [Calculate retained and deep sizes](#) [Use retained objects](#)

Object	Retained Size	Shallow Size	Allocation Time (h:m:s)
org.apache.calcite.rex.RexCall	255 MB	32 bytes	n/a
digest (declared by org.apache.calcite.rex.RexNode) java.lang.String (0x550ec7) ["CASE(OR(CASE(CASE(OR(=(\$4, 'BN4'), =(\$4, '410L')), CAST(true):BOOLEAN, AND(<>(\$4, 'BN4'),			
hash = 0			
value char[] (0x8a6152) ["CASE(OR(CASE(CASE(OR(=(\$4, 'BN4'), =(\$4, '410L')), CAST(true):BOOLEAN, AND(<>(\$4, 'BN4'), <>(\$4, '410L')), CAST(false):BOOLEAN, null), =(CAST			
op org.apache.calcite.sql.fun.SqlCaseOperator (0x19c832)			
operands com.google.common.collect.RegularImmutableList (0x4f3bc8)			
size = 5			
offset = 0			
array java.lang.Object[] (0x6d74fb)			
element org.apache.calcite.rex.RexCall (0x51caa2)			
element org.apache.calcite.rex.RexCall (0x421af4)			
element org.apache.calcite.rex.RexCall (0x4f3bc9)			
element org.apache.calcite.rex.RexCall (0x421ad2)			
element org.apache.calcite.rex.RexLiteral (0x421ad1)			
type org.apache.calcite.sql.type.BasicSqlType (0x41d65b)			
type org.apache.calcite.rex.RexCall	255 MB	32 bytes	n/a
org.apache.calcite.rex.RexCall	255 MB	32 bytes	n/a
org.apache.calcite.rex.RexCall	125 MB	32 bytes	n/a

Design

来自Calcite社区的一个方案，[\[CALCITE-3774\] Add option to RelBuilder to prevent it from merging projects - ASF JIRA](#)，这是一个间接的修复方案。这个设计的目的就是减少多余节点的产生导致OOM。

这个方案定义了Project的复杂度，这个复杂度是对Project节点中子节点数量的加和，举个例子：这个例子中表达式：a+b+c+d有7个子节点，复杂度即为7。

- 1 Project(a+b+c+d AS w, b+c+d+e AS x, c+d+e+f AS y, d+e+f+g AS z) 【复杂度:28】
- 2 Project(w*x AS p, x*y AS q, y*z AS r) 【复杂度:9】

在进行Project合并前，会计算如果进行合并，则合并后的复杂度的值。

```
1 Project((a+b+c+d) * (b+c+d+e) AS p, (b+c+d+e) * (c+d+e+f) AS q, (c+d+e+f) * (d+e+f+g) AS r) 【(a+b+c+d) * (b+c+d+e)复杂度为15，总复杂度：45】
```

如果合并后的复杂度大于合并前两个Project复杂度的加和，则认为这个合并是不划算的，终止这次合并的操作。这样就减少了不必要RelNode的产生。

在实现时，需要在部分Calcite节点中增加一个变量nodeCount，用于在节点创建时对其子节点进行计数并记录。

参数：

kylin.query.project-merge-with-bloat-enabled，开关，系统级，默认值为true

kylin.query.project-merge-bloat-threshold，控制阈值，系统级，默认值为0，说明：将合并后Project包含的RelNode节点数计为m，合并前两个Project包含的节点数分别记为a, b。则当 $m > a + b + \text{threshold}$ 时不进行合并。