一种用于搜索可能响应查询的候选实化视图的索引*)

陈长清[1] 程 恳[2] 冯玉才[1]

(华中科技大学数据库与多媒体技术研究所 武汉430074)[1] (华中科技大学成教学院 武汉430074)[2]

An Index for Searching Candidate Materialized Views Possibly Answering Queries

CHEN Chang-Qing[1] CHENG Ken[2] FENG Yu-Cai[1]

(Institute of Database and Multimedia Technology[1], Adult Education College[2], Huazhong University of Science & Technology, Wuhan 430074)

Abstract Materialized views can be used to improve the performance of queries. When there are large amounts of views, it will take much time to sequentially search views that possibly answer queries. A level index is presented to partition views into much smaller subsets and can be used to separate those candidate views fast.

Keywords Materialized view, Aggregated query, Level index

1 引言

实化视图是存储了实际数据的视图,在响应查询时如果能直接利用实化视图,那么就可以避免相应的重新计算,从而提高查询性能。对一个复杂查询,一个实化视图往往只能替代查询的一部分(子表达式),在构造一个重写查询时,这种视图替代过程将被多次调用。当实化视图的数量很多时,如果每次视图替代都要把所有视图检查一遍,那将花费大量时间,从而降低利用视图响应查询的效果。

为了支持决策查询,我们在国产数据库系统DM3的基础上研制了DM_DW数据仓库系统[8]。本文首先描述了在DM_DW中判断一个视图能否用于替代一个查询子表达式的测试条件,即使视图涉及的关系表不全在查询子表达式的关系表中。然后介绍层次索引并利用该索引来快速找到可能用于替代查询子表达式的视图,它和文[6]中过滤树(Filter tree)的不同点在于:它增加了索引条件,使搜索得到的可能响应查询的视图数进一步减少。实验结果表明层次索引的搜索时间随着视图数量的增加缓慢地增长,效率比过滤树有了显著提高。

2 单视图替代

实化视图相对于(虚)视图来说是在数据库中存有实际数据的视图。DM_DW数据仓库所支持的视图必须定义在单块(single-block)的SQL语句上,SQL语句由选择(select)、投影(project)、连接(join)和分组(group by)子句组成,不包括union,这个视图被称做SPJG视图。这些视图是包语义(bag semantics)的,即实化视图中有重复元组。不失一般性,我们假设视图都定义在基表(base table)上,并且不包括子查询。我们把输出列分为非聚集列和聚集列,聚集列必须包含一个记数列count(*),它用于增量维护,其它的聚集函数则限制为sum、min和max;通常非聚集列必须是分组列的子集,为了保证聚集视图的增量维护,所有非聚集列一起作为视图的关键字,因此非聚集列必须等于分组列。

给定一个SPJG形式的查询子表达式,从实化视图中,找出可能用于计算该表达式的视图,并利用该视图构造出与这个表达式等价的替代表达式,这一过程叫做视图替代(view substitute)。如果要求一个SPJG表达式只能用一个视图完全替代,这种替代就叫做单视图替代(single view substitute)。一个SPJG查询表达式可以看作由一个SPJ查询再加一个group-by操作组成,视图也是类似的。一般来说,如果一个实化视图V满足以下条件,可以认为一个SPJG查询Q能从该视图中计算出来得到重写查询Q'。

1. 视图的SPJ部分包含满足查询的SPJ部分的所有元组,并且这些元组都可以从视图中选择出来。

2. 视图不做聚集或者聚集比查询弱,即查询结果可以通过对视图作进一步聚集得到。

3. 进一步聚集所要的分组列都在视图的输出中。

4. 计算查询输出所需要的列都可以从视图的输出中得到。

2.1 查询(SPJ部分)所需要的元组可以从视图中选出

我们用P(Q)、P(V)和P(Q')分别表示查询Q、视图V和重写查询Q'的谓词条件(即WHERE子句),假设查询和视图的表集相同,判断查询所需要的元组都可以从视图选出来而可能得到重写查询Q',直观上就是要保证(P(Q)(=)P(V)∧P(Q'))成立。这是因为P(Q)的条件应比P(V)强,为了保证重写查询Q'和初始查询Q有相同的结果,它们作用在基表上的谓词条件必须相同,因此要用P(Q')加强P(V)。P(Q')是重写查询Q'施加在视图V上的条件,它的谓词也称作用于视图V上的补偿谓词。一个简单的判断方法是检查P(V)中的每一个谓词$P_{v,i}$是否与σ[f(Q)]中的某一个谓词$P_{Q,i}$完全匹配,这种匹配是纯句法上的,它的缺点是没有考虑语义,因而句法上的小差别就会导致不匹配:如谓词(A>3)和(3<A)就会不匹配,(A=B and B=C)和(A=C and B=C)不匹配,(A+B)和(B+A)也不匹配。

为了避免这些问题,我们在判断时综合利用了列的等价性、列的范围约束特性和操作符的交换性与结合性。本文不考虑带"or"的选择条件,并假设经过语法分析后,查询的选择条件已经转换成合取谓词。这样,σ[f(Q)]可表示成EP∧RP∧OP的形式,EP(Equality Predicates)包含了形式为$R_i.C_m=R_j.C_n$的列等价谓词,这里R_i、R_j是表,C_m、C_n是引用列;RP

*)本文研究得到科技部科技电子政务系统关键技术及应用系统的研究(项目编号2001BA110B01)资助。

(Range Predicates)包含了形式为 $R_i.C_m$ op c 的列范围谓词，这里 c 是一个常量，op 是操作符"<"，"<="，"="，">="，">"；OP(Other Predicates)包含了除 EP 和 RP 外的所有其它谓词。根据 EP 中的列等价谓词可以生成等价类，一个等价类就是等值的列的集合，等价类的生成是很直观的。应用等价类，RP 和 OP 的谓词及输出中的某些列可以被替换。根据三类谓词的不同作用，我们判断($\sigma[f(Q)]<=>\sigma[f(V)]\wedge\sigma[f(Q')]$)时做下面三个测试：

$EP(Q)\wedge RP(Q)<=>EP(V)\wedge EP(Q')\wedge RP(Q')$(等价类匹配测试)

$EP(Q)\wedge RP(Q)<=>EP(Q)\wedge RP(V)\wedge RP(Q')$(列范围匹配测试)

$EP(Q)\wedge RP(Q)\wedge OP(Q)<=>EP(Q)\wedge OP(V)\wedge OP(Q')$(其它谓词匹配测试)

对等价类匹配检测，我们先检查视图中的每个等价类是否是查询中某个等价类的子集，然后在视图上添加补偿谓词，使查询和视图的各个等价类相等。在所有测试中，补偿谓词中的引用列都必须是视图的输出列，否则补偿谓词添加不成功，该视图不能替代查询。通过使用等价类可以抓住传递性的影响，如(A=B and B=C)和(A=C and B=C)可以匹配。在这个测试中考虑 RP(Q)，是因为它可以抓住查询的列在某个具体值上的等价性，如查询中的$(A=3)\wedge(B=3)$和视图中的(A=B)可以匹配(但反之不然)。视图和查询的等价类相同后，在第二个和第三个测试中可以利用等价类来替换 RP 和 OP 中的各引用列。

在列范围匹配测试中，对视图中带约束的一个等价类(不在任何等价类中的引用列称为平凡等价类)，在查询中找到与之匹配的等价类，然后检查查询的这个等价类的范围是否包含在相应的视图等价类的范围中。如果不符合，列范围约束检测失败，这个视图不能用于替代该查询表达式。在这个过程中，可以得到需要作用在视图上的补偿谓词。必须注意，如果查询中某个带约束的等价类不与视图中任何带约束的等价类匹配，它将直接作为补偿谓词。

在其它谓词匹配测试中，我们主要考虑利用等价类和操作符(如+、*、>和<)的交换性。对视图中的一个其它谓词，它必须和查询的一个其它谓词匹配，即它们的引用列属同一等价类，有相同的操作符，并且引用列和操作符的出现顺序一致，除了可交换操作符两边的引用列可以交换顺序。在这项测试中还考虑了 RP(Q)，它可以使如查询的$(A=1)\wedge(B=2)$和视图的(A+B=3)匹配。没有被匹配的查询的其它谓词作为补偿谓词。同样，补偿谓词中的引用列必须是视图输出列在相应等价类中的一列。

三个测试都通过后，$(P(Q)<=>P(V)\wedge P(Q'))$成立，查询所需要的元组都可以从视图中选出。

2.2 利用实化视图替代 SPJG 查询的其它条件

下面考虑利用实化视图替代 SPJG 查询的其它三个条件，这些条件同样需要结合等价类。第二个条件要求查询能通过对视图的进一步聚集得到，也就是视图不包含聚集，或聚集程度比查询低。用等价类表示各分组列，那么查询的每个分组等价类中必有一列在视图的某个分组等价类中。当它们分组列相同时，不用做进一步聚集。第三个条件要求对视图做进一步聚集时，查询所要的分组都在视图的输出中。因为我们的视图定义已保证输出的非聚集列和分组列是相同的，所以二、三两个条件在本文中是一致的。

下面考虑第四个条件。在等价类条件下，一个 SPJG 查询输出的非聚集列应包含在视图输出的非聚集列中，对聚集列的处理则有所不同。例如，如果 $SUM(b_i)$在查询 Q 中，那么 $SUM(b_i)$也应当在视图 V 中，并且在重写查询 Q' 中要用 SUM(N)代替以合并 V 中 $SUM(b_i)$的值，N 是 V 中 $SUM(b_i)$的别名；如果 $count(b_i)$在查询 Q 中，$count(b_i)$也应当在 V 中，在 Q' 中则用 SUM(N)以合并 V 中 $count(b_i)$的值，N 是 V 中 $count(b_i)$的别名。总之对每个 $f_i(b_i)\in Q$，应当有对应的 $f_j(b_j)$和 $f_k(b_k)$分别在 V 和 Q' 中，我们用 $p_Q(V)$和 $a_Q(V)$分别表示 Q' 的非聚集列 p(Q')和聚集列 a(Q')以说明上述转换。

2.3 带多余表的视图

尽管有些视图包含了不属于查询的表，称为多余表(more tables)，但它们仍有可能用于回答查询。这里我们假定除了做连接的列以外，多余表的其它列没有参与分组和聚集操作。对一个引用了表 $R_1,R_2,\cdots,R_n$ 的查询和有一个多余表的视图 M_1，如果表 R_i 中的每一元组刚好只和表 M_1 中的一个元组作连接，R_i 和 M_1 之间的连接就叫无损连接，这样可以将 M_1 视为以 M_1 中的列对 R_i 作扩展，即和 M_1 做连接不会使结果行发生增减。表 R_i 中一个非空外键和表 M_1 中关键字之间的等价连接就有这种特征。但实际上，只要保证那些查询需要的元组，而不是所有的元组具有无损的特征，因此对允许为空的外键，如果查询包含了该外键列上的范围约束谓词，那么外键允许为空。例如有一个基于表 R_i 和 M_1 的视图，用 $R_i.F=M_1.P$ 连接，F 作为 P 的外键，P 是 M_1 的主键。现在有一个表 R_i 上的谓词为 $R_i.F>15$的选择查询，因为 $R_i.F$ 列中为空的元组在查询中将被去除，R_i 和 M_1 的连接还能保持 R_i 的基数，视图仍可被接受。

多余表不会影响视图结果，但它要替代一个 SPJG 查询还必须通过前面的检测，然而这些检测假设视图和查询引用了相同表集合。为了使它们相同，将多余表 $M_1,M_2,\cdots,M_m$ 加到查询上，并施以相应的主外键连接与 $R_1,\cdots,R_n$ 作连接，因为这些连接是无损连接，所以不会改变查询结果。实际上，这里只是通过修改查询等价类来模拟增加这些多余表，然后仍按前面的条件进行检测。

3 层次索引

为了加速视图替代，我们将每个视图的描述保存在内存中，视图描述包含了进行视图替代的所有信息(条件)，如它的表集合(source table set)，输出列集合(output column set)等。即使这样，由于每次调用视图替代算法时，都要对所有视图进行比较，因此当视图数目很大时，匹配算法执行得很慢。为此，我们引入了层次索引，它是一个内存索引，可以快速搜索出所有可能用于回答查询的相关视图。

层次索引对应一个多路搜索树，每一级(层次)对应了不同的匹配条件。所有的叶子结点在同一层次上，叶子结点包含了视图描述的集合，它对应于满足从树根到叶结点的路径上的各种条件的所有视图。树的每个非叶结点包含了一个(key，d_pointer，h_pointer)对，每个关键字(key)是一组值；指针(d_pointer)指向下一级的一个结点，指针(h_pointer)指向同一级的(水平的)一个结点，这些结点在本层的条件不同，在上面的各个层次条件相同。需要指出 key 值允许为空，因为不是每个视图都具备所有的条件，层次索引的每一级子索引逐步地把视图集合划分成越来越小的分区。

3.1 划分条件

本文只讨论一个查询表达式能完全被一个视图取代的情况，因此这里不考虑视图被部分利用的情况。根据上述的单视图替代条件，我们有下面的索引条件，这些条件是视图可能响应查询的必要条件，条件中所说的查询可以是实际的查询或一个查询的子表达式，但都是 SPJG 的形式。

① *源表条件* 实化视图可以比查询有多余的表，因此查询的源表集必须是视图源表集的子集。

源表条件：查询的源表集必须是视图源表集的子集。

② *外键表条件* 从视图中可以去掉只有主外键连接关系的主表，保留外键表，即去掉视图中的多余表，那么剩下的表集必是查询源表集的子集。

外键表条件：去掉多余表的视图表集是查询源表集的子集。

③ *等价类条件* 根据等价类匹配条件，视图的每个等价类都应是查询的某个等价类的子集。

等价类条件：视图的每个等价类都是查询等价类的子集。

④ *输出列条件* 查询的输出列都必须能通过视图的输出列计算出来。假设一个视图的输出列表是{A,D,G},{E},{B},{C, H}，具有下划线的列是实际的输出列。逻辑上，我们可以认为视图输出所有这些列，也就是输出列表扩展为 A, D,G,E,B,C,H。有一个查询的输出列是 A, B, C，查询的列等价谓词(equal-join)产生了等价类：{A, D,E},{B, F},{C}，因此查询的输出列表用这三个等价类表示，而每个等价类中都至少有一列在视图的扩展输出列表中，该查询的输出列都能通过视图的输出列计算出来。

输出列条件：对于查询输出列表中每个等价类，至少有一个列在视图的扩展输出列表中。

⑤ *分组列条件* 一个聚集查询不能从一个聚集视图计算出来，除非查询的分组列是视图分组列的子集(在等价类下)。这和输出列条件恰好相同。

分组列条件：对查询分组的等价类输出列表，每个等价类中至少有一个列在视图的扩展分组列表中。

⑥ *列范围约束* 根据列范围匹配条件，在添加补偿谓词后，查询和视图的列范围应一致。和等价类条件类似，因为补偿谓词的引用列必须是视图的输出列，对查询的每个带约束的等价类，其中至少有一个列应在某个带约束的视图等价类或视图输出列中。因此，我们可以把带约束的视图等价类和视图输出列结合起来考虑，将它们合并得到扩展约束列表。

列范围约束条件：每个带约束的查询等价类，至少要有一个列在视图的扩展约束列表中。

⑦ *其它谓词条件* 其它谓词是除了列等价谓词和列范围谓词以外的谓词，根据其它谓词的匹配条件，在添加补偿谓词后，查询和视图的其它谓词应一致。因为它们的操作符出现顺序相同，我们将谓词转换成字符串，并删去引用列，这样查询和视图的其它谓词列表就由若干这样的字符串组成，视图的字符串集合是查询字符串集合的子集。

其它谓词条件：视图的其它谓词列表是查询其它谓词列表的子集。

我们可以看到，上述的每个条件都可以作为索引的基础来划分视图。这些条件是相互独立的，可以任意组合构成层次索引。我们选取的顺序是视图的外键表，视图的源表集，等价类，输出列，列范围约束，其它谓词，输出表达式；对聚集视图，还有两个额外的条件：分组列和分组表达式。下面的图1是一个层次索引的例子。

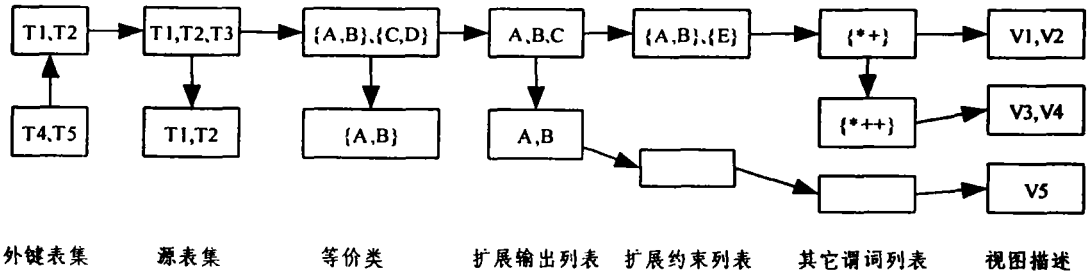

图1 层次索引示意图

进行索引树的搜索时，是否继续沿着某一结点的指针继续前进，这是由搜索条件决定的，条件是：指针所指结点的关键字是给定搜索关键字的子集(超集)或是等价。我们可以对结点逐个顺序扫描比较，但当结点很多时，为了避免顺序扫描，我们把结点组成一个格的结构，可以很容易地找到一个搜索关键字的子集(超集)。

3.2 关键字的格结构

同一层次上各结点关键字之间的子集和超集关系可以用偏序(partial order)表示，它们一起形成了格结构，每个层次都可以包含一个格结构。为了表示偏序关系，在结点中保留 d_pointer 指针，去掉 h_pointer 指针，同时增加两类指针集合：超集指针集合和子集指针集合。一个结点 V 的超集指针指向 V 的关键字的最小超集所对应的各结点，结点 V 的子集指针指向 V 的关键字的最大子集对应的结点。例如：结点 V_1、V_2、V_3、V_4的关键字分别是{A,B}、{A,B,C}、{A,B,D}、{A,B,C,D}，则结点 V1有两个超集指针分别指向 V_2、V_3，V_4有两个子集指针分别指向 V_2、V_3。没有子集的结点叫做根(root)，没有超集的结点叫做顶(top)。格结构还包含了一个顶结点头指针和一个根结点头指针。图2表示了一个保存了6个关键字：{A}、{B}、{A,B}、{B,E}、{A,B,C}、{A,B,D}、{A,B,C,D}的格结构。

结点 V 的最小超集结点的确定算法：

① 首先找出 V 所有超集结点，即 V 的关键字是其子集的所有关键字。

② 在这些超集结点中，子集结点指针没有指向 V 的其它超集结点的结点就是 V 的(直接)最小超集结点。

结点 V 的最大子集结点的确定算法：

① 首先找出 V 所有子集结点，即 V 的关键字是其超集的所有关键字。

② 在这些子集结点中，子集结点指针没有指向 V 的其它子集结点的结点就是 V 的(直接)最大子集结点。

搜索格结构是一个简单的递归过程，从顶结点出发顺着

子集指针可以搜索一个结点的超集，从根结点出发顺着超集指针可以搜索一个集合的子集，在搜索过程中应记住哪些结点已被访问过，以避免重复。

3.3 格结构中结点的插入和删除

在格结构中插入和删除结点需要小心处理超集指针和子集指针。当删除一个顶结点时，要把该顶结点的子集指针所指各结点的指向该顶结点的超集指针删除，子集结点升为顶结点，然后再删除该顶结点；当删除一个根结点时，要把根结点的超集指针所指各结点的指向该根结点的子集指针删除，超集结点降为根结点，然后再删除该根结点；中间结点的删除要复杂一些，需要同时考虑结点的超集指针和子集指针。结点的插入进行与此相反的操作，下面的图3是在图2插入结点{B,D,E}后得到。

在按格结构组织层次索引的各层后，就可以从格结构的顶结点或根结点开始，根据查询的不同条件进行检索。例如：对视图的源表集的格结构和一个查询，从顶结点开始递归检索查询源表集的超集。

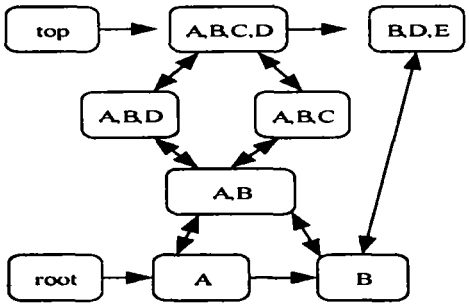

图2 格结构

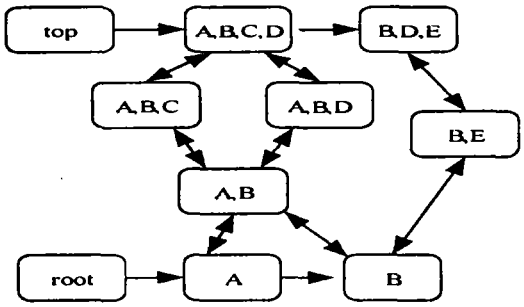

图3 插入结点{B,D,E}

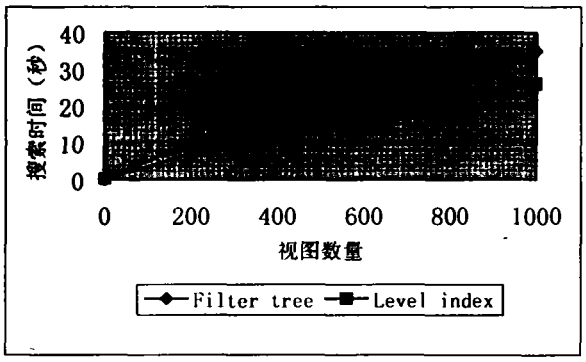

图4 层次索引和过滤树的搜索时间比较

表1 层次索引和过滤树的搜索结果(视图数)比较

MV	200	400	400	600	1000
FT	6	13	19	28	36
LI	4	9	12	19	26

4 实验

我们在国产数据仓库系统 DM_DW 上执行了这些算法。实验运行在一台 P Ⅲ 500、128M 内存的机器上。我们根据 TPC-H 基准测试生成一个大小为1G 的数据库，规模因子为1。在实验中，我们随机生成数量不同的实化视图，每个视图基于1到8个关系，分组属性从视图所涉及的关系属性中随机选取10%到30%。查询采用类似的生成方法，分组属性的选取比例为5%到20%，以增加实化视图被选中的比例。

我们将层次索引和文[6]中的过滤树进行了比较，图4表明：层次索引比过滤树节约了大约12%的搜索时间，表1表明：层次索引比过滤树的候选实化视图范围平均减少了30%，表中 MV 表示视图数，FT 表示过滤树，LI 表示层次索引。这是因为我们从使原始查询和重写查询保持等价的角度来考虑单视图替代，在层次索引增加了等价类条件和引入了新的列范围约束条件，更加细分了视图集合，这样尽管多了一个判断条件，但每个层次上需要比较的结点数减少了，并且进一步缩小可能用于响应查询的实化视图的范围。

结论 层次索引可以大大加快寻找潜在有用的实化视图的过程，从而减少利用实化视图响应查询的优化时间，提高优化效率。选择哪个视图响应查询使得查询的执行代价最小，还需要结合代价估算，这是我们进一步要做的工作。

参考文献

1 Agrawal S,Chaudhuri S,Narasayya V R. Automated Selection of Materialized Views and Indexes in SQL Databases. VLDB 2000. 496～505

2 Bello R G,et al. Materialized Views in Oracle. VLDB. 1998. 659～664

3 Chang J,Lee S. Query Reformulation Using Materialized Views in Data Warehousing Environment. In:First ACM Int. Workshop on Data Warehousing and OLAP(DOLAP), 1998. 54～59

4 Chaudhuri S,Krishnamurthy S,Potamianos S,Shim K. Optimizing Queries with Materialized Views. ICDE. 1995. 190～200

5 Cohen S,Nutt W,Serebrenik A. Rewriting Aggregate Queries Using Views. PODS, 1999. 155～166

6 Goldstein J, Larson P. Optimizing queries using materialized views: a practical, scalable solution. In: Proc. of SIGMOD, 2001. 331～342

7 王珊. 数据仓库与联机分析处理. 北京:科学出版社,1998

8 陈长清,冯玉才,袁磊. 国产数据仓库系统 DM_DW 的设计. 小型微型计算机系统, 2002, 23(5):596～599