一文详解物化视图改写

作者：阿里云数据库OLAP产品部 云曦

预计算和缓存是计算机领域提高性能以及降低成本的最常见的手段之一。对于那些经常重复的请求，如果可以通过缓存回答，比重新计算结果或从速度较慢的数据存储中读取要快得多，消耗更少的系统资源。在数据库领域中，物化视图是预计算和缓存的自然体现。

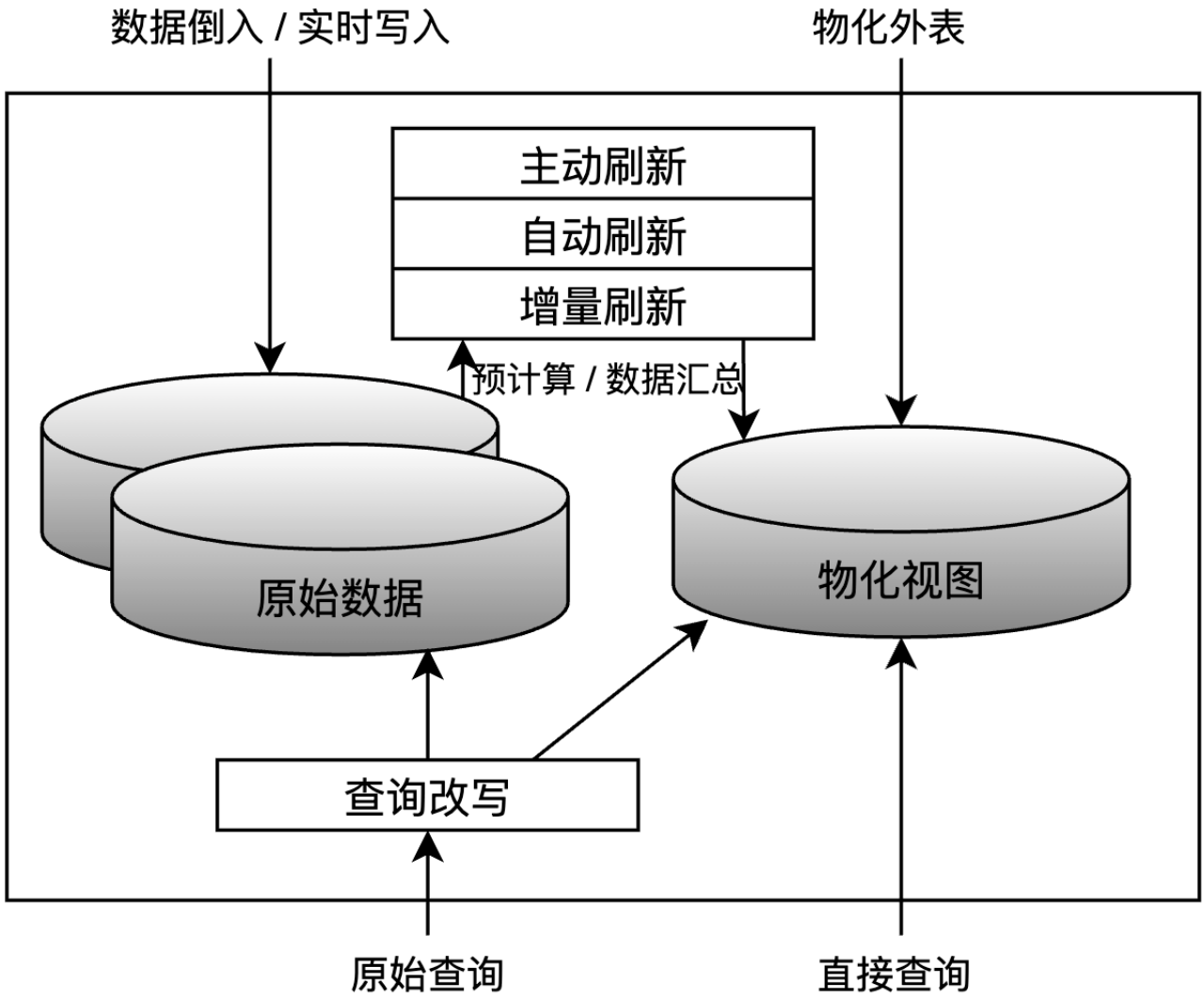
本文主要介绍什么是物化视图，以及如何实现基于物化视图的查询改写。

在第一部分，我们会简单介绍物化视图，并介绍基于物化视图的查询改写的用途。在第二部分，我们将介绍查询优化器使用物化视图进行查询改写的匹配和改写过程。最后，我们将介绍查询改写的几种实现方式，及其优缺点。

背景介绍

物化视图

物化视图是将查询结果预先计算并存储的一张特殊的表。"物化"(Materialized) 这个词是相对于普通视图而言。普通视图较普通的表提供了易用性和灵活性，但无法加快数据访问的速度。物化视图像是视图的缓存，它不是在运行时构建和计算数据集，而是在创建的时候预先计算、存储和优化数据访问，并自动刷新来保证数据的实时性。



对于数据仓库，物化视图最重要的功能就是查询加速。数据仓库中存在大量在大型表上执行复杂的查询，这些查询会消耗大量资源和时间。物化视图可以通过预计算的结果回答查询，消除昂贵的联接和聚合所带来的开销，大幅度改善查询处理时间，降低系统负载。对于可以预见并反复使用相同子查询结果的查询，物化视图特别有用。

为了实现物化视图的潜力，需要解决三个问题：

1. 物化视图选择：选择哪些查询和表构建物化视图；
2. 物化视图维护：减少物化视图更新成本和时间；
3. 物化视图运用：如何使用物化视图加速查询。

本文主要从查询优化器的角度，介绍使用物化视图加速查询背后的技术实现。

基于物化视图的查询改写

直接查询物化视图可以大幅度改善查询处理时间，但是需要用户修改查询语句。使用物化视图加速查询的一个重要问题是，如何采用一种系统化和自动化的方法，自动使用物化视图回答查询。通过这种透明改写，物化视图可以像索引一样添加或删除，而不会影响已有 SQL。

查询改写使得物化视图具有广泛的用途：

1. 物化视图可以透明地改写查询，无需改造业务就能使用物化加速查询；
2. 方便地应用缓存公共结果集，以及预计算等跨查询优化手段；
3. 对于数据仓库，数据集成场景，物化视图可以物化外表结果，屏蔽多个数据源的差异，实现本地副本或读写分离；
4. 查询改写结合自动构建物化视图，实现数据库自治加速。

查询改写的问题定义

为了实现更大范围的改写，查询改写通常被集成在优化器规则中。这有几个方面的好处。

首先查询改写可以利用优化器其他规则。依靠优化器其他规则将查询转换成标准和统一的形式，简化匹配流程，增加改写范围。其中比较重要的规则是列消除，谓词下推，解关联子查询等。解关联子查询规则允许物化视图对包含关联子查询的查询进行改写加速。

其次，优化器可以递归每一个子树能否被某个视图进行改写。每个相关视图都会对每个子树产生多次改写，一个查询语句不同部分可能被不同的视图改写。最终所有的改写都进入基于成本的选择器中，与原始查询一起选出最优的查询计划。

查询改写算法只需要考虑给定的查询表达式和视图，判断这个查询表达式能否从视图中计算出来，然后从视图上构造一个等价的补偿表达式，与原查询表达式等价。查询改写的范围应该尽可能大，查询改写的目标是使用少量物化视图改写大量查询。最终由优化器选择出一个最优的查询计划。

查询改写检查

优化器通过多种方式来改写查询。最简单的一种情况是物化视图的查询与查询完全匹配，符合这种查询重写类型的查询数量很少。为了进行更通用的匹配，优化器会尝试使用各种规则构造一个等价表达式改写查询。

查询改写检查包含两个步骤，改写匹配检查和构建等价表达式。一个查询能被视图回答需要满足下面两个条件：

1. 物化视图 Join 关系在查询中存在
2. 物化视图有足够的数据来回答查询

部分条件下即使不满足条件，视图也可能会被用于改写。例如视图包含一部分查询所需要的数据，可以使用物化视图回答部分查询，剩下的数据从原始数据中计算。这部分改写检查会放在高级改写规则中进行介绍。

具体来说改写检查会依次进行 Join 检查和 Ouput 检查，如果查询或视图中含有 Grouping By 和 Aggregation，还会进行额外的检查，如果需要，会尝试对视图进一步聚合。

Join 检查

当查询和视图的表的 Join 关系相同时，视图才可能包含查询需要的所有的行和列。一种简单的方式是只考虑没有子查询的 Inner Join，这时只用比较查询表和数量是否完全一致即可。或者通过一系列规则检查查询和视图的关系代数树包含的 Join 关系是等价的。更通用的一种方法是构建 Join Graph。Join Grpah 是一个以关系为结点，联接为边的图。Inner Join 的条件表示为无向边，Outer Join 的条件表示为有向边。Join Graph 由查询关系代数树构建而来，通过比较 Join Graph 就可以检查物化视图和查询包含相同的 Join 关系。

例如对于下面的查询

```
select c_custkey, c_name, l_orderkey, l_partkey, l_quantity
from (
  select l_orderkey,l_partkey, l_quantity
  from lineitem
  where l_orderkey < 1000
  and l_shipdate = l_commitdate
) subquery
```

```
join orders on subquery.l_orderkey = o_orderkey
left join customer on o_custkey = c_custkey
```

对于物化视图

```
create materialized view mview1
enable query rewrite as
select *
from lineitem
join customer On subquery.l_orderkey = o_orderkey
left outer join orders On o_custkey = c_custkey
```

查询和物化视图具有相同的 Join 关系，查询可以被改写

```
select c_custkey, c_name, l_orderkey,l_partkey, l_quantity
from mview1
where l_orderkey < 1000
and l_shipdate = l_commitdate
```

Output 检查

Ouput 检查保证物化视图有足够的数据回答查询，这包括 3 个步骤：

1. 验证查询所有输出需要的列能够从视图中计算出来
2. 优化器需要检查视图包含查询所有需要的行数，也就是视图的谓词的范围大于查询的范围
3. 补偿谓词能否从视图中计算得来

等价关系

查询改写能够通过查询中的等价关系扩展改写的范围。因为等值条件具有传递性，等价关系可以从等值条件或者代数关系推导而来。例如可以从 A=date_format(now(), '%Y%m%D') 和 B=date_format(now(), '%Y%m%D')，因为函数是确定性的，可以得到 A=B，如果我们还有 inner join 的条件 B=C，可以进一步得到 A=B=C。

可能会有某些其他的条件可以推导出等价关系，例如可以从 A=2，B=3，C=5 推导出 C = A + B。相比等值关系，寻找这种关系会使搜索过程更加复杂，而且我们无法实现所有的可能的相等关系，因此很少考虑这

样的表达式。等价关系推导只搜索等值条件，即使可能错过一些改写机会，这样的浅层搜索可以保证速度。

表达式检查

为了确保查询输出的列和补偿谓词需要的列都能从视图中计算出来，我们需要一种方法确定来自查询的表达式是否和视图中的表达式相等，或者能从中计算出来。表达式检查无法纯粹从语法上实现，两个表达式或者符号的文本相同，并不说明他们关系相等。例如别名的存在就会破坏基于语法的检查。

表达式检查需要通过等价关系和表的对应关系推导而来。如果视图和查询中所有的表都是唯一的，那么来自同一个表的列是等价的；如果视图和查询中有表出现多次，即 self join，那么查询到视图的表的映射就存在多种可能，每种可能都需要进行一次改写尝试。有了视图和查询之间列的对应关系和上一节的等价关系，通过代数系统确定来自查询的表达式是否和视图中的表达式相等，或者能否从中计算出来。

存在一些启发式的规则，允许一个表达式从另外的表达式中计算出来。比如算数规则从 x + 1 中计算出 x，例从 SUM(x) 和 COUNT(x) 计算出 AVG(x)，还存在一些 Function Dependency 规则，例如时间函数，可以通过返回的天的结果计算年等。

谓词检查

视图改写需要物化视图中存在查询所有需要的行数，即视图的谓词的范围大于或等于查询的范围。使用 Wq 表示查询的谓词，Wv表示视图的谓词，我们需要检查 Wq => Wv，其中 => 表示 Wq 满足 Wv 的含义。

优化器提取所有的谓词，并将他们转换为 CNF 的形式，即 W=P1 ^ P2 ^ ... ^ Pn。将谓词按照等值谓词，范围谓词和剩余谓词进一步分为 W= PE ^ PR ^ PU。PE，PR，PU 分别代表等值谓词，范围谓词和剩余谓词。谓词检查变成了 PEq ^ PRq ^ PUq => PEv ^ PRv ^ PUv，由于正交的性质，最终将问题分解成 PEq => PEv，PEq ^ PRq => PRv，PEq ^ PUq => PUv 三类检查。

查询的等值谓词用于推导等价关系。查询中任何视图不满足等价关系的谓词都构成补偿谓词，视图中存在查询无法满足的等价关系则无法改写。

范围谓词在优化器中可以存储为范围的形式，可以方便的计算差集。对于查询中每一个范围谓词，如果视图存在对应的范围精准匹配，不需要进行补偿，否则视图范围必须大于查询的范围，并通过差集构造一个补偿谓词。

查询表达式和试图剩余的谓词共同构成剩余谓词PE，只能进行精准匹配。查询中任何与视图不匹配的谓词都构成补偿谓词。查询与视图不匹配则无法改写。

所有的补偿谓词都需要通过表达式检查，确保可以从视图的输出中计算出来。

例如查询

```
select l_orderkey, o_custkey, l_partkey,l_quantity*l_extendedprice
from lineitem, orders, part
where l_orderkey = o_orderkey
and l_partkey  = p_partkey
and l_partkey >= 150 and l_partkey <= 160
and o_custkey = 123
and o_orderdate = l_shipdate
and p_name like ‘%abc%’
and l_quantity*l_extendedprice > 100
```

物化视图

```
create materialized view mview2
Enable Query Rewrite
as select l_orderkey, o_custkey, l_partkey,l_shipdate,
o_orderdate,l_quantity*l_extendedprice as gross_revenue
from lineitem, orders, part
where l_orderkey = o_orderkey
and l_partkey  = p_partkey
and p_partkey >= 150
and o_custkey >= 50 and o_custkey <= 500
and p_name like ‘%abc%’
```

查询可以被改写为

```
select l_orderkey, o_custkey, l_partkey, gross_revenue
from mview2
where l_partkey <= 160
```

```
and o_custkey = 123
and o_orderdate = l_shipdate
and gross_revenue > 100
```

Grouping 和 Aggregation 检查

如果视图和查询带有 GroupBy 或 Aggregation 函数，需要进行额外检查：

1. 检查查询请求的数据分组是否与物化视图中存储的数据分组相同，如果不同，优化器会尝试对物化视图进行汇总。
2. 如果需要进一步汇总计算，所有需要的列的都可以从视图输出中计算出来。

例如查询

```
select c_nationkey, sum(l_quantity*l_extendedprice)
from lineitem, orders, customer
where l_orderkey = o_orderkey
and o_custkey = c_custkey
group by c_nationkey
```

物化视图

```
create materialized view mview3
enable Query rewrite as
select o_custkey, count_big(*) as cnt, sum(l_quantity*l_extendedprice) as
revenue
from lineitem, orders
where l_orderkey = o_orderkey
group by o_custkey
```

查询可以被改写为

```
select c_nationkey, sum(revenue)
from customer join mview3 on c_custkey = o_custkey
group by c_nationkey
```

如果分组列表不同，只能改写 Group By 在查询最外部的情况，否则无法进行进一步聚合，无法满足第二个改写要求。只有特定的聚合函数支持进一步聚合，常见的聚合函数有 MIN，MAX，SUM，COUNT。

例如如下查询

```
select o_custkey, count(*) as cnt, sum(l_quantity*l_extendedprice) as revenue
from lineitem, orders
where l_orderkey = o_orderkey
group by o_custkey
```

物化视图

```
create materialized view mview4
enable Query rewrite as
select o_custkey, l_partkey, count(*) as cnt, sum(l_quantity*l_extendedprice)
as revenue
from lineitem, orders
where l_orderkey = o_orderkey
group by o_custkey, l_partkey
```

可以被改写为

```
select c_nationkey, sum(cnt) as cnt, sum(revenue) as revenue
from mview4
where c_custkey = o_custkey
group by o_custkey
```

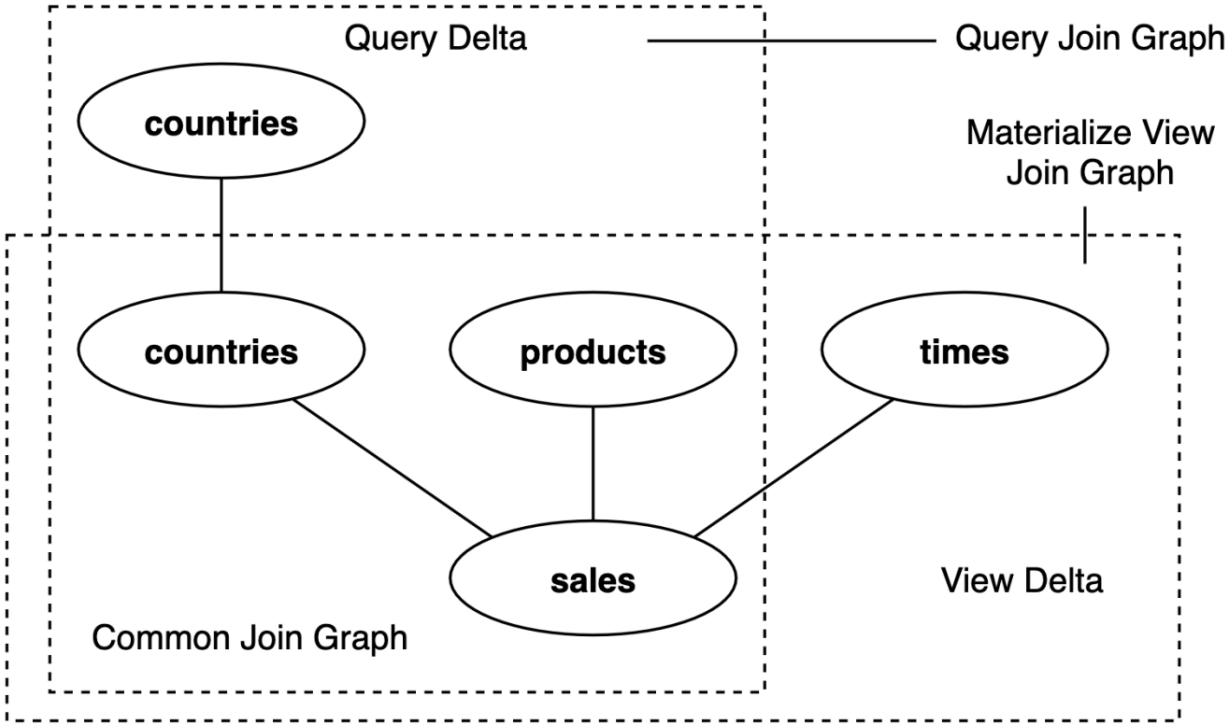
如果聚合函数在视图中不存在，可以通过一些规则进行计算，例如从 SUM(x) 和 COUNT(x) 计算出 AVG(x)，从 SUM(x) + SUM(y) 中计算出 SUM(x + y) ，这些依赖表达式改写检查中启发式的规则。

高级改写规则

如果物化视图和查询不满足上一节的改写规则，还可以通过其他规则进行转换。

Join 补偿

如果只考虑 Inner Join，视图和查询的 Join 关系 不一致有两种情况：查询比视图包含更多的联接，或者视图包含更多的联接。



如果查询比视图包含更多的表，我们可以简单把缺少的 Join 添加在视图之上，使其满足改写要求。Join 条件会通过谓词改写正确的添加进来，优化器也可以通过后续规则调整计划。

Join 补偿使优化器可以更早的进行匹配改写，减少改写的尝试次数。

例如如下查询

```
select c_custkey, c_name, l_orderkey,l_partkey, l_quantity
From lineitem, customer, orders
Where l_orderkey = o_orderkey
And o_custkey = c_custkey
Where l_orderkey between 1000 and 1500
And l_shipdate = l_commitdate
```

物化视图

```
Create materialized view mview5
Enable query rewrite as
Select l_orderkey,l_partkey, l_quantity
From lineitem, orders
Where l_orderkey = o_orderkey
And o_orderkey >= 500
```

可以被改写为

```
Select l_orderkey, l_partkey, l_quantity
From mview5 join customer on o_custkey = c_custkey
Where l_orderkey between 1000 and 1500
And l_shipdate = l_commitdate
```

Join 消除

如果一个 Join 出现在视图中但是没有出现在查询中，可以尝试使用 Join 消除规则。Join 消除是优化器优化中的一项常见方法，如果 Join 满足以下5个条件，则表 T1 在与表 T2 的联接中时不变的：

1. 联接条件是一个简单的等值条件
2. T1.fk 是 T2.pk 的外键
3. T1.fk 满足非 null 约束
4. T2 没有任何的谓词
5. T2 在与 T1 以外的表的联接是不变的

这意味着表 T1 在与 T2 的 Join 关系不会影响 T1 的结果，视图中的 T2 可以在 Join Graph 中忽略。不变联接的存在允许在基础物化视图上创建更大的并集或超集，从而允许物化视图包含更大的预计算，也可以改写更多的查询。

例如查询

```
Select l_orderkey, l_partkey, l_quantity
From lineitem
Where l_orderkey between 1000 and 1500
And l_shipdate = l_commitdate
```

物化视图

```
Create materialized view mview6
Enable query rewrite as
Select c_custkey, c_name, l_orderkey,l_partkey, l_quantity
From lineitem, orders, customer
Where l_orderkey = o_orderkey
And o_custkey = c_custkey
And o_orderkey >= 500
```

可以被改写成

```
select l_orderkey, l_partkey, l_quantity
from mview6
where l_orderkey between 1000 and 1500
and l_shipdate = l_commitdate
```

Join 派生

如果存在 Outer Join，视图和查询 Join 关系不一致，可以尝试利用 Join 派生性，从物化视图中的联接重新计算查询中的联接。例如，能从物化视图中的 left Outer Join 的结果里，计算 Inner Join，Anti Join 的结果，Inner Join 又能进一步计算 Semi Join，这样就能使用物化视图过滤某些行来回答不同具有不同 Join 关系的查询。

Join 派生可以扩展改写的范围，允许优化器将基于物化视图的改写与解关联的规则规则结合，改写带有 IN，EXISTS 等的查询，也可以避免其他优化规则对 Join 的调整，例如 EliminateOuterJoin 和 PredicatePushDown 可能会将 outer join 优化成 inner join。

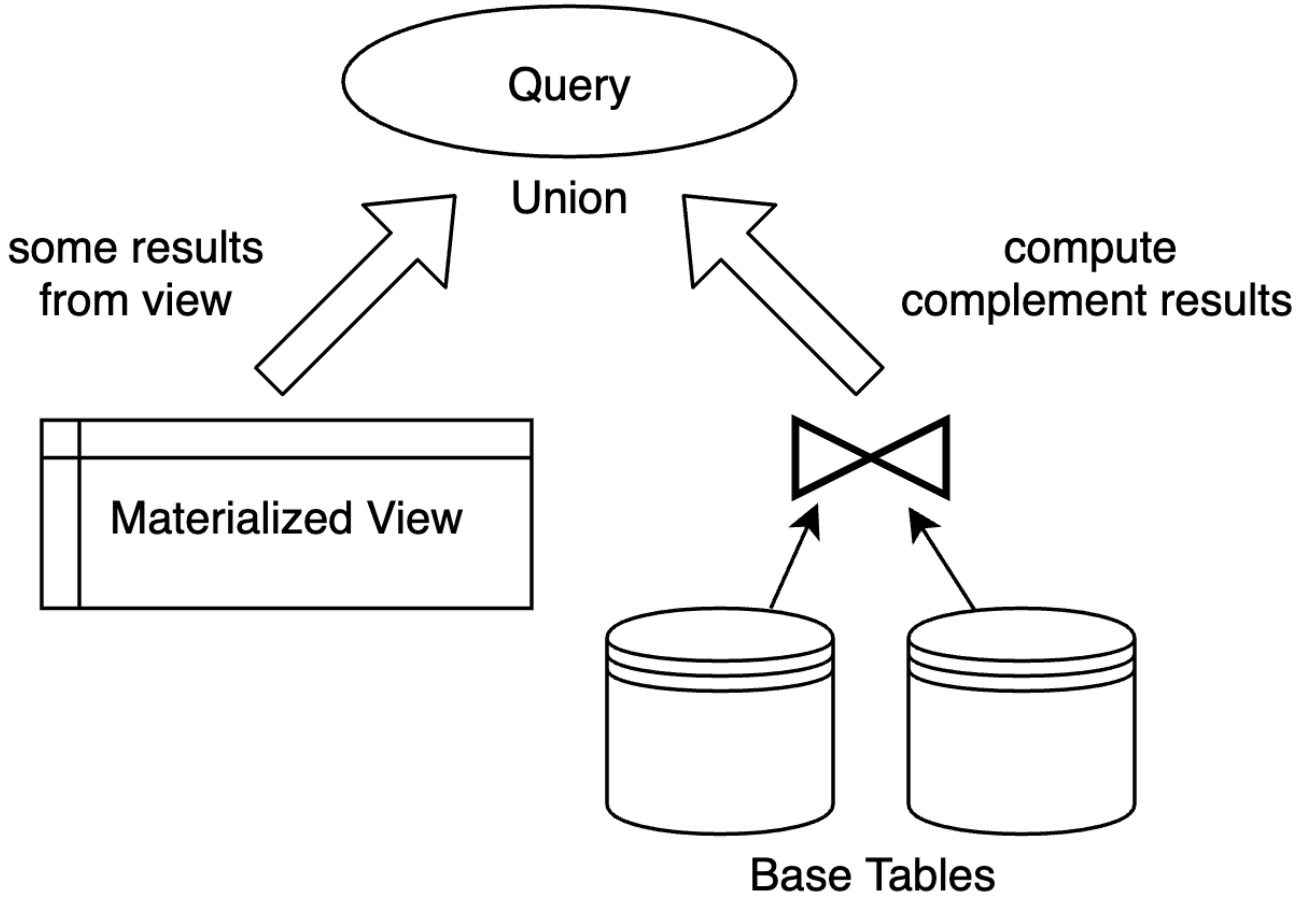
Union 改写

物化视图只包含一部分查询所需的数据，也可以用于查询改写。在很多场景下，物化视图不会也无法存储全部的数据。

一个典型的情况是，数据在不断的写入，但是写入只发生在最近一段时间内。在持续刷新的表上构建全量物化视图，这可能导致因为数据插入视图频繁刷新，产生高昂的刷新成本，甚至视图因为持续刷新而完全

不可用。更好的做法是构建一个 T+1 条件刷新的物化视图，存储不变的数据，可以降低刷新成本。

另一种常见的情况是，数据仓库存储全量的数据，而查询集中在最近几个月的数据，构建全部数据的物化视图成本过于高昂，物化视图只构建最近几个月数据，也能改写绝大多数查询。



Union 改写会尝试使用视图回答部分查询，减少查询中实时计算的数据量。Union 改写可以进一步应用 Aggregation 改写，支持使用物化视图部分数据回答聚合查询。

例如查询

```
select l_orderkey, l_partkey, l_quantity
from lineitem
where l_orderkey l_orderkey > 500 and l_orderkey <= 1500
and l_shipdate = l_commitdate
```

物化视图

```
create materialized view mview8
enable query rewrite
as
select l_orderkey, l_partkey, l_quantity
from lineitem
where l_orderkey > 1000
and l_shipdate = l_commitdate
```

改写结果

```
select l_orderkey, l_partkey, l_quantity
from lineitem
where l_orderkey > 500 and l_orderkey <= 1000
and l_shipdate = l_commitdate
union
select o.shippriority
from mview8
where l_orderkey > 1000 and l_orderkey <= 1500
```

查询改写的实现

视图改写通常有三种查询改写的实现方式：

1. 基于语法的改写
2. 基于规则的改写
3. 基于结构的改写

基于语法的改写

文本匹配或者语法匹配是最简单的改写方法，将查询的文本与物化视图的文本或语法树进行比较，完全匹配可以进行改写。这种改写只能匹配完整的查询语句或子语句，细微的变化就会导致查询无法改写，适用的范围很小。基于语法的改写虽然简单，但是效率很高，改写的成本可以忽略不计。

基于规则的改写

基于规则的改写和其他优化器规则相同，针对不同 Pattern 的查询和视图编写不同的规则，寻找等价的替代关系树。

最简单的一条规则就是直接比较子查询和视图的计划，如果相同就能改写。高级的改写规则不需要物化视图等同于被替换的计划，会尝试计算补偿谓词，构建等价查询表达式。例如 Join 改写，比较 Join 查询的子表达式是否和视图 Join 的某个子表达相同或者能否从中计算出来，每一个Join子表达式都存在映射关系，最后检查补偿表达式能否从视图中计算得到。

基于规则的改写可以实现大量重写，实现也比较简单，改写匹配速度快，但是也存在局限性。这种改写依赖转换规则来寻找等价关系，因此需要穷举所有可能的转换关系来实现复杂视图的重写。一些复杂的视图不可能穷举所有的等价关系，例如存在很多的 Join 联接或者复杂的 Project 关系，基于规则的改写适用的范围取决于规则的数量。

基于结构的改写

基于结构的改写与基于规则的改写相反，通过提取查询中的特征，使用一套规则进行匹配改写。优化器将查询表示为 SPJG 标准形式 (Join-Select-Project-GroupBy)，提取查询中的 Join，Projects，Filters，Grouping 和 Aggregations 五种表达式，分别与物化视图对应的表达式进行匹配和改写。

这个方法是由微软在 2001 年 SIGMOD 论文《Optimizing queries using materialized views: A practical, scalable solution》系统化的提出。这种方法可以改写包含可以改写包含 Join，Filter，Project 的任意查询的方法，运用一系列的步骤匹配并得到补偿表达式。还可以进一步改写含有 Aggreagtion 的查询，在需要时添加 Aggregation 节点返回进一步汇总的结果。

基于结构的改写很容易扩展，例如改写 Outer Join 和子查询等，可以完成几乎全部的改写。但是搜索成本较高，尤其是在查询复杂，改写尝试次数很多的情况下。

关于我们

AnalyticDB 是阿里巴巴自主研发、唯一经过超大规模以及核心业务验证的 PB 级云原生数据仓库。

AnalyticDB 在去年10月份推出了物化视图功能，目前的版本支持定时全量刷新和查询改写。AnalyticDB 完整实现了基于文本的匹配和基于结构的匹配，可以改写包含 Join，Filter，Project，Group By 的任意查询，支持 Aggregation Rollup，Union，Subquery，Outer join 等高级改写规则。ADB 会在未来的几个版本内上线基于规则的匹配，提高改写的效率。并在未来扩展更多的改写手段，例如 Grouping sets 改写支持，使物化视图有传统数仓中 Cube 的能力。

我们是AnalyticDB for MySQL的优化器团队，负责 OLAP 分布式查询优化器相关研发工作。欢迎投递简历到 jiannan.jjn@alibaba-inc.com，期待与你共同打造世界一流的查询优化器。

参考资料

1. Goldstein J, Larson P Å. Optimizing queries using materialized views: a practical, scalable solution[J]. ACM SIGMOD Record, 2001, 30(2): 331-342.
2. Bello R G, Dias K, Downing A, et al. Materialized views in Oracle[C]//VLDB. 1998, 98: 24-27.
3. Zhou J, Larson P A, Goldstein J, et al. Dynamic materialized views[C]//2007 IEEE 23rd International Conference on Data Engineering. IEEE, 2007: 526-535.
4. Jindal A, Qiao S, Patel H, et al. Computation reuse in analytics job service at microsoft[C]//Proceedings of the 2018 International Conference on Management of Data. 2018: 191-203.
5. Calcite: Materialized View.
6. Oracle: Database Data Warehousing Guide: Advanced Query Rewrite for Materialized Views.
7. Redshift: Automatic query rewriting to use materialized views.
8. Snowflake: Creating and Working With Materialized Views.