

# ROLAP 环境中改进的视图物化和索引选择方法

Improved Method of View Materialization and Index Selection in ROLAP

林琳<sup>1,2</sup> 王新军<sup>1</sup>

(山东大学计算机科学与技术学院 济南 250061)<sup>1</sup> (61938 部队 北京 100089)<sup>2</sup>

**Abstract** Using summary-view to accelerate OLAP queries is one of the most common methods used in ROLAP systems. However, high storage and computation cost make this method very difficult to be implemented in the actual environment. Among various issues associated with this, index selection and view materialization are two of the top common methods. In this paper, we discuss how to select and build index on subsets of the primary keys, and how to implement view materialization according to these indexes. Based on Nested Relation Approach, we propose an improved view materialization method—Nested Table Approach. In this method, we use Oracle denormalization to store this nested relation in Nested Table Type. In addition, we make some suggestions on Dominate Prime (DPrime) Index Set Filter.

**Keywords** Index selection, View materialization, Dprime, Nested table

## 1 引言

在 OLAP 系统中,多维模型和概要视图是最常用的查询优化方法。但是,随着属性维数的增长,概要视图和联合索引的存储开销与计算开销将呈爆炸性增长<sup>[3]</sup>。因此,在现实环境中实现这些方法非常困难。在本文中,我们将在视图组织及为视图建立索引方面进行深入研究。

概要视图上的索引对于 OLAP 系统的查询优化是十分重要的,然而它们也是数据库中开销最大的部分。选择正确的索引对 OLAP 系统的设计来说是至关重要的。实践证明在 OLAP 环境中我们有可能仅仅通过在主键上找到一个子集并以此建立索引,就能取得满意的执行效果。以此为基础,我们认为应该在主键的子集上建立索引并筛选出那些索引选择性不佳的索引。

另外,我们认为应该采用嵌套关系来进行视图物化,因为通过该方法可以使拥有非唯一索引的概要视图最优化。通过将几个相关的元组归结为一个超级元组,我们可以将一些相关的数据打包到更少的且邻近的磁盘块,这样 I/O 便可更为有效地运行。而且,由于我们是在元组群的基础上创建索引的,因此可以大幅减小索引空间。此外,还可以降低概要视图的空间需求。在文章中,我们采用 Oracle Nested Table 来实现这个方法。

在第 2 节中,我们将讨论背景与相关工作。在第 3 节中,我们介绍一种索引选择方法——优选属性法,以及一种以此为基础的视图物化方法——嵌套关系法。在第 4 节,将对优选属性法和嵌套关系

法提出改进意见。最后得出了结论。

## 2 背景及相关工作

通过运用概要视图,我们可将 OLAP 查询过程分解为若干个步骤,并预先以高代价执行集合操作。在 Oracle 8i 中通过使用物化视图的方法,可以对 OLAP 查询进行“查询重写”从而获得满意的查询性能<sup>[2]</sup>。关于概要视图方法,最严重的问题之一就是空间激增,即当更多的概要视图和索引被添加到 OLAP 系统时,空间需求会爆炸式地增加。为解决这个问题,有人曾经建议使用两种方法来筛选出概要视图<sup>[6]</sup>,但对于 OLAP 系统都不是很有效。

由于“空间激增”问题的产生,人们开始关注如何挑选视图及索引。以对给定查询的作用为依据,有多种视图及索引的选择方法。最经常被提到的方法之一是文[5]中所阐述的方法,即将贪心算法作为处理视图中索引的一种“简便”的近似算法。文[5]中宣称该方法的运行效率至少达到最优算法的 53%。另外,在文[5]中还指出,对于 n 维数据方体,其可能的索引规模估计将达到约  $3n!$ 。因此,需要一种索引过滤方法筛选出那些索引选择性不佳的索引。常见的索引过滤方法按主键属性的不同次序进行考虑,以此作为索引的候选集。这样,在大多数情况下,候选集的数量将无法满足索引过滤的实际要求,因此无法在实际应用中使用。针对索引选择和过滤问题,文[7]曾提出了 Dominant Prime (DPrime) Index Set Filter 方法。该方法使得依靠关键字子集来建立索引并提供满意的查询性能成为可能。

如何实现视图是 OLAP 领域里的又一个热点话题。为了提高 OLAP 性能,可以使用一种特殊技术,它是关系型数据库的一种非主流技术。该技术的功能异常强大,除能够满足查询的性能要求外,还可以使查询所用到的表及索引大大减少,它就是维的逆规范化处理。

### 3 优选属性法和嵌套关系法简介

文[7]对索引选择、索引过滤和视图物化的方法进行了深入研究。该文主要论述了在主键子集上选择和创建索引的方法——优选属性法,以及以此为基础进行视图物化的方法——嵌套关系法。下面将详细介绍这两种方法,并给出一个使用范例。

首先给出几个定义:

**定义 1(索引选择性)** 是选择索引的标准。在 OLTP 体系中,一般用与每个索引实例相关联的元组平均所占的“百分比值”来定义索引的选择性<sup>[1]</sup>。低“百分比值”的索引可以节省大量可以预见的 I/O 操作。在文[8]中,曾建议索引选择性的百分比值应低于 2%~4%。我们将这种索引选择性称为“百分比选择性”。在文[7]中,将“索引选择性”重新定义为与每个索引实例相关联的元组的平均数量。

**定义 2(优选属性 Dominant Prime (DPrime))**

假设有一个概要视图  $SV(V)$ ,  $V$  是形成概要视图的 group-by 属性集,  $X$  是主键的一个子集。如果对于根据属性集  $X$  所创建的索引,其索引选择性的值( $size(SV(V))/size(SV(X))$ )小于一个预先给定的阈值,那么我们就将属性集  $X$  定义为概要视图  $SV(V)$  的 DPrime。  $Size(SV(V))$  表示  $SV(V)$  中的元组数。

**定义 3(主优选属性 Primary Dprime)** 在多维模型中,  $X$  是  $SV(V)$  的一个 DPrime, 如果对于  $SV(V)$  不存在另一个 DPrime  $Y$ , 使  $Y$  属于  $X$ , 我们便将  $X$  定义为  $SV(V)$  的 Primary DPrime。有时, 一个概要视图会有不止一个 Primary DPrime。这种情况下, 我们需要考虑在这些 Primary DPrime 的并集上创建索引。容易证明, Primary DPrime 的并集也是 DPrime, 并且在 DPrime 的并集上创建索引的代价比在每一个 DPrime 上都创建索引的代价要小。

**定义 4(索引集 INDEX SET)**  $SV(V)$  上关于  $X$  的一个 INDEX SET 是指  $SV(V)$  中建立在相同维属性集  $X$  上的一组索引(不考虑属性次序), 记为  $I_{sv(v)}(X)$ 。

基于以上定义, 文[7]提出 DPrime Index Set Filter 索引过滤法, 即对一个概要视图, 只有它的 Primary DPrime 及其并集的 INDEX SET 才可以作为视图的候选索引, 我们将该方法简称为“优选属

性法”或 Dprime 法。下面让我们根据一个例子来进一步理解这种方法。

例 1: 下面的多维模型(如图 1 所示)来自于 TPC-D。该模型包括零件、供应商和客户三个属性。现将它们分别简写为 P、S 和 C。下面列出了 8 个视图的统计数据:

Size(SV(PSC))= 6M      Size(SV(PC))=6M  
 Size(SV(PS))=0.8M      Size(SV(SC))=6M  
 Size(SV(P))=0.2M      Size(SV(S))=0.01M  
 Size(SV(C))=0.1M      Size(SV( $\emptyset$ ))=1

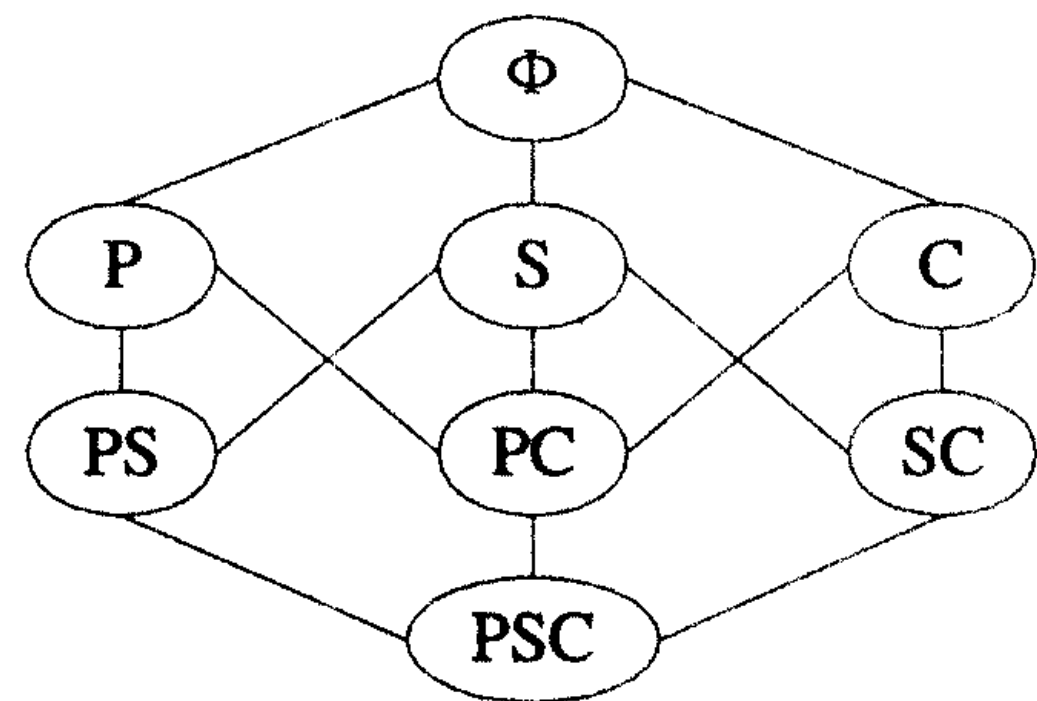


图 1 TPC-D

• 步骤 1. 索引过滤。

我们将阈值选定为 20, 所以有:

•  $SV(PSC)$  的 DPrime 是 PSC, PS, PC, SC, 它们的索引选择性分别为 1, 7.5, 1, 1。

•  $SV(PSC)$  的 Primary DPrime 是 PS, PC 和 SC。

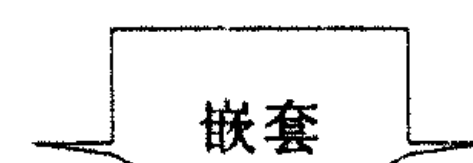
• 备选 INDEX SET 是  $I_{sv(psc)}(PS)$ ,  $I_{sv(psc)}(PC)$ ,  $I_{sv(psc)}(SC)$ ,  $I_{sv(psc)}(PSC)$ 。

• 步骤 2. 索引选择。

通过 DPrime 法进行索引过滤后, 候选集中基于属性集 (CS), (PC), (SP) 创建的索引将成为  $SV(PSC)$  的索引。

SALES\_FACE\_VIEW

Store	Code	User	Sum
S1	C1	U1	100
S1	C1	U2	200
S1	C1	U3	400
S1	C2	U1	300
S1	C2	U2	100



VARRAY\_SALES\_FACE\_VIEW

Store	Code	User1	User2	User3
S1	C1	100	200	400
S1	C2	300	100	

图 2 将 SALES\_FACE\_VIEW 中数据以 VARRAY 形式保存

通过例 1,我们可以发现通过 DPrime 法,可以使概要视图的索引规模大大降低。并且,根据选择出来的这种非唯一索引,文[7]提出了视图物化方法——嵌套关系法,该方法使用 Oracle VARRAY 将对于索引属性具有相同值的元组聚合成为一个超级元组,如图 2 所示。

#### 4 针对“嵌套关系法”和“优选属性法”的改进

虽然文[7]的索引选择方法和视图物化方法可以大大减少索引和视图的存储空间,并且可以取得满意的查询性能,但是,不难看出该方法仍旧存在一些缺点。

首先,虽然利用 VARRAY 实现的嵌套关系法具有一系列优点,但是该方法存在一个严重的缺陷,即 VARRAY 结构的维护是相当困难的。原因之一是,当给视图加入新的数据时,在传统的视图中仅需要插入一行,但是在这种方案中,就要更新视图中的所有行。另一个原因是,随着对数据仓库使用的逐渐深入,将迫使使用一些非常规(unconventional)结构<sup>[3]</sup>。因此 VARRAY 的功能虽然非常强大,但是维护起来的代价也是十分巨大的。这种缺陷限制了该方法在某些数据条件下的实现。

但是嵌套关系法的优点是极具吸引力的,因为利用该方法,我们仅仅访问一个元组就可以得到原来的需要访问多个元组才能得到的结果。这样不仅使视图的存储空间大为减少,也使对概要表的查询效率大为提高。于是我们便考虑是否存在一种新的方法,使得既能实现嵌套关系的全部优点,又能使系统易于维护并具有广泛的适用性。恰好,我们发现 Oracle 8i 为了实现嵌套结构定义了一种新的数据类型——Nested Table。我们以此为基础对文[7]的嵌套关系法进行了改进,提出了一种改进方法——嵌套表法。嵌套表法利用 Nested Table 数据类型,以嵌套表形式实现嵌套关系。在该方法中,我们将对于索引属性具有相同值的元组聚合成为一个超级元组,并用嵌套表的形式对这些超级元组进行保存,以此减少视图的存储空间和保持满意的查询性能。Nested Table 与 VARRAY 相比,最大的优势是数据组织灵活,便于进行数据维护。因此,在创建需要进行频繁刷新的物化视图时尤其应该考虑使用嵌套表法来组织数据。这样既可以充分发挥逆规范化处理的优势,又可以减轻 OLAP 系统的维护负担。此外,这种方法与前端工具兼容,通过在 Oracle 8i 中增加 TABLE 语法,可以方便地在传统的关系模型表中查看嵌套关系。嵌套表法的具体实现

方式如图 3 所示。

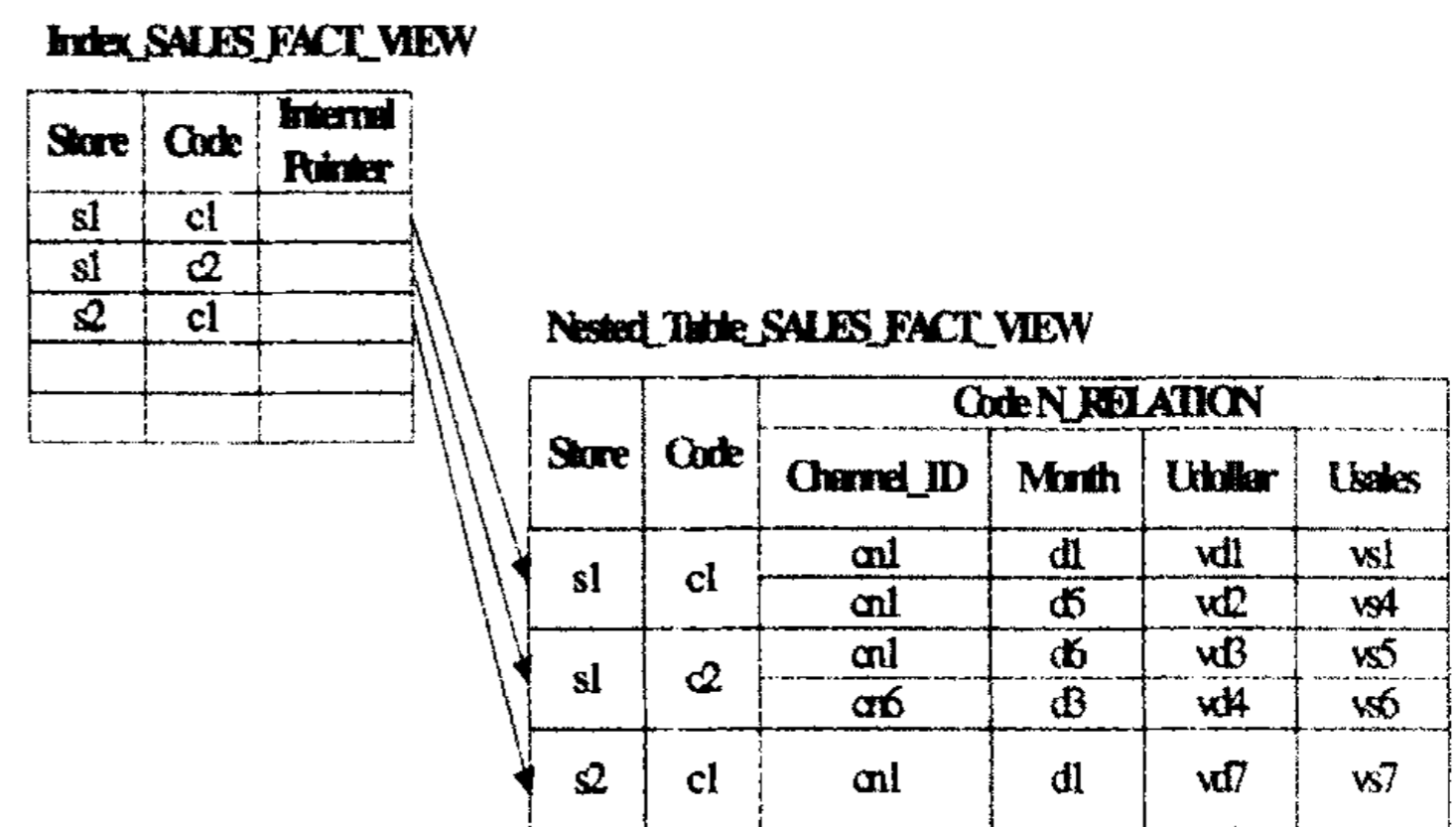


图 3 使用 Oracle Nested Table 实现的嵌套表法

其次,“优选属性法”也应该进一步优化。第一,在该索引选择过滤方法中,“索引选择性”被定义为与每个索引实例相关联的元组的平均数量,而这是远远不够的。在实际应用过程中,对索引选择性的考虑必须兼顾“百分比选择性”。使用“百分比选择性”的重要原因如下:1、使用“百分比选择性”便于准确地确定阈值;2、如果使用基于 rule 的最优化,优化器在确定执行路径时不会考虑非唯一索引的索引选择性(DPrime Index Set Filter 所产生的索引绝大多数为非唯一索引);3、对于碎片含量较高的索引,考虑“百分比选择性”有时会显得更有意义。第二,对于 INDEX SET 中属性的次序应该进行考虑。因为在 Oracle 数据库中索引属性的次序往往会直接影响查询性能,不合适的索引属性次序将大大降低查询效率。因此,除了选择出正确的 INDEX SET 外,还应该通过进一步实验验证或获得合适的索引属性次序。第三,必须指出的是,在实际的 OLAP 环境中,规定维数必须限定在 32 维之内(相当于传统关系型数据库中 32 个主键列)。而大多数 OLAP 应用只需要使用 5~7 维就足够了,并且几乎所有的 OLAP 应用都使用 10 维~12 维以下<sup>[4]</sup>。这样的话,将保证在实际的 OLAP 环境中,对于所有的 X 属于 V,可以通过有限次运算求得所有  $I_{sv(v)}(X)$  的索引选择性。

**总结** 在本文中,我们讨论了文[7]中所提出的“优选属性法”和“嵌套关系法”。我们对使用 VARRAY 实现的“嵌套关系法”进行了改进,提出了改进的视图物化方法——“嵌套表法”。另外,我们也对“优选属性法”的一些不足提出了改进意见。通过我们的方法使得索引过滤更加有效,使得视图物化的方法能够适应更多的实际应用情况。

因此可以得出如下结论:改进后的方法具有更广泛的适用性,除了具备原方法的优点,如有效地增加 OLAP 查询效率,极大地削减了存储开销,还使

得 OLAP 系统变得更加易于维护。

### 参考文献

- 1 Han Jiawei, Kamber M. Data Mining Concepts and Techniques. 机械工业出版社
- 2 KYTE T. Expert one-on-one Oracle. 清华大学出版社
- 3 Corey M. Oracle8i Data Warehousing. 机械工业出版社
- 4 Burleson D K. Oracle High-Performance SQL Tuning. 机械工业

出版社

- 5 Gupta H, Harinarayan V, Rajaraman A, Ullman J D. Index selection for OLAP. In: Proc. ICDE97, pp. 208~219
- 6 Qiu SG, Ling TW. View Selection in OLAP Environment, DEXA 2000
- 7 Qiu S G, Ling T W. Index Filtering and View Materialization in ROLAP Environment, CIKM'01, Atlanta, Georgia, USA. No. 2001
- 8 Oracle 8 Turning, release 8.0, pp. 10~13

(上接第 187 页)

C4.5 算法固有缺点的改进是有价值的。本文提出了一个实用有效的决策树改进模型——R-C4.5 决策树模型。在 R-C4.5 算法中,通过合并分类效果差的分枝,有效避免了 C4.5 决策树的不足,例如减少了碎片,避免了过度拟合,提高了分类准确率。

本文针对 R-C4.5 决策树模型的简化版本 R-C4.5s 模型进行了实验验证,并将实验结果与 C4.5 决策树模型进行了比较,充分证明了本文的分析。本文所提出的 R-C4.5 决策树模型未对连续型属性进行改进,对缺失数据的改进也未能加入到实验当中,这些将在后续的工作中完成。

表 1 C4.5 模型和 R-C4.5s 决策树模型的预测准确率比较及叶子和节点数比较

数据集	样本数	准确率			叶子数目			节点总数			叶子平均样本数		
		C4.5 (%)	R-C4.5s (%)	变化率 (%)	C4.5	R-C4.5s	变化率 (%)	C4.5	R-C4.5s	变化率 (%)	C4.5	R-C4.5s	健壮比
Autos	159	74.21	69.81	-5.93	30	11	-63.33	37	15	-59.46	5.30	14.45	2.73
Cylinder Bands	360	72.50	71.15	-1.86	45	16	-64.44	57	24	-57.89	8.00	22.50	2.81
Soybean	550	90.73	90.91	0.20	49	33	-32.65	70	52	-25.71	11.22	16.67	1.48
German	651	70.81	72.20	1.95	32	20	-37.50	43	33	-23.26	20.34	32.55	1.60
Car Evaluation	1728	92.36	87.21	-5.58	131	18	-86.26	182	31	-82.97	13.19	96.00	7.28
Mushroom	5571	99.96	100.00	0.04	22	15	-31.82	28	20	-28.57	253.23	371.40	1.47
Nursery	12960	97.05	86.81	-10.55	359	26	-92.76	511	44	-91.39	36.10	498.46	13.81
Adult	30162	82.49	80.99	-1.82	375	117	-68.80	422	143	-66.11	80.43	257.79	3.21
平均	—	—	—	-2.94	—	—	-59.70	—	—	-54.42	—	—	4.30

### 参考文献

- 1 Han J, Kamber M. Data Mining: Concepts and Techniques. Morgan Kaufmann Publish, 2001
- 2 Provost F, Kolluri V. A survey of methods for scaling up inductive algorithms. Data Mining and Knowledge Discovery, 1999, 3(2):132~169
- 3 Quinlan J R. C4.5, Programs for Machine Learning. Morgan Kaufmann, 1993
- 4 Breiman L, Friedman J H, Olshen R A, Stone C J. Classifica-

tion and Regression Trees. Wadsworth & Brooks/Cole Advanced Books & Software, Pacific Grove, CA, 1984

- 5 Lim T-S, Loh W-Y, Shih Y-S. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. Machine Learning, 2000, 40(3):203~228
- 6 Chaudhuri S, Dayal U, Ganti V. Database Technology for Decision Support Systems. Computer, 2001, 34(12)
- 7 史忠植. 知识发现. 北京:清华大学出版社, 2002. 29