ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.2, February 2009, pp.193−213
doi: 10.3724/SP.J.1001.2009.03416

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

实视图选择研究*

林子雨[1,2+], 杨冬青[1,2], 王腾蛟[1,2], 宋国杰[3]

[1](高可信软件技术教育部重点实验室(北京大学),北京 100871)

[2](北京大学 信息科学技术学院,北京 100871)

[3](机器感知与智能教育部重点实验室(北京大学),北京 100871)

Research on Materialized View Selection

LIN Zi-Yu[1,2+], YANG Dong-Qing[1,2], WANG Teng-Jiao[1,2], SONG Guo-Jie[3]

[1](Key Laboratory of High Confidence Software Technologies for the Ministry of Education (Peking University), Beijing 100871, China)

[2](School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

[3](Key Laboratory of Machine Perception for the Ministry of Education (Peking University), Beijing 100871, China)

+ Corresponding author: E-mail: cainiu@263.net

Lin ZY, Yang DQ, Wang TJ, Song GJ. Research on materialized view selection. *Journal of Software*, 2009, 20(2):193−213. http://www.jos.org.cn/1000-9825/3416.htm

Abstract: Definition of view selection issue in the field of data warehouses is presented, followed by the discussion of related problems, such as cost model, benefit function, cost computation, restriction condition, view index, etc. Then three categories of view selection methods, namely, static, dynamic and hybrid methods are discussed. For each method, some representative work is introduced. Finally some future trends in this area are discussed.

Key words: materialized view; view selection; data warehouse

摘 要: 定义了数据仓库领域的视图选择问题,并讨论了与该问题相关的代价模型、收益函数、代价计算、约束条件和视图索引等内容;介绍了 3 大类视图选择方法,即静态方法、动态方法和混合方法,以及各类方法的代表性研究成果;最后展望未来的研究方向.

关键词: 实视图;视图选择;数据仓库

中图法分类号: TP311 **文献标识码**: A

数据仓库是面向主题的、集成的、相对稳定的、反映历史变化的数据集合,用于支持管理决策[1].数据仓库中包含了来自多个分布式的、自治的,并且可能是异构的数据源的数据,为了改善 OLAP(online analytical processing)查询分析的性能,这些数据通常以实视图的形式进行存储.实视图是一种包括查询结果的数据库对象,用来生成基于数据表求和的汇总表.但是,在实际应用中不可能实化所有视图.一方面是由于数据仓库中可

* Supported by the National Natural Science Foundation of China under Grant No.60473051 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant Nos.2007AA01Z191, 2006AA01Z230 (国家高技术研究发展计划(863))

Received 2007-10-30; Accepted 2008-07-09

能存在的视图的数量非常巨大,而系统可用空间则限制了可以被实化的视图的数量;另一方面是由于当基本事实表发生更新时,实视图也需要相应作更新,这就产生了视图维护代价的问题,实化视图数量过多,会导致视图维护代价过高.实视图选择问题[2]就是在满足限制条件(实视图空间限制或视图维护代价限制)的前提下,从大量的候选视图中选择一部分进行实化,从而提高查询的性能.本文中,我们将混用视图选择和实视图选择,二者是等价的概念.

1996 年,Harinarayan 等人发表了关于视图选择问题的研究论文,使得视图选择问题在数据库研究领域逐渐升温.此后,该问题吸引了越来越多的研究人员,新的研究成果不断涌现,产生了以文献[3–5]等为代表的静态选择方法、以文献[6–9]等为代表的动态选择方法和以文献[10]为代表的混合方法.

本文第 1 节定义视图选择问题,并讨论与该问题相关的代价模型、收益函数、代价计算、约束条件、视图组织和视图索引等内容.第 2 节介绍不同的视图选择方法,包括静态选择方法、动态选择方法和混合方法等.第 3 节作总结并展望未来的研究方向.

1 视图选择问题

定义 1(视图选择问题)[2]. 给定一个数据库模式 R、约束条件 T、查询集合 Q 和代价评估函数 C,视图的选择问题就是在 R 之上选择一个视图集合 V,使得代价 $C(R,V,Q)$最小化,并使得 V 中的视图满足约束条件 T.

视图选择问题涉及代价模型、收益函数、代价计算、约束条件、候选视图、视图粒度、视图组织、视图索引和应用环境等内容.针对这些问题,下面我们分别予以论述.

1.1 代价模型

定义 2(代价模型)[10]. 假设一个查询集合 $Q=\{q_i:i=1,2,\ldots,k\}$和实视图集合 $V=\{v_i:i=1,2,\ldots,n\}$,Q 中的查询 q 可以从 V中的视图得到最好的响应.现在令 $QC(q,V)$表示用实视图集合 V中的视图来回答查询 q的代价,$UC(v,V)$表示在实视图集合 V 中更新 v 的代价,f_q 表示查询 q 的频率,g_v 表示视图 v 的更新频率,S_v 表示视图 v 的尺寸,则有:

查询集合 Q 的总查询代价: $QC_{total}=\sum_{i=1}^{k} f_{qi}QC(q_i,V)$.

实视图集合 V 的更新总代价: $UC_{total}=\sum_{i=1}^{n} g_{qi}UC(v_i,V)$.

实视图存储总代价: $SC_{total}=\sum_{i=1}^{n} S_{vi}$.

考虑查询代价和视图维护代价的总代价:$TC(V)=QC_{total}+UC_{total}$.

需要说明的是,在用 V 中的视图回答 Q 中的查询 q 时,查询 q 可能用到 V 中的 1 个或多个视图.但是,Kotidis 等人在 Microsoft SQL Server 2000 项目的研究中,通过大量的实验发现,在实际应用中,一个查询由多个视图来共同回答的情形可能性不大[11],因此,一些研究(比如文献[8,10,11])都假设一个查询可以用一个唯一对应的视图来回答.

代价模型是视图选择问题中的一个关键内容,需要考虑的代价通常包含 3 个方面,即查询代价、视图维护代价和存储代价.不同的研究方法采用的代价模型也不同,有的研究(比如文献[2,12–14])只考虑查询代价,而不考虑视图维护代价;而有的研究(比如文献[3,15,16])则综合考虑查询代价和视图维护代价;同时,大部分研究都考虑了存储代价(比如文献[17–20]).

1.2 收益函数

在确定了代价模型以后,就可以确定选择某个视图进行实化所带来的收益,从而决定选择哪个视图进行实化.通常在每轮视图选择时,都是选择收益最大的视图.

定义 3(收益函数)[10]. 现在假设 c 是数据库模式 R 上的任意一个视图,V 是实视图集合,那么把 c 增加到 V

中所带来的收益,可以采用下面的收益函数计算得到,

$$B(c,V)=TC(V)-TC(V\cup c).$$

收益函数 $B(c,V)$的值通常用来作为是否选择某个视图进行实化的标准.文献[13]的研究发现,在某些时候,采用单位空间收益($B(c,V)/|c|$,其中$|c|$表示视图 c 的尺寸)作为标准可以取得更好的视图选择性能.

定义 4(收益函数的单调性)[3]. 假设视图选择收益函数 B,实视图集合 V 和未被实化的视图 $O_1,O_2,\ldots,O_m$,如果 $B(O_1\cup O_2\cup O_m)\leq\sum_{i=1}^{i=m}B(O_i,V)$,我们就说 B 对于 V 具有单调性.

直观地讲,所谓的收益函数的单调性是指在视图选择过程中已经被选中的视图的收益在以后的视图选择中不会发生变化[21].文献[22]的研究指出,当不考虑视图维护代价时,就不用担心视图更新引起的收益变化,这时视图选择收益函数具有单调性;但是,当考虑视图维护代价时,视图选择收益函数不具备单调性.换句话说,一个被选中的视图可能使前面被选中的视图的收益减小,因为当有更多的视图被实化时,选中的视图集合的总维护代价可能会减小.

一些研究(比如文献[2,12,13])之所以不考虑视图维护代价,就是因为当考虑视图维护代价时,视图选择收益函数会违反单调性[10],而这种单调性是贪婪算法的基础,也就是说,那些使用贪婪算法进行视图选择的方法在采用某种代价模型时,该代价模型必须保证视图选择收益函数具有单调性.文献[23]的研究也指出,由于考虑视图维护代价而带来的非单调性问题,使得基于单位维护代价的查询收益的视图选择贪婪算法可以产生任意坏的解.

1.3 代价计算

代价计算包括 3 个方面的内容,即查询代价的计算、视图维护代价的计算和存储代价的计算.

1.3.1 查询代价

查询代价的计算要受到很多因素的影响,比如实视图在某个属性上的聚类、索引的存在[12]和使用的连接算法等等.通常来说,并不是所有的查询都会请求一个视图中的所有记录,如果我们已经为一个实视图 v 建立了索引,那么用这个实视图回答一个查询 q 只需要 $O(1)$的时间.但是,如果 v 不存在索引,那么,为了回答 q 就必须访问 v 的所有记录,这时,无论 q 需要 v 中的 1 条记录还是全部记录,总体而言,查询代价没有什么区别[2].

对于需要聚集计算的查询,为了减小查询代价,可以采用哈希或排序的方法[24].在这种情况下,聚集计算的代价是内存大小和输入、输出记录数目比的函数[24].在最好的情况下,比如整个哈希表都可以放入内存中,只需要“一趟”输入就可以完成计算.在实际应用中,大部分聚集计算都只需要 1 趟或两趟输入来完成聚集计算.

连接查询的代价有两种计算方法,即积-代价模型及和-代价模型[24].采用不同的代价计算方法,会影响到视图选择问题的复杂度.在积-代价模型中,连接查询的代价与参与连接的各关系的尺寸之积成正比,即采用嵌套循环方式计算连接.比如,连接查询 $q=r\bowtie s$,则连接查询的代价是 $m+n$,其中 m 是 r 包含元组的数量,n 是 s 包含元组的数量.在和-代价模型中,连接代价和“参与连接的各关系的尺寸与连接操作输出关系的尺寸之和”成正比,即采用哈希函数或排序的方式计算连接.比如,连接查询 $q=r\bowtie s$,则连接查询的代价是 $m+n+p$,其中 m 是 r 包含元组的数量,n 是 s 包含元组的数量, p 是连接操作输出结果中包含元组的数量.

在计算查询代价时,文献[2]较早采用了线性代价模型,以后的很多研究(比如文献[10,13,25,26])也都沿用了该模型.在线性代价模型中,查询 q 的代价是用来回答查询 q 的视图所包含的元组总数目[2],在计算代价时不考虑索引的存在.虽然在实际应用中,一个查询可能只访问 1 个视图的 1 个元组,或者访问一部分元组,或者也可能访问全部元组,但是,为了简化问题,绝大多数研究都假设查询访问视图的全部元组.另外,大部分研究在估算查询代价时都没有考虑查询中所包含的操作的类型,只有少部分研究(比如文献[27])对查询中不同的操作类型采用不同的代价计算方法,比如,对于包含选择操作的查询和包含连接操作的查询,分别采用不同的代价估算方法.

1.3.2 视图维护代价

视图维护是指当基本事实表的内容发生变化时,需要对预先计算存储的实视图集合进行更新.对于数据仓

库应用,视图维护过程一般都是自动进行的,并且对用户而言是透明的.在视图维护期间,停止对外提供服务,当视图维护过程结束时,OLAP 查询系统就可以立即获得最新的视图.

视图维护可以通过两种方式来实现,即增量更新和重新计算[1].采用哪种方式取决于需要更新的数据量,在多数情况下,增量更新的代价比重新计算代价要小.当增量更新无法实现时,也必须采用重新计算的方法.目前有不少研究(比如文献[28–30])提出了快速重新计算视图的方法,也有很多工作(比如文献[31,32])对增量更新进行了深入研究.

1.3.3 存储代价

存储代价是在选择实视图时需要考虑的一个非常关键的问题.视图的尺寸通常影响到某个视图能否被视图选择算法选中.比如,在选择某个视图进行实化时,可能会造成查询时间的少量增加,但同时却带来存储空间的大量减小,那么视图选择算法就应该选择该视图.因此,对于视图选择问题而言,视图尺寸(存储代价)的估计至关重要.视图尺寸估算的准确程度甚至还会直接影响到视图选择的复杂度[33].

目前,文献[34]是视图尺寸估算方面比较有代表性的研究.该文献的作者 Shukla 在视图研究方面积累了丰富的经验,曾经提出了 PBS(pick by size)视图选择算法[13].Shukla 等人在文献[34]中提出的针对多维数据立方体中视图的尺寸估算方法包括:基于采样的方法、基于数学估算的方法和基于概率统计的方法,后来的一些视图尺寸估算方面的研究(比如文献[35])也都采用了类似的方法.从研究人员的大量实验结果来看,基于概率统计的方法其准确度比另外两种方法要高.基于采样的方法通常都假设数据在多维数据立方体中是均匀分布的[36],如果数据分布发生扭曲,一般来说,视图尺寸会有所减小.文献[35]中提出的 PSE(proportional skew effect)方法考虑了数据分布扭曲时的情形,该方法的准确性与其效率成反比,也就是说,采样的样本越大,PSE 的准确性就越好,但是效率也越低.另外,有的研究使用原本用于估算多重集(multiset)尺寸的线性时间概率统计方法[37]来对视图尺寸进行估算,该方法在数据分布扭曲时能够取得很好的效果.在实际应用中,高估视图尺寸带来的影响不大,但是,低估视图尺寸有时会带来问题.Shukla 等人在文献[34]中提出的对数概率统计方法通常会低估视图的尺寸[10].

1.4 约束条件

在视图选择问题中,约束条件 T 可以是存储空间(用来存储实视图的空间)的限制,也可以是视图维护窗口的限制,或者是二者的结合[38].大部分研究都考虑了存储空间的限制,但也有些研究(比如文献[11,39])认为,随着单位磁盘空间价格的不断下降,视图选择问题的主要限制因素将不再是存储空间限制,而是视图维护时间的限制.通常,数据仓库的维护工作都被安排在夜间或者周末进行.在维护期间,数据仓库不对外提供服务[1].维护工作主要包括对数据仓库中的实视图进行更新,因而,实视图的数量越多,实视图更新的时间就越长,但是,可用更新时间最终要受到视图维护窗口大小的限制[40].因此,视图维护窗口是数据仓库设计的关键问题,也是影响实视图选择问题的核心因素.

目前,已有不少视图选择算法把视图维护代价最小化作为视图选择的约束条件.比如,文献[3,15]考虑了视图维护代价,并且在保证满足实视图空间限制的前提下,最小化查询代价和视图维护代价.在文献[41]中,作者把数据仓库配置问题定义成状态空间的优化问题,在这种优化问题中,在保证实视图集合可以回答所有查询的前提下,尽量使视图维护代价最小;但是,文献[41]没有考虑实视图空间限制.文献[23]也针对 OR 图和 AND-OR 图的情形,分别给出了 Inverted-tree 贪婪算法和 A*启发算法来解决这个难题.文献[39]则提出了针对视图维护时间限制的实视图选择问题的两种有效的启发算法:2-阶段算法和集成算法.

虽然,考虑视图维护时间限制的实视图选择问题和考虑实视图空间限制的实视图选择问题有很多相似之处,但是,它们之间仍然存在很大的区别.在考虑空间限制时,优化算法的目标是利用有限的空间选择一定数量的实视图,从而优化查询总体性能,当被选中的实视图的数量逐渐增加时,这些实视图所占用的空间也逐渐增大,因而优化算法具有单调性(monotonicity)[42].但是,在考虑视图维护时间限制时可能就不存在这种单调性.比如,当实视图数量增加时,有可能会使总的视图维护时间增加,但也有可能导致总的视图维护时间的减少.这种非单调性使得考虑实视图空间限制的实视图选择问题在设计优化算法时更加复杂.

1.5 候选视图

与候选视图相关的内容包括视图的筛选、视图的粒度、视图的组织和视图的索引等,下面我们分别予以论述.

1.5.1 视图筛选

候选视图的筛选是影响视图选择算法可扩展性的一个关键因素.已有的许多视图选择方法(比如文献[2−4])都直接假设候选视图已经存在,然后采用适当的方式对候选视图进行组织,最后采用各种视图选择算法从候选视图中选择一部分视图进行实化.因为有这样一个假设,所以当存在大量查询时,就会有大量的候选视图,这些方法的可扩展性也就比较差.

针对这个问题,已有一些相关的研究.其中一种比较典型的方法就是通过视图合并来减少候选视图的数量.视图合并时有两个重要的问题,一个是判断对哪些视图进行合并以及如何合并,另一个就是如何描述一个将要被合并的视图空间.视图与索引不同,视图是一种可能包含 select,group-by 和 aggregation 等操作的多表结构,因此,视图合并算法与索引合并算法有很大的区别.在实际应用中,可能有大量的视图需要合并,尤其对于复杂的查询负载而言更是如此.因此,必须找到一种有效的方法,减小可以用来合并的视图的搜索空间.文献[43]提出了视图树的方法,有效地解决了这两个问题.

减少候选视图数量的另一种典型方法就是尽可能地把公共子表达式作为候选视图.这些公共子表达式代表了回答某个查询的一部分计算工作.当公共子表达式被确定以后,它们只需被计算一次,计算得到的结果可以被多个查询共享,这对于减少候选视图数量和节省查询处理时间来说是非常重要的.但是,探寻公共子表达式并不是一个容易的问题.另外,需要根据公共子表达式对原来的查询进行重写.文献[38,44]深入研究了有效计算公共子表达式的方法,提出使用两个查询之间的最近公共衍生 CCD(closest common derivator)来代表两个查询之间的最大公共性,从而缩小视图搜索空间,提高视图选择算法的性能.

此外,文献[45]提出采用 Union-view 和 Partial-view 作为候选视图,其中,Union-view 是一个包含了某个查询的视图,也就是说,一个 Union-view 不必与其他关系连接就可以直接用来回答某个查询;而 Partial-view 则代表了查询的中间结果,必须与其他关系连接以后才可以回答某个查询.

1.5.2 视图粒度

实视图集合中的视图的粒度会影响到查询的性能.如果粒度太细,一方面,固定的实视图空间内可以容纳更多的属于部分聚集视图的多维数据碎片,但这同时也意味着,在实视图集合中寻找可以用来回答某个查询的视图需要耗费更多的时间;另一方面,相对于粗粒度视图而言,细粒度视图能够回答的查询的数量很有限.我们这里举一个实例进行说明.

例 1:假设多维数据模式包含两个维,即 *Product* 和 *Location*,每个维又具有各自的层次结构.*Product* 维的层次结构是:*ProductId*→*Category*→*All*;*Location* 维的层次结构是:*StoreId*→*Area*→*All*.实视图集合中存储了包括 p_1,p_2,p_3 在内的多个多维数据碎片,其中,p_1={(*ProductId*,{1,1000}),(*StoreId*,{125})},p_2={(*ProductId*,⟨1,1000⟩),(*StoreId*,{225})},p_3={(*ProductId*,{50}),(*StoreId*,{50})},假设查询为 q={(*ProductId*,{50}),(*StoreId*,⟨1,250⟩)}.如图 1 所示,即使使用 p_1,p_2,p_3 这 3 个视图的组合,也无法得到查询 q 所需要的全部数据.

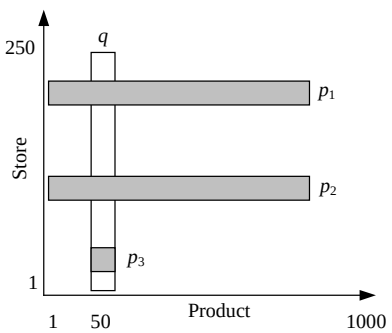


Fig.1 Answering a query by materialized views

图 1 使用实视图回答查询

不同的研究方法采用的视图粒度也不大相同.一些研究(比如文献[2,12−14,19])把多维数据格中的数据节点视图作为视图选择的基本单位,这类方法采用的视图粒度相对比较大.还有些研究(比如文献[18,25,46])把查询作为候选对象,这些研究大多属于视图动态选择方法的范畴.相对于前面一类方法而言,这类方法采用的视图粒度更小,在某些情形下可以取得更好的选择效果,但

是也提高了视图选择问题的复杂性.因为视图粒度越小,可能的候选视图数量就越多,而视图选择问题的复杂性是随着候选视图数量的增加而呈指数增加的.因此,在采用这种粒度的候选视图时,为了提高视图选择性能,往往需要采用上面介绍过的视图合并技术来减少候选视图数量.另外,文献[11]中把一个数据立方体(data cube)看成是多个方体(cubeid)的集合,这些方体被进一步划分成多个碎片(fragment),最后从这些碎片中选择视图进行实化.Deshpande 等人于 1998 年在文献[7]中提出了一种基于块文件的方法来解决查询缓存问题,它把多维查询均匀地分成多个块,并对这些块进行缓存.

1.5.3 视图组织

视图组织是指按照某种方式对候选视图有效地进行组织,为视图选择算法奠定基础.不同的研究提出了不同的视图组织方式,由此也产生了不同的视图选择方法.总体来看,比较有代表性的视图组织方式包括多维数据格、AND-OR 图、MVPP(multiple view processing plan)和视图簇等.

多维数据格是由 Harinarayan 等人于 1996 年在文献[2]中提出来的概念,它根据视图之间的依赖关系对相关视图以“格”的形式进行组织,采用这种视图组织方法的研究为数不少,比如文献[2,12−15,47,48].MVPP 是 Yang 等人于 1997 年在文献[4]中提出的一种组织和表示查询的方法,它实际上就是一个关于多个查询的全局最优查询执行计划,采用这种视图组织方法的研究包括[4,5,49−52].Gupta 于 1997 年在文献[3]中提出了 AND-OR 图的概念,并在以后的相关研究中得到了应用,比如文献[3,27,53−56].视图簇(view cluster)的方法与以上这些方法不大相同,该方法根据视图之间的相似性对视图进行分类,视图被划分到不同的簇,采用这类视图组织方法的研究包括文献[18,26,57]等.

另外,针对一些特殊的应用环境,视图组织方式则是与环境紧密结合的.比如,文献[58]中就提出了一种针对空间数据仓库的 3D 视图依赖框架.文献[59]则针对 XML(extensible markup language)数据流环境设计了一种视图组织框架.文献[47]对多维数据格进行了扩展,融合了分布式语义,可以对分布式数据仓库环境下的视图进行有效的组织.

1.5.4 视图索引

理论上,实视图和索引都是可以显著提高查询性能的物理结构.一个有效的物理数据库设计工具必须能够同时考虑二者之间的交互从而设计出有效的物理方案.如果忽略二者之间的关系,将影响到数据仓库的质量.虽然视图和索引二者比较相似,但是,视图的结构要比索引复杂得多,因为视图可能定义在多个表上,可能会用到 GROUP BY 函数.视图的这种复杂结构使得视图选择要比索引选择更加复杂.正因为这个原因,大部分视图选择研究都不考虑索引(比如文献[2,25]),或者对二者分离进行考虑.文献[12]考虑了视图选择以及针对视图的索引选择的交互,文献[46]则进一步考虑了视图选择和针对基本表的索引选择的交互.

1.6 应用环境

视图选择问题最开始产生于操作型的关系数据库,用于提高一些简单查询的效率.后来,随着数据仓库的普及和应用,视图选择开始应用于数据仓库的配置和优化,以提高数据仓库的总体查询性能;关于视图选择问题的大多数研究都属于这一类,比如文献[2,4,5,10,13,49].另外,很多研究(比如文献[6−9,60,61])也研究了针对缓存的视图选择来提高未来到达查询的响应速度,这类视图选择问题通常也可以称为查询缓存问题.除此之外,一些研究人员也努力尝试把视图选择应用于其他一些比较特殊的环境中,比如 XML 数据库[62,63]、分布式数据仓库[47,64]、空间数据仓库[58]、数据流[59]、实时主动数据仓库[14]等等.

从近几年出现的一些研究来看,把视图选择问题与某个具体的应用环境相结合,设计为该环境“量体裁衣”的视图选择方法已经逐渐成为一种新的研究思路.基于特定的环境寻找视图选择问题的解决方案,可以比一些通用的解决方案更加有效,比如文献[14]充分利用了实时主动数据仓库的分析规则使用 CUBE 的规律,实现了更加合理、有效的视图选择.

2 视图选择方法

视图选择方法的分类可以有多种标准,比如:(1) 确定候选视图的方式;(2) 用来组织候选视图的框架;

(3) 所选用的代价模型;(4) 调用查询优化器的方式;(5) 视图选择的环境是关系型还是多维数据模型;(6) 解决方案属于技术性的还是理论性的[18];(7) 查询计划是否事先确定[65].本文根据视图维护的频率,将视图选择方法分成 3 大类:静态选择方法、动态选择方法和混合方法.

根据所考虑的问题的不同,实视图可以分为长期性的和短期性的[38].长期性实视图用于数据仓库的设计,存储在数据仓库中,用于提高数据仓库总体查询性能;而短期性实视图则用于中间层,存储在缓存中,用于提高以后到达查询的性能.在数据仓库的实施中,可以采用长期性实视图,也可以采用短期性实视图,或者二者相互结合.由此,分别产生了静态、动态和混合视图选择方法.

2.1 静态视图选择方法

在静态视图选择方法中,系统会在维护时间窗口内,根据查询的统计数据,把那些比较频繁发生的查询进行实化,从而提高以后到达的查询的响应速度,在下一个维护时间窗口到来之前,这些实视图不发生变化[6].目前已经有许多关于静态视图选择的研究,其中比较典型的几种方法是:基于多维数据格、基于 MVPP、基于 AND-OR 图和基于随机算法的方法.

下面的内容,我们先归类介绍各种静态视图选择方法,然后讨论算法性能和复杂性,最后给出静态视图选择方法的缺点.

2.1.1 基于多维数据格的方法

定义 5(多维数据格)[2]. 多维数据格$\langle M,\leq\rangle$是由多个数据节点 T 构成的,其中,$T=(a_1,a_2,\ldots,a_n)$,M 是多个节点 T 的集合,每个 a_i 代表第 i 维上的一个级别,并且有:

(1) 对于两个节点 $T_1=(a_1,a_2,\ldots,a_n)$和 $T_2=(b_1,b_2,\ldots,b_n)$,$T_1\leq T_2$(即 T_1 依赖于 T_2),当且仅当对于所有的 i 都有 $a_i\leq b_i$,即 T_2 在各维上的级别均低于或等于 T_1 在相应维上的级别,并且利用 T_2 可以计算得到 T_1;

(2) 格中基本元表示为 $DB=(c_1,c_2,\ldots,c_n)$,其中,c_i 表示第 i 维上最低的一个级别;通常假定 DB 是已知的,从而可以根据 DB 计算格中各个其他节点的数据.

例 2:多维数据模式包含两个维,即 *Product* 和 *Location*,每个维又具有各自的层次结构.*Product* 维的层次结构是:*ProductId*→*Category*→*All*;*Location* 维的层次结构是:*StoreId*→*Area*→*All*.该多维数据格如图 2 所示,其中 *p*,*c*,*a*,*s* 分别表示 *ProductId*,*Category*,*StoreId*,*Area*,根据多维数据格的定义,基本元 $DB=(p,s)$.这种特殊的多维数据格通常被称为超立方体(hypercube)[2].

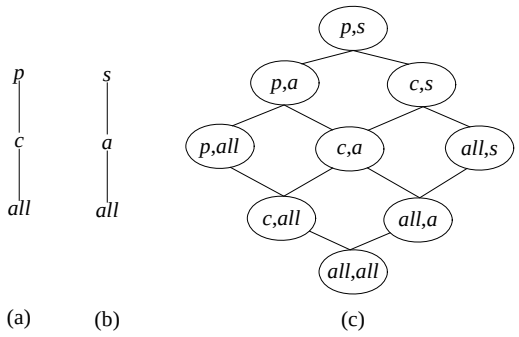


Fig.2 An example of lattice

图 2 一个多维数据格的例子

Harinarayan 等人于 1996 年在文献[2]中首先提出了多维数据格的概念(见定义 5),并给出了基于多维数据格的视图选择贪婪算法 BPUS(benefit per unit space),证明了该算法可以取得和其他多项式时间算法一样好的性能.在视图选择时,文献[2]只考虑了查询代价,而没有考虑视图维护代价,在进行代价计算时,采用的是线性代价模型,即把查询所涉及关系的元组数目作为查询代价.在不考虑空间限制时,该贪婪算法与最优方法性能之比不小于 $1-(1-1/k)^k$,其中,k 表示实视图的数量,在 k 取值足够大的情况下,贪婪算法的性能至少是最优方法的性能的 0.63 倍.在考虑空间限制时,贪婪算法的性能至少是最优方法的性能的 $0.63-f$ 倍,其中,f 表示被尺寸最大的视图所消耗的实视图空间的比例.

虽然 BPUS 比穷尽搜索要快,但是,文献[13]的作者 Shukla 等人通过大量实验发现,BPUS 算法往往需要花掉几天甚至几个月的时间来生成数据库管理员使用的实视图空间-查询响应时间曲线.因此,Shukla 等人在文献[13]中提出了一种基于多维数据格的、简单快速的 PBS 算法来选择实视图.PBS 的运行速度要比 BPUS 快几个数量级,可以很快生成实视图空间-查询响应时间曲线.PBS 继承了 BPUS 的线性代价模型,与 BPUS 不同,PBS

采用了单位空间收益为视图选择标准.当考虑实视图空间限制时,在最坏情况下,PBS 算法的性能至少是最优方法性能的 0.63–f 倍.

文献[12]扩展了文献[2]的工作,在视图选择时同时考虑了视图和索引.但是,与文献[2,13]一样,文献[12]也没有考虑视图维护代价.文献[15]中的方法同时考虑了视图维护代价和查询代价,并提出了 MD-格的概念.与文献[2]提出的多维数据格不同,MD-格同时考虑了事实表和维表属性层次结构,而文献[2]中的多维数据格只包含了事实表.通过实验,文献[15]的作者发现,与 MD-格中包含大量节点相比,具有代表性的查询的数量相对较少.这种相对稀疏性使得在最小化查询代价时,只需考虑一小部分满足一定条件的相关视图,即格中的视图 v 需要与具有代表性的某个查询 q 相关,并且存在两个候选视图 v_j 和 v_k,v 是 v_j 和 v_k 的最近父亲节点.根据相关候选视图建立 MD-格,可以大大缩小格尺寸.同时,文献[15]利用尺寸估计方法来进一步缩小格空间.根据尺寸估计,当候选视图的聚集粒度太高时,就可以抛弃掉,因为与选择更高层次的视图比起来,选择这些视图带来的总体查询时间的减少量不会太大.文献[47]扩展了多维数据格的概念,并把贪婪算法应用到分布式环境下,在这种环境下,查询和数据可以在不同的分布式节点间自由移动.文献[19]的研究发现,虽然文献[2]证明了 BPUS 算法可以保证取得一个性能下限,但是,当实视图空间 S 较小时,下限可能很小,甚至为负数,这就使得贪婪算法的性能无法得到很好的保障,因此,文献[19]提出了更新的 A*算法来改善这种情况下的视图选择算法的性能.文献[66]提出了一种 PGA(polynomial greedy algorithm)算法.该算法在数据仓库的维数量增加的同时,仍然能够保持较好的可扩展性,并且与之前的基于多维数据格的方法相比有明显的性能改进.

但是,基于多维数据格的方法也有一些局限性[48],首先,视图之间的依赖是单向的,即子视图依赖于父视图;其次,存在冗余信息,因为不仅要存储基本表,还要存储视图,多维数据格就很好地解释了这种信息冗余性,因为格中的所有视图都依赖于基本表.文献[48]提出了一种新方法来克服这些局限性.在该方法中,数据立方体(data cube)被分解成许多个视图元素.在对数据立方体进行分解时,视图元素具有比视图依赖层次结构(view dependency hierarchy)更细的粒度.对于每个聚集操作,文献[48]中设计了一对局部分解操作,这些操作具备完整性、非膨胀性、分布性和隔离性这些性质.这些局部分解操作可以被层叠在一起来计算数据立方体中的聚集视图.完整性使得子视图可以从父视图中组合得到.非膨胀性使得视图元素全集的生成不会增加数据立方体的尺寸.此外,该文献中还设计了一个视图元素图来对视图元素进行管理.视图元素图可以帮助对视图元素集的相对收益进行评估,从而可以快速地检索得到特定的查询.

2.1.2 基于 MVPP(multiple view processing plan)的方法

定义 6(MVPP)[4]. 一个 MVPP 是一个带标记的图 $M=(V,A,C_a^q,C_m^r,f_q,f_u)$, 其中 V 是顶点集合,A 是有向边的集合,并满足以下条件:

- 对于每个基本关系、每个不同的查询,以及一个查询树中的每个关系代数操作,都创建一个与之对应的顶点.
- 对于 $v\in V$,$T(v)$表示顶点 v 产生的关系.$T(v)$可以是基本关系,也可以是查询过程产生的中间结果或者查询的最终结果.
- 对于任何叶子节点 v(不存在以 v 为终点的有向边),$T(v)$表示一个基本关系,用“□”表示;假设 L 是叶子节点集合,则对于任何 $v\in L$,$f_u(v)$表示 v 的更新频率.
- 对于任何根节点 v(不存在以 v 为起点的有向边),$T(v)$表示一个查询,用“●”表示;假设 R 是根节点集合,则对于任何 $v\in R$,$f_q(v)$表示 v 的查询频率.
- 如果一个节点 u 对应的基本关系或中间结果关系 $T(u)$需要在节点 v 作进一步处理,则画一条从 u 到 v 的边,即 $u\rightarrow v$.
- 对于任何顶点 v,令 $S(v)$表示所有以 v 为终点的边的起点的集合.对于任何 $v\in L$,$S(v)=\varnothing$.令 $S^*\{v\}=S(v)\cup\{\bigcup_{v'\in S(v)}S^*\{v'\}\}$ 表示 v 的所有后代.
- 对于任何顶点 v,令 $D(v)$表示所有以 v 为起点的边的终点的集合.对于任何 $v\in R$,$D(v)=\varnothing$.令 $D^*\{v\}=D(v)\cup\{\bigcup_{v'\in D(v)}D^*\{v'\}\}$ 表示 v 的所有祖先.

- 对于每个 $v\in V$, $C_a^q(v)$ 表示查询 q 访问 $T(v)$的代价; $C_m^r(v)$ 表示在 $T(v)$被实化时,当 $S^*(v)\cap R$ 发生变化时,维护 $T(v)$的代价.

MVPP 是 Yang 等人于 1997 年在文献[4]中提出来的一种组织和表示查询的方法.一个 MVPP(如图 3 所示)实际上是一个关于多个查询的全局最优查询执行计划,它是对多个局部查询计划进行合并以后得到的.在 MVPP 的基础上,文献[4]提出了一个用来描述分布式环境下视图选择的框架,从而使问题的表述更加正式;并给出了实视图设计的代价模型,同时考虑了查询代价和视图维护代价.此外,该文献中还给出了如何选择实视图的算法以及两种生成 MVPP 的算法,其中一种方法可以产生比较迅速、可行的解决方案,而另一种方法则产生最优解决方案,它把最优 MVPP 生成问题转换成 0-1 编码问题,这种用整数编码来建模视图选择问题的方法也在以后的研究(比如文献[20])中得到了采用.

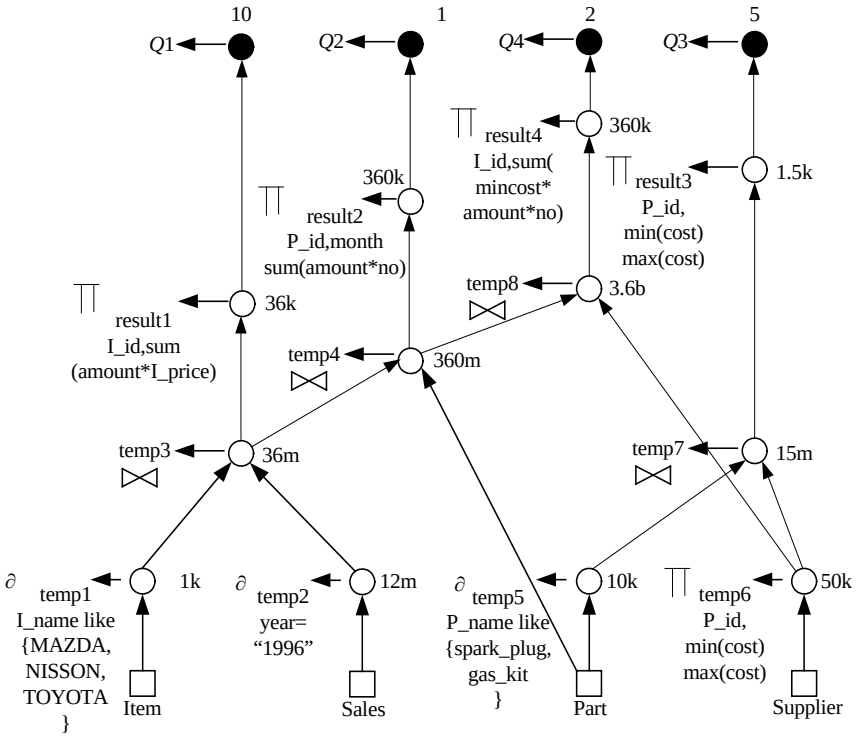


Fig.3 An example of MVPP

图 3 一个 MVPP 的例子

但是,文献[4]中的方法重点关注的是在最小化查询代价时的多查询优化,虽然也考虑视图维护代价,但却忽略了视图维护过程的优化问题,只是把重新计算作为视图更新策略.因此,为了弥补这个不足,文献[50]引入了视图维护策略,同时考虑了多查询优化和视图维护过程优化问题.

MVPP 已被证明是一种较好的查询表示方法,并在以后的很多研究中得到了应用.比如,在以后出现的利用遗传算法解决视图选择问题的研究(比如文献[5,49,51,52])中,MVPP 被用来组织和表示查询,从而很容易将视图集合转换成 0-1 字符串,作为遗传算法的输入.

2.1.3 基于 AND-OR 图的方法

定义 7(AO-DAG)[3]. 一个视图(或查询)V 的 AO-DAG,是一个有向无环图.该图以 V 作为根节点,以基本关系作为叶子节点,每个非叶子节点都有 1 个或多个 AND 弧与之关联,每个 AND 弧捆绑以 V 为起点的多条边,并且每条 AND 弧都有与之关联的代价和操作数.

定义 8(AND-OR 图[3]). 一个图 G 被称为视图(或查询)$V_1,V_2,\ldots,V_k$ 的 AND-OR 图,如果对于每一个 V_i,在 G

中都存在一个子图 G_i,并且 G_i 是一个关于 V_i 的 AO-DAG.在 AND-OR 图中的每个节点 u 附带以下参数:f_u,节点 u 的访问频率;S_u,节点 u 占用的空间;g_u,节点 u 的更新频率.

Gupta 于 1997 年在文献[3]中提出了 AND-OR 图的概念,并给出了构建 AND-OR 图(如图 4 所示)的方法.文献[3]对于多种不同情形分别给出了不同的视图选择算法.对于不考虑更新代价的 AND 图,分别给出了 Greedy 算法和 Greedy-interchange 算法;对于考虑更新代价的 AND 图,重点讨论了实视图集合 M 中的各个视图的更新频率小于查询频率时的情形;对于考虑索引的 AND 图,则采用了 Inner-level Greedy 算法;对于不考虑更新的 OR 图,采用 Greedy-interchange 算法;对于不考虑更新代价的 AND-OR 图,分别给出了 AO-Greedy 算法和 Muli-level Greedy 算法.

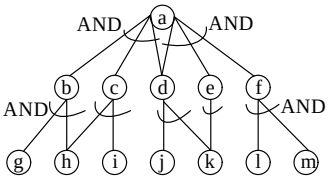


Fig.4 An example of AND-OR graph

图 4 一个 AND-OR 图的例子

但是,文献[3]中没有考虑选择、投影、连接等情况.另外,文献[3]对视图选择给出的形式化描述也忽略了一个事实,即查询需要被局部地回答,也即只能使用实视图来回答查询.以后发表的文献[23,54]又对基于 AND-OR 图的视图选择方法作了进一步的完善.文献[23]提出了考虑视图维护时间限制的实视图选择问题理论框架,用 AND-OR 图对视图进行建模,并提出了考虑视图维护代价的 Inverted-tree Greedy 算法和 A*算法.但是,文献[39]通过实验证实,文献[23]中针对 OR 图的启发算法的性能并没有朴素算法的性能好,并给出了一个实例进行验证.同时,文献[22]通过大量实验发现,文献[23]中的 A*算法并不是总能找到最优解.

AND-OR 图在以后的其他研究中也得到了应用.文献[55]采用 OR 图表示查询,然后把 OR 图转换成 0-1 字符串作为遗传算法的输入.文献[67]提出了基于 AND-OR 图的由视图相关性驱动的视图选择方法.文献[68]中为一个虚拟的公司 R 设计了一个企业数据仓库,为了加快决策支持查询的速度,采用 AND-OR 图对视图进行组织,并采用了文献[3]中的贪婪算法来选择实视图.但是,文献[18]指出,上述绝大多数方法都只能停留在理论层面上,在实际应用中,可扩展性较差.文献[53]采用 AND-OR 图组织视图,并提出了一种与代价模型无关的视图贪婪选择算法.文献[27]在 AND-OR 图的基础上提出了一种称为 MVMG(multi view materialization graph)的视图组织形式;MVMG 是双向的有向无环图,每个 MVMG 包含两种类型的节点,即 AND 节点和 OR 节点;MVMG 中的每个节点都对应一个候选视图.一个 MVMG 是由多个查询对应的有向无环图合并得到的,合并的规则是,一旦两个查询存在公共子表达式,就把二者的有向无环图的相应部分加以合并.

2.1.4 基于随机优化的方法

视图选择问题是一个 NP-完全问题[2].对于这类问题,一个比较可行的解决方案就是随机优化算法.随机算法以统计学理论为基础,使用评估函数来指导搜索进程不断向最优解逼近.随机算法可以通过牺牲解的质量来获得更快的执行时间,从而在较短的时间内找到合理的解.虽然用这种方法只能得到近似最优解,但是,与执行时间减少的程度相比,解的质量下降的程度还是比较小的.通常,随机算法得到的解非常逼近最优解,相似程度可以达到 90%以上[55].

遗传算法是随机算法的一种形式,它分成 3 个过程:(1) 确定一个初始参数集合;(2) 从这些初始参数的后代中选择较好的方案;(3) 使用概率转移规则来搜索问题空间.采用遗传算法来解决视图选择问题基于的原理是:数据仓库中存在大量的视图和查询,并且是不断变化的.因此,视图选择问题就可以看成是一个具有大量状态空间的复杂问题.当采用遗传算法来解决视图选择问题时,需要考虑两个问题[5]:一个是表示方法问题,即如何把视图选择问题表示成遗传算法可以理解的内容;遗传算法只接受 0-1 编码的字符串,但是视图选择问题通常表示成 DAG(directed acyclic graph),因此,DAG 不能直接作为遗传算法的输入.另一个问题是,遗传算法中的 0-1 编码字符串只要发生一点小变化,就会使 DAG 产生非常大的变动,有时这种变动会产生无效的结果,因此就需要采取有效的方法剔除无效的结果.

文献[5]最先提出用遗传算法解决视图选择问题,以后的许多相关研究(比如文献[16,49,51,52,55,56,69])都

与文献[5]大同小异.在表示方法上,大部分都用 MVPP[4]来组织和表示大量的查询(比如文献[5,49,51,52]),然后把 MVPP 转换成 0-1 字符串;转换的方法是,当 MVPP 中的节点(视图)已被实化时,用 1 表示,否则用 0 表示.也有少部分方法采用 OR 图[3]来表示查询(比如文献[55,56]),然后把 OR 图转换成 0-1 字符串.但是,在遗传算法中,不可行解的存在是一个很大的问题.在文献[5,52]中,没有包含惩罚函数来抑制不可行解的数量.文献[55]引入惩罚函数解决了这个问题,但是这篇文章的实验只是在很小规模的数据集上进行的,只包含了 8 个视图和 8 个维度.另外,遗传算法的相关研究显示,在处理不可行解问题时,修复模式比惩罚函数的效果要好.因此,文献[56]引入了修复模式来解决遗传算法在视图选择过程中产生的不可行解问题.总的来说,从方法的完整性、性能评估和质量的角度来看,最好的进化方法还是文献[51]中提出的方法.

文献[55]的研究指出,对于视图选择问题,遗传算法本身并不保证能够获得好的近似最优解,该方法的效果好坏取决于很多因素,比如正确的问题定义、算法的设置以及繁琐的算法参数调整.

与采用遗传算法解决视图选择问题类似,文献[70]提出采用模拟退火算法来解决视图选择问题,模拟退火算法可以找到全局最优解,并且只需搜索一部分状态空间.但是,利用模拟算法进行视图选择的效果取决于选择正确的参数,需要通过多次实际测试得到最好的效果.

2.1.5 基于框架的方法

与其他方法不同,基于框架的方法本身并不是针对视图选择问题提出具体的新算法,而是通过设计一种视图选择框架来提高其他视图选择算法的性能,其他算法可以嵌套在框架中发挥作用.这方面比较典型的研究如文献[65].

文献[65]提出了一个称为 ANNE 的视图选择框架.在 ANNE 中,被激发的算法的复杂度取决于它所处理的数据格的复杂性.ANNE 避免被触发的算法处理过于复杂的数据格,所以 ANNE 的总体复杂度就受到了有效的限制.一般来说,贪婪算法的时间复杂度是 $O(m2^{2d})$,其中 d 是维的数量,m 表示选中的视图数量.贪婪算法的空间复杂度是 $O(2^d)$.ANNE 则有更好的性能,ANNE 的时间复杂度是 $O(dS^3)$,其中 d 是维的数量,S 表示空间限制;ANNE 的空间复杂度是 $O(S^2)$.ANNE 比其他非框架方法要好,因为:(1) 以前算法的运行时间都会随着维数的增加呈指数级增长;ANNE 反复调用嵌入算法,降低了嵌入算法每次需要处理的视图的数量,因此,具有更低的线性时间复杂度.(2) ANNE 对维的重要性进行排序,先处理重要性比较大的维,从而可以获得更好的解.

2.1.6 其他方法

文献[41]使用“多查询图”表示视图,并在此基础上把视图选择问题建模成状态空间搜索问题.每个状态包括实视图集合的多查询图和为该实视图集合而重新书写的查询.从一个状态转移到另一个状态,会导致多查询图的改变,从而需要相应地重新书写查询,使其建立在新的实视图集合上.文献[41]使用穷尽算法来搜索操作代价最小的状态,并使用启发算法来缩小搜索空间.

但是,文献[41,46]都是建立在两个假设之上:(1) 视图选择只需考虑那些属于多个查询的公共子表达式的视图;(2) 视图选择结果中包含的视图数量有相对较小的上限.然而,文献[33]的研究发现,这两个假设并不是在所有条件下都成立.

文献[11]把一个数据立方体(data cube)看成是多个方体(cubeid)的集合,这些方体被进一步划分成多个碎片(fragment),最后从这些碎片中选择视图进行实化.为了回答用户的查询,通常需要用到多个实视图,为此,该文献中设计了启发算法来选择合适的碎片进行实化.

文献[39]提出了针对视图维护时间限制的实视图选择问题的两种有效的启发算法:2-阶段算法和集成算法.在 2-阶段算法中,第 1 个阶段根据最小化总体查询响应时间的原则选择得到初始视图集合,第 2 个阶段继续选择满足视图维护时间限制的实视图集合.在该方法中,还把考虑视图维护时间限制的实视图选择问题转化成最小权重最大基数匹配问题,后者是图论中被深入研究的一个问题,已有不少好的求解方法.集成算法则同时考虑查询响应时间和视图维护时间的限制,因此,它的性能比 2-阶段算法及其变种都要好些.此外,该文献还讨论了 2-阶段算法的几种改进方法.但是,集成算法比 2-阶段算法要消耗更多的时间来求解.文献[39]指出,两种启发算法可以在多项式时间内找到可行的解;但是,正如文献[21]所指出的那样,文献[39]并没有对提出的启发算法进

行定量分析,也没有给出任何实验结果.另外,在用最小权重最大基数匹配问题来模拟视图选择问题时,某个视图不能与自身进行匹配.

文献[71]提出了临时实视图和永久实视图的概念,并提出有效的方法来选择那些可以被临时实视图共享的表达式和索引,并为每个永久实视图选择额外的表达式和索引,从而在很大程度上改善了视图维护时间.

文献[72]则提出了最近实化父视图(NMPV(nearest materialized parent view))的概念,并用 B^+树创建对聚集视图的索引,还给出一个以相对效益为标准的 PVMA(progressive view materialization algorithm)算法.该算法的时间复杂度为 $O(V^2+I+D+U)$,其中,V 表示视图的数量,I 表示各种类型的查询中属于插入查询的数量,D 表示删除查询的数量,U 表示选择查询的数量.

2.1.7 算法性能和复杂性

视图选择问题是一个 NP 完全问题[2,73],也就是说,该问题很难找到最优解,因为随着视图数量的增加,解空间呈指数级增长.对于这类问题,除非 P=NP,否则没有确定性的多项式时间算法可以得到比贪婪算法更好的性能.文献[2]提出的基于偏序多维数据格的贪婪算法提供了一种获得近似最优解的途径.在不考虑实视图空间限制时,该贪婪算法与最优方法性能之比不小于 $1-(1-1/k)^k$,其中,k 表示实视图的数量.在 k 取值足够大的情况下,贪婪算法的性能至少是最优方法性能的 0.63 倍.在这之后提出的类似方法(比如文献[3,12,23])也都得到了类似的性能比.在计算性能比时,文献[2]采用的标准是收益比而不是响应时间比,而文献[23]的研究发现,在采用时间比时,当视图总数量 n 接近无穷大时(不考虑 P 和 NP 问题),文献[2]算法得到的性能比至少是 $n/12$.文献[42]的研究同时也表明,当采用偏序多维数据格时,如果 P≠NP,那么视图选择问题的任何一个多项式时间近似算法得到的性能比至少是 $n^{1-\varepsilon}(\varepsilon>0)$.这意味着,当采用偏序多维数据格时,如果 P≠NP,那么视图选择问题不存在性能好的近似算法,即使实视图的数量增加一个常量倍数,仍对解决问题没有帮助.因此,文献[42]建议未来的理论研究应该集中在特殊情形,比如超立方体格(hypercube lattice)和积图(product graph)[2],从而降低问题的难度.但是,对于超立方体格的情形,目前的研究仍未能找到性能比下限,也没有出现可以保证性能比的算法.

对于考虑视图维护代价的视图选择问题,目前的研究对于能够在多项式时间内找到近似最优解这个问题仍然存在分歧.文献[23]认为,由于考虑视图维护代价而带来的非单调性问题,使得基于单位维护代价的查询收益的视图选择贪婪算法会产生任意坏的解;而文献[39]则认为,利用启发算法可以在多项式时间内找到可行的解.

2.1.8 静态视图选择方法的缺点

实视图选择的静态方法确实改善了数据仓库的总体查询性能,但是,它与决策支持分析的动态特性不相符.尤其对于即席查询,专家为了在数据仓库中找到一些感兴趣的趋势,他们提出的查询通常是难以预测的.另外,数据仓库中的数据和查询的特征都是随着时间而变化的,静态选择方法得到的结果可能很快就过期了.这就意味着,管理员需要监督查询的模式,周期性地运行这些静态视图选择算法,从而对实视图集合进行调整,满足用户的需求.在大型数据仓库中,通常存在大量具有不同访问特征的用户,这使得管理员的监督工作量非常巨大,甚至是不可行的.

静态选择方法的另一个缺点是,系统没有办法改变一个错误的选择结果,更无法利用那些不能被实视图集合回答的查询的中间结果.在实际应用中,虽然 OLAP 查询需要大量的 I/O 开销和 CPU 处理开销,但是,它们的输出结果通常都很小,比如“找出过去 10 年的总销售额”.处理这种查询可能需要耗费几个小时的大量的磁盘扫描和聚集计算,但是,计算得到的结果却只需要 8 个字节的存储空间,很容易被以后的其他查询引用.而且,在上卷(roll-up)操作中,当我们在一个较粗的粒度上访问数据时,未来的查询很可能从以前的结果中计算得到,从而根本不用访问基本表.因此,根据 OLAP 查询之间存在的大量的依赖性,我们可以有效地利用查询中间结果.

2.2 动态视图选择方法

鉴于静态视图选择方法存在的缺点,为了满足用户变化的查询需求,我们有时需要动态的选择方法.目前,研究人员已经提出不少动态选择方法,比如文献[6–9,25,60,61].这些方法所基于的原理与内存管理原理相似,视图可以根据需要被选择,或者通过某种策略被预先实化.当存在空间限制时,可以使用替换算法来对实视图集合

进行调整.

动态方法可以看成是查询负载驱动的视图选择方法,它会对查询负载进行语法分析,从而枚举相关的候选视图.该方法通过调用查询优化器,采用贪婪方法对最相关的视图进行组织.查询负载对于预测未来查询而言是一个很好的参考,因为通常来说,未来的查询与它之前的查询负载在语法上会比较相近.另外,从查询负载中提炼出候选视图,可以大大增加实视图在未来一段时间内被使用的可能性.

比较典型的动态视图选择方法包括基于块文件的方法、DynaMat、基于谓词的方法、基于缓存预测的方法、基于查询分类的方法和其他方法.

2.2.1 基于块文件的方法

传统的缓存方法对整个查询进行缓存,以提高以后查询的响应速度.对于一个新到达的查询,只有它的数据全部包含在缓存中的某个查询中,才可以使用缓存中的数据来回答这个查询.另外,查询级缓存会带来存储的冗余,因为被缓存的查询之间可能存在重叠的数据.

Deshpande 等人于 1998 年在文献[7]中提出了一种基于块文件的方法来解决查询缓存问题.它把多维查询均匀地分成多个块,并对这些块进行缓存.一个查询的结果使用多个块来进行存储,如果一个新到达的查询 q 的边界不在块边界内,那么这个块就不会包含可以用来回答新查询 q 的数据.因为块的粒度比查询的粒度要低,因此,可以用块来回答新到达查询的一部分内容.通过在查询和块编号之间建立映射,就可以确定回答一个新到达查询需要哪些块.然后,把这个块集合分成两个不相交的集合,其中一个集合中的块可以从缓存中得到,而另一个集合中的块则可以由基本表计算得到.替换策略可以根据缓存中各个块被使用的频率来决定替换哪个块.

但是,基于块文件的方法的视图选择和替换策略以及具体实施都比较复杂.

2.2.2 DynaMat

Kotidis 等人于 1999 年在文献[11]中提出了称为 DynaMat 的方法,它用一个统一的框架综合考虑了视图维护和视图选择问题,并采用了新的收益标准,可以在多个粒度层次上动态地实化视图,从而更好地与用户的查询需求相配合.Dynamat 不断监视新到达的查询,并且在保证空间限制的前提下选择最佳的实视图集合.在更新期间,DynaMat 协调当前实视图集合,并且在给定的视图更新窗口内,更新收益最大的实视图子集.Dynamat 方法采用了基于需求的抓取策略,该方法固定保存一定粒度的实化视图,以满足被称为 MRQ(multidimensional range queries)类型的查询需要.一个 MRQ 与一个视图非常相似,并且 MRQ 可以在某个维上只取 1 个值,也就是 OLAP 查询中所谓的“切片”.MRQ 的粒度是在选择许多小视图(具有高度针对性且只能回答少量特定查询)和选择少量大视图(可以用来回答很多查询)二者的折衷.实验结果表明,DynaMat 的性能优于最优的静态视图选择算法,这意味着它优于其他近似最优的静态视图选择算法,比如文献[2,3,15]中提出的方法.

但是,这种方法在进行视图选择之前没有考虑用户的访问模式信息.而且,DyanMat 在视图选择时还对用户查询模式进行了限制,从而只能有效地支持以下 3 种类型的查询[6]:(1) 选择一个维的整个区间;(2) 选择一个维的某个值;(3) 空区间.

2.2.3 基于谓词的方法

Choi 等人于 2003 年在文献[6]中提出了基于谓词的动态视图管理方法——PREDICATE,它可以充分利用关系数据库系统的能力,同时也去掉针对多维数据碎片[11]的各种约束,使得它们可以回答更多的查询.与文献[7,11]的方法不同,该方法根据用户谓词对视图/表进行动态分区.对于实视图动态管理问题,文献[6]重点研究了以下几个问题:(1) 为分区进行谓词选择;(2) 重新分区;(3) 视图替换策略.

PREDICATE 是基于 ROLAP(relational OLAP)的,很容易在关系型 DBMS(database management system)上实现.PREDICATE 与块文件方法[7]不同,块文件方法使用数组来存储数据,而 PREDICATE 使用关系表来支持实视图.关系表的空间消耗要比多维数组小得多.另外,DynaMat 使用一个维对数据进行分区,PREDICATE 使用谓词对数据进行分区,即把一个视图水平划分成多个不相交的子视图.用于划分的谓词是那些出现在 WHERE 从句中的谓词.

PREDICATE 方法的系统实现包含如下 4 个组件:

(1) 查询分析器:对到达系统的查询进行分析,并把一些必要的信息转换成内部数据结构,以供其他组件使用.

(2) 分区引导者:它首先确定位于分区池中的哪些候选分区可以有效地回答查询,然后选择一个最好的候选分区来回答查询.如果不存在候选分区可以回答查询,就用基本表来回答查询.

(3) 视图管理器:监视到达的查询并执行两个任务.第一,确定选择哪个谓词对视图进行分区可以产生最大收益.这项工作需要采用一个代价评估模型来对每个谓词产生的收益进行评估.如果用来分区的谓词发生了变化,那么视图管理器就会进行重新分区.第二,当磁盘空间达到限制水平时,视图管理器采用替换策略替换分区池中的分区.

(4) 分区池:实视图分区的集中存储仓库.

但是,PREDICATE 方法没有给出具体的代价评估方法,也没有详细的视图替换策略.

2.2.4 基于缓存预测的方法

Sapia 于 2000 年在文献[9]中提出的 PROMISE 方法则是基于当前的查询来预测未来查询的结构和尺寸.该方法的原理与缓存预抓取原理相同.缓存预抓取技术已经被广泛地加以研究,但是,不能把这些技术直接应用到 OLAP 环境,因为 OLAP 应用与其他应用有着明显的区别:第一,在 OLAP 应用中要预测的查询的数量比其他应用大得多;第二,不同的缓存对象并不是彼此孤立的,一个新到达的查询可能可以使用缓存中的多个对象得到回答,这就意味着,如果某个查询可以使用缓存中的对象间接回答,那么预抓取这个对象就是有用的.

要想支持缓存预抓取策略,查询负载必须具有可导航性(即连续几个查询之间具有相关性),并且,前后两个查询之间的时间间隔必须大于预抓取某个对象所消耗的时间.文献[9]的作者对实际应用系统中的 260 个会话(共包含 3 150 个查询)进行了两个月的跟踪,得到的分析结果显示,缓存预抓取策略是可行的.

在 OLAP 系统中,如果用户在前一个分析结果的基础上继续得到下一个分析结果,那么 OLAP 查询负载就具有可导航性.这些可导航的查询被称为一个“会话”.文献[9]中的实验分析结果显示,典型的 OLAP 会话都有一定的长度,因此可以很好地支持预测.只有 1%的会话只包含单个查询,并且有一部分会话包含的查询数量达到 100 个以上.如果假设对于包含 5 个以上查询的会话的准确预测是可能的,那么将近有 63.8%的查询满足这个要求.另外,分析结果还显示,对于大部分查询而言,两个查询之间的时间间隔都比较长.如果假设预测和抓取查询要平均耗费 10 秒,就有大约 81%的查询有很长的查询间隔时间可以允许缓存抓取.

通过实验分析验证了缓存预抓取策略对 OLAP 应用的可行性以后,文献[9]引入了马尔可夫模型,分别设计了结构预测模型和基于值的预测模型,以对用户的查询行为进行预测.

但是,PROMISE 方法也存在缺点.在实际应用中,可能的查询数量通常比较大,要预测下一个阶段是哪个查询是非常耗时的.而且,单个实视图的粒度需要足够小,才能捕捉到查询之间细微的差别.然而,文献[8]已经指出,实视图的粒度越细,更新代价就越高.

2.2.5 基于查询分类的方法

基于查询分类的方法充分考虑不同查询类型的特点,并根据查询之间的相似性将其划分到不同的查询类中,在进行视图选择时只需考虑属于同一个查询类内的视图,极大地减小了候选视图的数量,而且不同查询类可以并行进行视图的选择,加快了视图选择的速度.这方面的典型研究包括文献[18,26,57]等等.

文献[26]中提出了采用多用户多窗口方法进行视图的选择.该方法充分考虑了每个用户的查询特点,可以实现更合理的视图选择.在此基础上,文献[57]对多用户多窗口方法作了进一步改进,采用了不同的视图分组策略和视图选择算法,得到了更好的性能.

文献[18]中方法的创新点在于,充分利用了关于如何使用视图回答一个查询集合的知识来对查询进行分类.为了达到对查询进行分类的目的,该文献中定义了查询间的相似性和非相似性,从而可以识别比较相似的查询.这些相似的查询被划分到同一个类中,并用来构建一个候选视图集合.另外,可以对候选视图进行合并,从而满足多个查询的要求.当候选视图数量较多时,这个合并过程可能非常耗费时间,但是,文献[18]中的方法只对那些属于某个类之内的候选视图进行“类内合并”,而不是像其他方法那样,对所有的候选视图都执行合并操作.这

就极大地降低了合并过程的复杂性,因为每个类内的候选视图数量相对于所有的视图数量而言要少得多.

2.2.6 其他方法

文献[46]指出,在研究中发现:(1) 为每个查询找到与之匹配的唯一的实视图是不可行的,因为查询表达式与实视图表达式通常不会一样;(2) 某一部分表集合即使被实化,也只能带来很少的收益.基于以上这两个发现,该文献中提出了两个关键技术来减小视图选择的搜索范围:一个是表集合选择技术,可以选择感兴趣的表集合,从而只需要从这个表集合中选择视图;另一个是视图合并技术,用来确定那些对任何单个查询来说都不可能是最优视图,却可以使多个查询得到优化的候选视图.

文献[25]中的方法则根据系统收集的查询,采用基于多维数据格的、以单位空间频率为视图选择标准的FPUS 算法来进行实视图的选择.然后,系统就可以利用这些实视图来响应新的查询,并继续收集这些查询.随着新查询的增加,系统中查询的分布情况也可能发生变化,使得原有的实视图集合不再适应新的查询分布情况.针对这种情况,该文献给出了具体算法来对实视图集合进行动态调整.

文献[61]提出了一种新方法,使用一些规范的查询来重新表示一批查询,这样,当这批查询执行时,前面查询的结果就可以被后面的查询所使用,从而减小总的执行代价.显然,这种方法需要事先知道系统中所有的查询,所以,虽然这种方法在操作型数据库中能够取得较好的效果,但是在 OLAP 环境下却无法取得较好的效果,因为在 OLAP 环境下,无法准确地知道一批查询的顺序.

动态视图选择方法期望根据变化的用户查询模式来动态地调整实视图集合.当大部分是高度聚集的视图时,动态视图选择能够得到较好的性能.但是,在某些情况下,当同时需要很多低度聚集的视图时,则临时计算聚集视图的响应时间太长.

文献[74]采用另一种方法进行动态视图选择.把 OLAP 系统分成两个操作阶段,启动阶段和在线阶段.在启动阶段,可以根据查询分布概率来选择一些实视图,这个阶段属于经典的静态视图选择.在在线阶段,只考虑正在使用的 OLAP 系统,此时的系统中已经存在一个实视图集合.随着时间的变化,用户的查询需求会发生变化,每种类型的聚集视图的相对重要性相应地也会发生变化,系统需要重新选择新的实视图,从而更好地为新到达的查询服务.但是,对视图进行实化的计算代价较大,用来实化新视图的时间窗口可能不允许实化所有视图.在这种情况下,在进行视图选择时就要丢弃原视图集合中的一部分视图,并把新视图添加进去.在启动阶段,采用随机优化算法(文献[16,49,51,52,55,56,69])选择视图;而在在线阶段,则采用包括随机算法和贪婪算法(文献[2,3])在内的多种视图选择算法.

文献[75]提出了输入是一系列语句时的视图动态选择和删除算法,可以保证限定的空间下最小化执行代价.作者把这个问题建模成有向无环图中的最短路径问题,提供了一种启发算法.该算法把以下两个过程结合在一起:(1) 寻找候选视图集合的过程;(2) 在工作负载语句执行过程中,动态创建和删除实视图的过程.实验结果显示,这种方法与以前的静态和动态方法相比能够得到更好的性能.

另外,传统的实视图选择方法大多会盲目地实化和维护所有的行,包括那些从来不被使用的行.针对这个问题,文献[76]提出了一个更加灵活的视图实化策略,目的在于减少存储空间和视图维护的代价.该方法选择性地实化动态实视图的一部分行,例如,最频繁访问的行.每个视图都会有 1 个或多个控制表与它关联,控制表中定义了当前已被实化的行.实视图的行集合可以动态地变化,可以通过手工的方式实现,也可以通过使用反馈环,采用自动的方式实现.动态查询计划可以决定某个视图在运行时是否有用.Microsoft SQL Server 环境下的实验结果显示,与传统的实视图相比,动态实视图极大地减少了存储需求和维护代价,同时也得到了更好的查询性能,并提高了缓冲池的使用效率.

2.3 混合视图选择方法

静态和动态视图选择方法有其各自的特点,因此,文献[10]提出了混合视图选择方法,实现了静态选择方法和动态选择方法的有效结合,既能充分利用静态选择方法改善查询响应时间的作用,又能发挥动态方法自动视图调整的功能.混合视图选择方法需要解决如下几个问题:(1) 如何区分静态视图集合和动态视图集合;(2) 如何对静态视图集合和动态视图集合进行实视图的选择;(3) 如何更新实视图.

混合方法的主要思想是,把视图集合划分为动态视图集合和静态视图集合.从动态视图集合中选出的实视图可以即时生成或被替换,而从静态视图集合中选出的实视图可以保留好几个视图维护窗口.可以采取预先实化算法来根据用户查询模式对一些高度聚集视图进行预先实化处理.静态视图集合中的视图则用来更好地维护动态视图集合中的视图,并回答一些粒度较细的查询.由于动态视图集合中的视图是不断变化的,所以,静态实视图选择策略需要同时考虑实化和非实化的动态视图集合的维护.混合方法满足单调性原则,从而可以使用贪婪算法.

3 总结与研究展望

实视图的选择是数据仓库研究领域的一个热点问题,经过众多研究人员 10 多年的努力,目前已经有不少相关的研究成果发表.在本文中,我们介绍了视图选择问题,讨论了与视图选择问题相关的内容,如代价模型、收益函数、代价计算、约束条件和视图索引.我们还对视图选择方法进行了归类介绍,并讨论了每类方法的代表性研究成果.视图选择问题是一个 NP 问题,从我们对视图选择算法性能和复杂性的讨论中可以看出,该问题很难找到最优解,因为随着视图数量的增加,解空间呈指数级增长.有些方法虽然可以降低选择的复杂性,但却仍然需要增加一些额外的限制,比如采用特殊的代价模型.

通观现有的实视图选择问题的相关研究,我们认为,以下几个方面是该问题的未来研究方向:

(1) 与人工智能领域知识的紧密结合.视图选择问题是一个 NP-完全问题,可以利用随机算法得到非常逼近最优解的近似解.遗传算法和模拟退火算法都是人工智能领域的经典研究成果,被引入实视图选择问题的研究以后,为该问题提供了更多的可选解决方案,比如文献[16,55],相信视图选择问题的研究今后将会融入更多的人工智能领域的知识.但是,遗传算法和模拟退火算法本身并不保证能够获得好的近似最优解,这些方法的效果好坏取决于很多因素.因此,未来这方面的研究应该重点着眼于以下几个方面:正确的问题定义、算法的设置以及合理而完善的算法参数调整机制.

(2) 与实际应用环境紧密结合的视图选择方法研究.目前的绝大多数视图选择方法属于可应用于各种数据仓库的通用方法,虽然提高了数据仓库的总体性能,但是,并没有有效利用不同类型数据仓库的具体特点.实际上,如果能够在选择算法的设计上考虑不同数据仓库的特点,就可以得到更好的查询性能改善,比如,文献[14]的研究就提出了为实时主动数据仓库“量体裁衣”的视图选择算法,取得了较好的效果.这方面研究的难点在于根据应用环境(比如分布式数据仓库、空间数据仓库、XML 数据库和数据流等)的特点,找出现有视图选择算法在具体应用环境中的不足,进而提出相应的解决方案.以实时主动数据仓库的视图选择为例,为了更好地利用分析规则的 CUBE 使用模式,就必须完善主动决策引擎机制,使其具备分析规则执行日志记录功能,然后,需要利用合适的数据挖掘算法从日志中挖掘得到 CUBE 使用模式,最后,利用 CUBE 模式指导视图选择.

(3) 有效结合动态方法与静态方法的混合方法的研究.单一的静态方法或单一的动态方法都有其各自的优点和局限性.静态方法可以提高系统整体查询性能,但却无法适应动态变化的查询.动态方法可以提高视图选择算法适应性,但是,过多的视图调整也会加大系统的开销.混合方法可以综合利用静态方法和动态方法的优点,并尽量避免二者的缺点,可以取得比单一的静态或动态方法更好的效果.但是,目前,混合方法的研究比较少,相信今后会有更多的这方面研究成果出现.这方面研究的重点包括静态视图集合和动态视图集合的划分机制以及动态视图的更新机制等.

(4) 实化视图中一部分行的研究.传统的实视图盲目地实化和维护所有的行,即使那些从来不被使用的行,这不仅降低了视图选择总体效率,也浪费了宝贵的系统资源.如果采用某种机制选择性地实化动态实视图的一部分行,就可以很好地改进性能,同时也减小了系统开销,比如文献[76]就提出了一个这样的视图实化策略.这方面的研究重点包括选择被实化视图行的标准、视图行统计特征的变化跟踪(比如,在某个时间段,某些行是被频繁使用的,但是在另一些时间段内则不然),以及视图行统计特征发生变化后的实视图行替换机制等.

(5) 面向查询负载的视图选择研究.以前的方法大都没有考虑查询负载的具体特点,比如,静态方法只是简单地实化使用频率高的视图,而后来的一些动态方法虽然是面向查询负载的,但是并没有对查询负载的特性进

行有效分析,不能充分利用特定查询负载的特点进一步改进算法性能.近年来发表的研究开始注意到利用查询负载特性的重要性,比如文献[77,78].这方面研究的重点在于如何发现查询负载的特性,以及如何有效利用查询负载特性来设计高效的视图选择算法等.

References:

[1] Inmon WH. Building the data warehouse. 4th ed., New York: Wiley, 2005.

[2] Harinarayan V, Rajaraman A, Ullman JD. Implementing data cubes efficiently. In: Jagadish HV, Mumick IS, eds. Proc. of the 1996 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD'96). Montreal: ACM Press, 1996. 205−216.

[3] Gupta H. Selection of views to materialize in a data warehouse. In: Afrati FN, Kolaitis PG, eds. Proc. of the 6th Int'l Conf. on Database Theory (ICDT'97). Delphi: Springer-Verlag, 1997. 98−112.

[4] Yang J, Karlapalem K, Li Q. Algorithm for materialized view design in data warehousing environment. In: Jarke M, Carey MJ, Dittrich KR, eds. Proc. of the 23rd Int'l Conf. on Very Large Data Bases (VLDB'97). Athens: Morgan Kaufmann Publishers, 1997. 136−145.

[5] Zhang C, Yang J. Genetic algorithm for materialized view selection in data warehouse environments. In: Mohania MK, Tjoa AM, eds. Proc. of the 8th Int'l Conf. on Data Warehousing and Knowledge Discovery (DaWaK'99). Florence: Springer-Verlag, 1999. 116−125.

[6] Choi CH, Yu JX, Lu HJ. Dynamic materialized view management based on predicates. In: Zhou XF, Zhang YC, Orlowska ME, eds. Proc. of the 5th Asia-Pacific Web Conf. on Web Technologies and Applications (APWeb 2003). Xi'an: Springer-Verlag, 2003. 583−594.

[7] Deshpande P, Ramaswamy K, Shukla A, Naughton J. Caching multidimensional queries using chunks. In: Haas LM, Tiwary A, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD'98). Seattle: ACM Press, 1998. 259−270.

[8] Kotidis Y, Roussopoulos N. A case for dynamic view management. ACM Trans. on Database Systems, 2001,26(4):388−423.

[9] Sapia C. PROMISE: Predicting query behavior to enable predictive caching strategies for OLAP systems. In: Kambayashi Y, Mohania MK, Tjoa AM, eds. Proc. of the 2nd Int'l Conf. on Data Warehousing and Knowledge Discovery (DaWaK 2000). London: Springer-Verlag, 2000. 224−233.

[10] Shah B, Ramachandran K, Raghavan V, Gupta H. A hybrid approach for data warehouse view selection. Int'l Journal of Data Warehousing and Mining, 2006,2(2):1−37.

[11] Kotidis Y, Roussopoulos N. DynaMat: A dynamic view management system for data warehouses. In: Delis A, Faloutsos C, Ghandeharizadeh S, eds. Proc. of the 1999 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD'99). New York: ACM Press, 1999. 371−382.

[12] Gupta H, Harinarayan V, Rajaraman A, Ullman JD. Index selection for OLAP. In: Gray A, Larson PA, eds. Proc. of the 13th Int'l Conf. on Data Engineering (ICDE'97). Birmingham: IEEE Computer Society Press, 1997. 208−219.

[13] Shukla A, Deshpande P, Naughton JF. Materialized view selection for multidimensional datasets. In: Gupta A, Shmueli O, Widom J, eds. Proc. of the 24th Int'l Conf. on Very Large Data Bases (VLDB'98). New York: Morgan Kaufmann Publishers, 1998. 488−499.

[14] Lin ZY, Yang DQ, Song GJ, Wang TJ, Tang SW. Materialized views selection of multi-dimensional data in real-time active data warehouses. Journal of Software, 2008,19(2):301−313 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/301.htm

[15] Baralis E, Paraboschi S, Teniente E. Materialized view selection in a multidimensional database. In: Jarke M, Carey MJ, Dittrich KR, eds. Proc. of the 23rd Int'l Conf. on Very Large Data Bases (VLDB'97). Athens: Morgan Kaufmann Publishers, 1997. 156−165.

[16] Lawrence M. Multiobjective genetic algorithms for materialized view selection in OLAP data warehouses. In: Cattolico M, eds. Proc. of the 2006 Genetic and Evolutionary Computation Conf. (GECCO 2006). Seattle: ACM Press, 2006. 699−706.

[17] Afrati F, Chirkova R, Gergatsoulis M, Pavlaki V. View selection for real conjunctive queries. Acta Informatica, 2007,44(5):289−321.

[18] Aouiche K, Jouve PE, Darmont J. Clustering-Based materialized view selection in data warehouses. In: Manolopoulos Y, Pokorný J, Sellis TK, eds. Proc. of the 10th East European Conf. on Advances in Databases and Information Systems (ADBIS 2006). Thessaloniki: Springer-Verlag, 2006. 81−95.

[19] Gou G, Yu JX, Lu HJ. A* Search: An efficient and flexible approach to materialized view selection. IEEE Trans. on Systems, Man, and Cybernetics—Part C: Applications and Reviews, 2006,36(3):411−425.

[20] Li JN, Talebi ZA, Chirkova R, Fathi Y. A formal model for the problem of view selection for aggregate queries. In: Eder J, Haav HM, Kalja A, Penjam J, eds. Proc. of the 9th East European Conf. on Advances in Databases and Information Systems (ADBIS 2005). Tallinn: Springer-Verlag, 2005. 125−138.

[21] Choi CH, Yu XJ, Gou G. What difference heuristics make: maintenance-cost view-selection revisited. In: Meng XF, Su JW, Wang YJ, eds. Proc. of the 3rd Int'l Conf. on Web-Age Information Management (WAIM 2002). Beijing: Springer-Verlag, 2002. 247−258.

[22] Yu JX, Choi CH, Gou G, Lu HJ. Selecting views with maintenance cost constraints: issues heuristics and performance. Journal of Research and Practice in Information Technology, 2004,36(2):89−110.

[23] Gupta H, Mumick I. Selection of views to materialize under a maintenance cost constraint. In: Beeri C, Buneman P, eds. Proc. of the Int'l Conf. on. Database Theory (ICDT'99). Jerusalem: Springer-Verlag, 1999. 453−470.

[24] Silberschatz A, Korth HF, Sudarshan S. Database System Concepts. 4th ed., Columbus: McGraw-Hill Companies, Inc., 2002.

[25] Tan HX, Zhou LX. Dynamic selection of materialized views of multi-dimensional data. Journal of Software, 2002,13(6): 1090−1096 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/1090.htm

[26] Xue YS, Lin ZY, Duan JJ, Lv XH, Zhang W. Dynamic selection of materialized views of multi-dimensional data with multi-users and multi-windows method. Journal of Computer Research and Development, 2004,41(10):1703−1711 (in Chinese with English abstract).

[27] Baril X, Bellahsene Z. Selection of materialized views: A cost-based approach. In: Eder J, Missikoff M, eds. Proc. of the 15th Int'l Conf. on Advanced Information Systems Engineering (CaiSE 2003). Klagenfurt: Springer-Verlag, 2003. 665−680.

[28] Agrawal A, Agrawal R, Deshpande P, Gupta A, Naughton J, Ramakrishnan R, Sarawagi S. On the computation of multidimensional aggregates. In: Vijayaraman TM, Buchmann AP, Mohan C, Sarda NL, eds. Proc. of the 22nd Int'l Conf. on Very Large Data Bases (VLDB'96). Mumbai: Morgan Kaufmann Publishers, 1996. 506−521.

[29] Ross KA, Srivastava D. Fast computation of sparse data cubes. In: Jarke M, Carey MJ, Dittrich KR, Lochovsky FH, Loucopoulos P, Jeusfeld MA, eds. Proc. of the 23rd Int'l Conf. on Very Large Data Bases (VLDB'97). Athens: Morgan Kaufmann Publishers, 1997. 116−125.

[30] Zhao YH, Deshpande P, Naughton JF. An array-based algorithm for simultaneous multidimensional aggregates. In: Peckham J, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD'97). Tucson: ACM Press, 1997. 159−170.

[31] Liu B, Chen ST, Rundensteiner EA. Batch data warehouse maintenance in dynamic environments. In: Proc. of the 2002 ACM CIKM Int'l Conf. on Information and Knowledge Management (CIKM 2002). McLean: ACM Press, 2002. 68−75.

[32] Lee KY, Son JH, Kim MH. Efficient incremental view maintenance in data warehouses. In: Proc. of the 2001 ACM CIKM Int'l Conf. on Information and Knowledge Management (CIKM 2001). Atlanta: ACM Press, 2001. 349−356.

[33] Chirkova R, Halevy A, Suciu D. A formal perspective on the view selection problem. In: Apers PM, Atzeni P, Ceri S, Paraboschi S, Ramamohanarao K, Snodgrass RT, eds. Proc. of the 27th Int'l Conf. on Very Large Data Bases (VLDB 2001). Roma: Morgan Kaufmann Publishers, 2001. 59−68.

[34] Shukla A, Deshpande PM, Naughton JF, Ramasamy K. Storage estimation for multidimensional aggregates in the presence of hierarchies. In: Vijayaraman TM, Buchmann AP, Mohan C, Sarda NL, eds. Proc. the 22th Int'l Conf. on Very Large Data Bases (VLDB'96). Mumbai: Morgan Kaufmann Publishers, 1996. 522−531.

[35] Runapongsa K, Nadeau TP, Teorey TJ. Storage estimation for multidimensional aggregates in OLAP. In: MacKay SA, Johnson JH, eds. Proc. of the 10th Conf. of the Centre for Advanced Studies on Collaborative Research (CASCON'99). Toronto: ACM Press, 1999. 40−54.

[36] Hass PJ, Naughton JF, Seshadri S, Stokes L. Sampling-Based estimation of the number of distinct values of an attribute. In: Dayal U, Gray PM, Nishio S, eds. Proc. of the 21st Int'l Conf. on Very Large Data Bases (VLDB'95). Zürich: Morgan Kaufmann Publishers, 1995. 311−322.

[37] Whang KY, Vander-Zanden BT, Taylor HM. A linear-time probabilistic counting algorithm for database applications. ACM Trans. on Database Systems, 1990,15(2):208−229.

[38] Xu WG, Theodoratos D, Zuzarte C. Computing closest common subexpressions for view selection problems. In: Song IY, Vassiliadis P, eds. Proc. of the 9th ACM Int'l Workshop on Data Warehousing and OLAP (DOLAP 2006). Arlington: ACM Press, 2006. 75−82.

[39] Liang WF, Wang H, Orlowska ME. Materialized view selection under the maintenance time constraint. Data and Knowledge Engineering, 2001,37(2):203−216.

[40] Do L, Drew P, Jin W, Junami V, Rossum DV. Issues in developing very large data warehouses. In: Gupta A, Shmueli O, Widom J, eds. Proc. of the 24th Int'l Conf. on Very Large Data Bases (VLDB'98). New York: Morgan Kaufmann Publishers, 1998. 633−636.

[41] Theodorators D, Sellis T. Data warehouse configuration. In: Jarke M, Carey MJ, Dittrich KR, Lochovsky FH, Loucopoulos P, Jeusfeld MA, eds. Proc. of the 23rd Int'l Conf. on Very Large Data Bases (VLDB'97). Athens: Morgan Kaufmann Publishers, 1997. 126−135.

[42] Karloff H, Mihail M. On the complexity of the view-selection problem. In: Proc. of the 18th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (PODS'99). Philadelphia: ACM Press, 1999. 167−173.

[43] Chen CQ, Feng YC, Feng JL. View merging in the context of view selection. In: Nascimento MA, Özsu MT, Zaïane OR, eds. Proc. of the Int'l Database Engineering & Applications Symp. (IDEAS 2002). Edmonton: IEEE Computer Society, 2002. 33−43.

[44] Theodoratos D, Xu WG. Constructing search spaces for materialized view selection. In: Song IY, Davis KC, eds. Proc. of the 7th ACM Int'l Workshop on Data Warehousing and OLAP (DOLAP 2004). Washington: ACM Press, 2004. 112−121.

[45] Lee TH, Chang JY, Lee SG. Using relational database constraints to design materialized views in data warehouses. In: Yu JX, Lin XM, Lu HJ, Zhang YC, eds. Proc. of the 6th Asia-Pacific Web Conf. on Advanced Web Technologies and Applications (APWeb 2004). Hangzhou: Springer-Verlag, 2004. 395−404.

[46] Agrawal S, Chaudhuri S, Narasayya V. Automated selection of materialized views and indexes in SQL databases. In: Abbadi AE, Brodie ML, Chakravarthy S, Dayal U, Kamel N, Schlageter G, Whang KY, eds. Proc. of the 26th Int'l Conf. on Very Large Data Bases (VLDB 2000). Cairo: Morgan Kaufmann Publishers, 2000. 496−505.

[47] Bauer A, Lehner W. On solving the view selection problem in distributed data warehouse architectures. In: Proc. of the 15th Int'l Conf. on Scientific and Statistical Database Management (SSDBM 2003). Cambridge: IEEE Computer Society, 2003. 43−51.

[48] Smith JR, Li CS, Castelli V, Jhingran A. Dynamic assembly of views in data cubes. In: Proc. of the 17th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (PODS'98). Seattle: ACM Press, 1998. 74−283.

[49] Horng JT, Chang YJ, Liu BJ. Applying evolutionary algorithms to materialized view selection in a data warehouse. Soft Computing—A Fusion of Foundations, Methodologies and Applications, 2003,7(8):574−581.

[50] Yousri NA, Ahmed KM, El-Makky NM. Algorithms for selecting materialized views in a data warehouse. In: Proc. of the 3rd ACS/IEEE Int'l Conf. on Digital Object Identifier (AICCSA 2005). Cario: IEEE Computer Society, 2005. 27−35.

[51] Yu JX. Yao X, Choi CH, Gou G. Materialized view selection as constraint evolutionary optimization. IEEE Trans. on Systems, Man and Cybernetics—Part C, 2003,33(4):485−467.

[52] Zhang C, Yao X, Yang J. An evolutionary approach to materialized view selection in a data warehouse environment. IEEE Trans. on Systems, Man and Cybernetics—Part C, 2001,31(3):282−293.

[53] Dhote CA, Ali MS. Materialized view selection in data warehousing. In: Proc. of the 4th Int'l Conf. on Information Technology (ITNG 2007). Las Vegas: IEEE Computer Society, 2007. 843−847.

[54] Gupta H, Mumick IS. Selection of views to materialize in a data warehouse. IEEE Trans. on Knowledge and Data Engineering, 2005,17(1):24−43.

[55] Lee M, Hammer J. Speeding up materialized view selection in data warehouses using a randomized algorithm. Int'l Journal of Cooperative Information Systems, 2001,10(3):327−353.

[56] Wang ZQ, Zhang DX. Optimal genetic view selection algorithm under space constraint. Int'l Journal of Information Technology, 2005,11(5):44−51.

[57] Lin ZY, Yang DQ, Song GJ, Wang TJ. User-Oriented materialized view selection. In: Miyazaki T, Paik I, Wei DM, eds. Proc. of IEEE the 7th Int'l Conf. on Computer and Information Technology (CIT 2007). Fukushima: IEEE Computer Society, 2007. 133−138.

[58] Yu SM. Hierarchical metadata driven view selection for spatial query evaluations. In: Fan WF, Wu ZH, Yang J, eds. Proc. of the 6th Int'l Conf. on Web-Age Information Management (WAIM 2005). Hangzhou: Springer-Verlag, 2005. 791−797.

[59] Gupta AK, Halevy AY, Suciu D. View selection for stream processing. In: Fernandez MF, Papakonstantinou Y, eds. Proc. of the 5th Int'l Workshop on the Web and Databases (WebDB 2002). Madison: ACM, 2002. 83−88.

[60] Shim J, Scheuermann P, Vingralek R. Dynamic caching of query results for decision support systems. In: Özsoyoglu ZM, Özsoyoglu G, Hou WC, eds. Proc. of the 11th Int'l Conf. on Scientific and Statistical Database Management (SSDBM'99). Cleveland: IEEE Computer Society, 1999. 254−263.

[61] Yao OS, An AJ. Using user access patterns for semantic query caching. In: Marík V, Retschitzegger W, Stepánková O, eds. Proc. of the 14th Int'l Conf. on Database and Expert System Applications (DEXA 2003). Prague: Springer-Verlag, 2003. 737−746.

[62] Gupta AK, Suciu D, Halevy AY. The view selection problem for XML content based routing. In: Proc. of the 22nd ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database Systems (PODS 2003). San Diego: ACM Press, 2003. 68−77.

[63] Mandhani B, Suciu D. Query caching and view selection for XML databases. In: Böhm K, Jensen CS, Haas LM, Kersten ML, Larson PA, Ooi BC, eds. Proc. of the 31st Int'l Conf. on Very Large Data Bases (VLDB 2005). Trondheim: ACM Press, 2005. 469−480.

[64] Ye W, Gu N, Yang GX, Liu ZY. Extended derivation cube based view materialization selection in distributed data warehouse. In: Fan WF, Wu ZH, Yang J, eds. Proc. of the 6th Int'l Conf. on Web-Age Information Management (WAIM 2005). Hangzhou: Springer-Verlag, 2005. 245−256.

[65] Yan Y, Wang P, Wang C, Zhou HF, Wang W, Shi BL. ANNE: An efficient framework on view selection problem. In: Yu JX, Lin XM, Lu HJ, Zhang YC, eds. Proc. of the 6th Asia-Pacific Web Conf. on Advanced Web Technologies and Applications (APWeb 2004). Hangzhou: Springer-Verlag, 2004. 384−394.

[66] Nadeau TP, Teorey TJ. Achieving scalability in OLAP materialized view selection. In: Theodoratos D, eds. Proc. of ACM 5th Int'l Workshop on Data Warehousing and OLAP (DOLAP 2002). McLean: ACM Press, 2002. 28−34.

[67] Valluri SR, Vadapalli S, Karlapalem K. View relevance driven materialized view selection in data warehousing environment. Australian Computer Science Communications, 2002,24(2):187−196.

[68] Chan GK, Li Q, Feng L. Design and selection of materialized views in a data warehousing environment: A case study. In: Proc. of the 2nd ACM Int'l Workshop on Data warehousing and OLAP (DOLAP'99). Kansas: ACM, 1999. 42−47.

[69] Lin WY, Kuo IC. A genetic selection algorithm for OLAP data cubes. Knowledge and Information Systems, 2004,6(1):83−102.

[70] Derakhshan R, Dehne F, Korn O, Stantic B. Simulated annealing for materialized view selection in data warehousing environment. In: Hamza MH, eds. Proc. of the 24th IASTED Int'l Conf. on Database and Applications. Innsbruck: IASTED/ACTA Press, 2006. 89−94.

[71] Mistry H, Roy P, Sudarshan S, Ramamritham K. Materialized view selection and maintenance using multi-query optimization. In: Aref WG, eds. Proc. of the 2001 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2001). Santa Barbara: ACM Press, 2001. 307−318.

[72] Uchiyama H, Runapongsa K, Teorey TJ. A progressive view materialization algorithm. In: Proc. of the 2nd ACM Int'l Workshop on Data Warehousing and OLAP (DOLAP'99). Kansas, 1999. 36−41.

[73] Cook SA. The complexity of theorem proving procedure. In: Proc. of the 3rd Annual ACM Symp. on Theory of Computing (STOC'71). Shaker Heights: ACM, 1971. 151−158.

[74] Lawrence M, Rau-Chaplin A. Dynamic view selection for OLAP. In: Tjoa AM, Trujillo J, eds. Proc. of the 8th Int'l Conf. on Data Warehousing and Knowledge Discovery (DaWaK 2006). Krakow: Springer-Verlag, 2006. 33−44.

[75] Xu WG, Theodoratos D, Zuzarte C, Wu XY, Oria V. A dynamic view materialization scheme for sequences of query and update statements. In: Song IY, Eder J, Nguyen TM, eds. Proc. of the 9th Int'l Conf. on Data Warehousing and Knowledge Discovery (DaWaK 2007). Regensburg: Springer-Verlag, 2007. 55–65.

[76] Zhou JR, Larson PA, Goldstein J, Ding LP. Dynamic materialized views. In: Proc. of the 23th Int'l Conf. on Data Engineering (ICDE 2007). Istanbul: IEEE Computer Society, 2007. 526–535.

[77] Agrawal S, Chu E, Narasayya VR. Automatic physical design tuning: Workload as a sequence. In: Chaudhuri S, Hristidis V, Polyzotis N, eds. Proc. of the Int'l Conf. on Management of Data (SIGMOD 2006). Chicago: ACM, 2006. 683–694.

[78] Phan T, Li WS. Dynamic materialization of query views for data warehouse workloads. In: Proc. of the 24th Int'l Conf. on Data Engineering (ICDE 2008). Cancún: IEEE Computer Society, 2008. 436–445.

附中文参考文献:

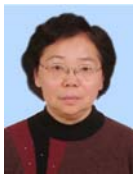
[14] 林子雨,杨冬青,宋国杰,王腾蛟,唐世渭.实时主动数据仓库中多维数据实视图的选择.软件学报,2008,19(2):301−313. http://www.jos.org.cn/1000-9825/19/301.htm

[25] 谭红星,周龙骧.多维数据实视图的动态选择.软件学报,2002,13(6):1090−1096. http://www.jos.org.cn/1000-9825/13/1090.htm

[26] 薛永生,林子雨,段江娇,吕晓华.用多用户多窗口处理多维视图动态选择.计算机研究与发展,2004,41(10):1703−1711.

林子雨(1978－),男,吉林柳河人,博士生,CCF 学生会员,主要研究领域为数据库,实时主动数据仓库,数据挖掘.

王腾蛟(1973－),男,博士,副教授,CCF 高级会员,主要研究领域为数据库,数据仓库,Web 数据集成,数据挖掘.



杨冬青(1945－),女,教授,博士生导师,CCF 高级会员,主要研究领域为数据库,数据仓库,Web 数据集成,移动数据挖掘.

宋国杰(1975－),男,博士,副教授,主要研究领域为数据仓库,数据挖掘.