第34卷 第18期 计 算 机 工 程 2008年9月
Vol.34 No.18 Computer Engineering September 2008
·软件技术与数据库· 文章编号：1000—3428(2008)18—0082—03 文献标识码：A 中图分类号：TP311.5

基于实例化视图的MDX语句执行性能优化

冯士心[1]，俞东进[2]
(1. 杭州市劳动保障信息中心，杭州 310003；2. 杭州电子科技大学计算机学院，杭州 310018)

摘 要：多维表达式(MDX)语句多用于实现对海量数据的多维分析，如何对MDX语句的执行过程进行优化，进而提高查询速度是构建在线分析处理系统的一个难点。该文采用预建实例化视图的方法，使得原先基于事实表和维表的多表连接查询在经过查询重写后，可直接利用单一的候选实例化视图完成，从而大大加快了MDX语句的执行速度。阐述了选取候选实例化视图的基本思路，并给出实验结果。实验证实，数据量越大、MDX语句越复杂，性能提升的效果越明显。

关键词：实例化视图；多维表达式；性能优化

Performance Optimization of MDX Statement Execution Based on Materialized Views

FENG Shi-xin[1], YU Dong-jin[2]
(1. Information Center of Labor and Social Security, Hangzhou 310003; 2. School of Computer, Hangzhou Dianzi University, Hangzhou 310018)

【Abstract】The Multi Dimensional Expression(MDX) statements are often used in the process of multi-dimensional analysis of massive data. How to optimize the statements in order to boost the query speed is one of key issues when constructing the OLAP systems. By the predefined materialized views, the query statements can be rewritten and executed against one single table instead of the join of one fact table and several dimensional tables. The paper presents the thought of how to select the candidate materialized views and also the experimental results, which show that the boosting effect is more encouraging while dealing with more data via more complicated MDX statements.

【Key words】materialized view; Multi Dimensional Expression(MDX); performance optimization

1 概述

Multi Dimensional Expression(MDX)支持对多维数据对象的定义和操纵，在很多方面与Structured Query Language (SQL)类似。一个MDX查询需要一个数据请求(SELECT子句)、一个出发点(FROM子句)和一个过滤器(WHERE子句)，这些子句以及其他一些关键词使得通过MDX查询能够在数据立方体中抽取特定的数据。MDX主要由微软公司提出，当前已经成为了多维查询语言的一个事实标准。由于MDX多用于实现对海量数据的多维分析，如何对MDX的执行过程进行优化、进而提高查询速度是构建在线分析处理(OnLine Analysis Processing, OLAP)系统的一个重点和难点[1]。

通过采用不同的MDX语句，在不同的数据量环境下进行测试，可发现建立数据库连接的时间一般保持在200 ms~300 ms左右，解析一条MDX语句的时间一般为1 000 ms左右，而MDX语句的执行时间则与具体MDX语句的复杂程度和数据量大小有很大关系。因此，对于OLAP系统性能改进的关键在于减少MDX语句的执行时间。

2 研究现状

实例化视图(materialized view)是指预先在数据存储层创建的、同时包含维度信息和测量信息的单一表结构和其中存储的数据内容。近来对实例化视图的研究已经成为了业界的一个热点。实例化视图本身无疑可以大大提高系统的响应性能，其技术关键是如何选择创建、维护实例化视图和如何合理选择使用实例化视图等。

实例化视图选择问题等价于一个最小集合覆盖问题，是一个NP问题，一般可采用试探法(如贪心法)求得此类问题的较优解。例如，文献[2]提出了在数据仓库中如何合理选取视图进行实例化的方法，其把视图分为2种：AND视图(不同的视图相互独立而需要单独计算)和OR视图(任意一个视图可以从与其关联的其他视图计算获得)，针对不同的视图采取不同的方法选择视图进行实例化，以达到优化总体查询响应时间的目的。文献[3]提出了一种判别一个SQL查询是否可以从实例化视图获得查询结果的启发式方法。另外，当实例化视图对应的基表数据发生更新变化时，实例化视图也要重新计算和增量更新，以保证更新后的实例化视图反映有效的基表状态，对于此类的实例化维护问题，也有不少文献加以研究。例如，文献[4]即深入讨论了这个问题。

3 实现思路

假设某个数据仓库采用由1个事实表和d个维表组成的星型存储模式，事实表由连接每个维表的外键以及测量值组成，每个维表只包含一个层次(Hierarchy)。维表DT_i的层次DH_i可以定义为：$DH_i=\left(L_0^i,L_1^i,\cdots,L_h^i,none\right)$，即表示为维度层次的不同级别的一个有序集。其中，$j$称之为$L_j^i$的高度；$L_0^i$为最底层级别，由$DT_i$的主键组成；$none$代表最高层级别(空

基金项目：浙江省重点科技计划基金资助项目“基于网格的分布式劳动力市场决策支持系统”(2007C2104)
作者简介：冯士心(1969－)，男，高级工程师，主研方向：电子政务，数据仓库技术；俞东进，教授级高级工程师
收稿日期：2008-07-10 **E-mail：**fsxyfbx@126.com

集)；并且有：$L^i_{j-1} \to L^i_j, (1 \leqslant j \leqslant h)$。定义 $DH = DH_1 \times DH_2 \times \cdots \times DH_d$，表示为所有维度层次的维度级别的一个笛卡儿有序集。该有序集的元素存在如下的偏序关系：

$$\left(L^1_{l_1}, L^2_{l_2}, \cdots, L^d_{l_d}\right) \leqslant \left(L^1_{m_1}, L^2_{m_2}, \cdots, L^d_{m_d}\right)$$

当且仅当 $L^i_{l_i} \to L^i_{m_i}$ 或 $L^i_{l_i} = L^i_{m_i}$ $(1 \leqslant i \leqslant d)$；

$$\left(L^1_{l_1}, L^2_{l_2}, \cdots, L^d_{l_d}\right) < \left(L^1_{m_1}, L^2_{m_2}, \cdots, L^d_{m_d}\right)$$

当且仅当 $\left(L^1_{l_1}, L^2_{l_2}, \cdots, L^d_{l_d}\right) \leqslant \left(L^1_{m_1}, L^2_{m_2}, \cdots, L^d_{m_d}\right)$，并且存在 j，有 $(1 \leqslant i \leqslant d)$：$L^i_{l_i} \neq L^i_{m_i}$；

$$\left(L^1_{l_1}, L^2_{l_2}, \cdots, L^d_{l_d}\right) <> \left(L^1_{m_1}, L^2_{m_2}, \cdots, L^d_{m_d}\right)$$

当且仅当既不 $\left(L^1_{l_1}, L^2_{l_2}, \cdots, L^d_{l_d}\right) \leqslant \left(L^1_{m_1}, L^2_{m_2}, \cdots, L^d_{m_d}\right)$，也不 $\left(L^1_{l_1}, L^2_{l_2}, \cdots, L^d_{l_d}\right) \geqslant \left(L^1_{m_1}, L^2_{m_2}, \cdots, L^d_{m_d}\right)$

在一般的多维数据查询中涉及的 OLAP 查询都为非嵌套的单块(single-block)聚集查询，这些查询为标准化的 OLAP 查询。一个标准化了的 OLAP 查询 Q 可以定义为如下的一个 5 元组：

$$Q(SG, R, AG, AGG, HAV)$$

其中，$SG = (S_1, S_2, \cdots, S_d)$ 表示 Q 的选择粒度，$S_i (1 \leqslant i \leqslant d)$ 是维度层次 DH_i 的一个维度级别；$R = \{R_i\}$ 表示 Q 的选择区域，即为一超矩形集合，超矩形 $R_i = (I_{i1}, I_{i2}, \cdots, I_{id})$ 是查询 Q 的选择谓词的维度级别的间隔值的一个有序集合，I_{ij} 表示 DH_j 的一个维度级别的间隔；$AG = (A_1, A_2, \cdots, A_d)$ 表示 Q 的聚合粒度，$A_i (1 \leqslant i \leqslant d)$ 是维度层次 DH_i 的一个维度级别；$AGG = \{agg(m) \mid agg \in \{MIN, MAX, SUM, COUNT\}\}$，$m$ 是一个测量属性；HAV 是基于 AGG 聚合函数和 AG 聚合属性的一个比较谓词逻辑表达式，表示 SQL 语句中的 HAVING 条件。

一个实例化视图 MV 存储了一个标准化的 OLAP 查询 Q 的查询结果，并且满足 $SG(Q) \geqslant AG(Q)$，且 $HAV(Q) = \text{null}$。所以，标准的实例化视图可以用一个 4 元组表示：

$$MV(SG, R, AG, AGG)$$

如果同时满足下列 2 个条件，则可称查询 Q'是查询 Q 利用实例化视图 MV 进行改写后的查询：(1)Q'和 Q 对于同一数据库计算得到一致结果；(2)Q'在至少一个查询块的 FROM 子句中包含 MV。

如果存在这样的 Q'，则可称 MV 在改写 Q 的过程中是有用的(usable)，或者称 MV 是 Q 的候选实例化视图。

下面讨论对于一个标准化 OLAP 查询的改写，如何找到有用的实例化视图。设 Q 和 MV 为一标准化 OLAP 查询和一实例化视图，定义：

$$R(Q) \cap^* R(MV) = \{R_i \cap R_j \mid R_i \in R(Q), R_j \in R(MV)\}$$

其中，$R_i \cap R_j$ 是 R_i 和 R_j 的一个超矩形交集。

$$R(Q) -^* R(MV) = \{R_k \mid R_k \in (R_i - R_j), R_i \in R(Q), R_j \in R(MV)\}$$

可以证明，满足以下 4 个条件的实例化视图 MV 可以作为一个 OLAP 查询 Q 的候选实例化视图：

$$R(MV) \cap^* R(Q) \neq \phi, AG(MV) \leqslant SG(Q),$$
$$AG(MV) \leqslant AG(Q), AGG(MV) \supseteq AGG(Q)$$

所以，采用基于维表 DT_i 的维度层次 DH_i 的最底层级别 L^i_0 与事实表 F 进行自然连接得到的实例化视图 $MV = F \triangleright\!\triangleleft DT_1 \triangleright\!\triangleleft DT_2 \triangleright\!\triangleleft \cdots \triangleright\!\triangleleft DT_d$，可以用来对标准化的 OLAP 查询 Q 进行改写，以提高查询性能。

4 实现方法

一般地，利用实例化视图提高系统响应性能的设计主要包括 2 个部分，即实例化视图的创建和查询改写。

在多维数据分析系统中，一个立方体(cube)可以预定义一个实例化视图，这可以通过解析模型定义文件完成，即以一个事实表和多个维表为基础，并通过事实表和维表之间的外键和主键联系，创建实例化视图。

创建了实例化视图后，必须通过自动的查询改写，才能使普通的对事实表和维表的连接查询转换为对实例化视图的查询。

在提交 MDX 查询语句时，首先转换为对应的 SQL 语句，然后判断是否存在对应该主题的候选实例化视图。如存在，则进行 SQL 语句改写，并把改写后的 SQL 语句递交给数据库执行，最后获取查询结果。

5 实验结果

在实验中，分别采用 3 条典型的 MDX 语句，针对对于不同的数据量，比较利用候选实例化视图前后的性能变化。其中，Sales 为事实表，Promotion Media, Product, Time 为维表，Unit Sales 为测量值。

3 条 MDX 语句分别如下：

MDX1: (简单 MDX 语句，实际包含 1 个事实表和 2 个维表的连接)

```
select {[Measures].[Unit Sales]} ON columns, {([Promotion Media].[All Media], [Product].[All Products])} ON rows from [Sales]
```

MDX2: (较复杂 MDX 语句，实际包含 1 个事实表和 2 个维表的连接，并含有一个过滤器)

```
select {[Measures].[Unit Sales]} ON columns, Hierarchize (Union({[Promotion Media].[All Media]}, [Promotion Media].[All Media].Children)) ON rows from [Sales] where [Time].[1997]
```

MDX3: (复杂 MDX 语句，实际包含 1 个事实表和 2 个维表的连接，并含有一个过滤器)

```
select {[Measures].[Unit Sales]} ON columns, Hierarchize (Union({([Promotion Media].[All Media], [Product].[All Products])}, Crossjoin({[Promotion Media].[All Media]}, [Product].[All Products].Children))) ON rows from [Sales] where [Time].[1997]
```

实验环境为：主频 1.4 GHz，内存 512 MB，Windows XP Professional, MySQL 4.0.20, JDK 1.4.2_04。事实表和维表的主键和外键都建有索引，候选实例化视图则无任何索引建立。

图 1 是 MDX1 语句在不同数据量条件下采用候选实例化视图前后的执行速度对照图。

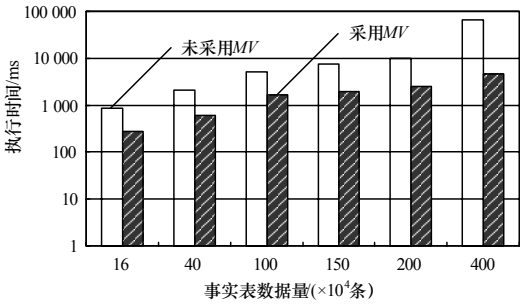

图 1 对于不同数据量 MDX1 采用实例化视图前后的性能对比

图 2 是 MDX1, MDX2 和 MDX3 3 条语句在 40 万条事实表记录中采用候选实例化视图前后的执行速度对照图。

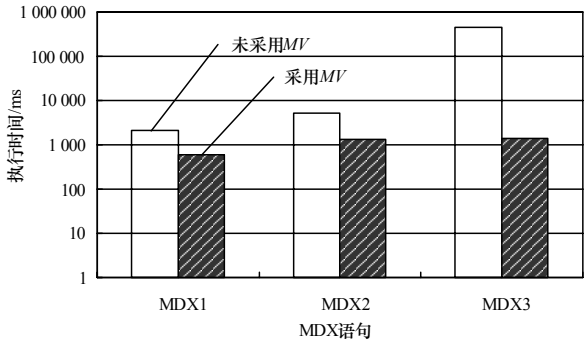

图 2　不同 MDX 语句采用实例化视图前后的性能对比

实验表明:

(1)采用实例化视图可明显地提高 MDX 语句查询速度;

(2)数据量越多，采用实例化视图提高性能越明显;

(3)MDX 语句越复杂，采用实例化视图提高性能越明显;

(4)采用实例化视图后，随着数据量的增大和 MDX 语句复杂度增加，性能的下降幅度远比在同等情况下未采用实例化视图时小。

6　结束语

实例化视图可以显著提高 MDX 语句的查询速度，其应用关键是针对性地创建有用的候选实例化视图，使得在 MDX 语句递交执行时，可以更改为面向候选实例化视图的 SQL 查询语句，从而通过变多表关联查询为单表查询，最终达到提高查询速度的目的。

预建实例化视图的一个缺陷是需要大容量的存储空间。因此，如何通过只创建少量实例化视图而满足绝大部分的 MDX 查询要求是未来进一步研究的重点。

参考文献

[1] 俞东进, 赵明德. 一个轻量级数据仓库平台的设计和实现[J]. 计算机工程, 2005, 31(15): 206-207, 213.

[2] Gupta H. Selection of Views to Materialize in a Data Warehouse[C]//Proc. of the 6th International Conference on Database Theory. Delphi, Greece: [s. n.], 1997: 98-112.

[3] Goldstein J, Larson P. Optimizing Queries Using Materialized Views: A Practical Scalable Solution[C]//Proc. of the ACM SIGMOD'01. Santa Barbara, California, USA: [s. n.], 2001: 331-342.

[4] 沈　波, 潘久辉. 数据仓库环境下的实化视图维护[J]. 计算机工程与科学, 2004, 26(3): 69-73.

(上接第 78 页)

本文方法本质上仍属于静态法，只是通过给定部分参数初值减少方程组变量的元数直至其成为一元方程组，以降低方程组求解的难度。此方法适用于含有数组下标变量和数组元素变量约束方程组的求解。

但本文方法存在回溯，且无法保证算法在有限次数回溯之后能求解。如何实行有效回溯，减少回溯次数有待进一步研究。

参考文献

[1] Gupta N, Mathur A P, Sofia M L. Generating Test Data for Branch Coverage[C]//Proceedings of the 15th IEEE International Conference on Automated Software Engineering. [S. l.]: IEEE Press, 2000.

[2] Gupta N, Mathur A P, Sofia M L. Automated Test Data Generation Using an Iterative Relaxation Method[C]//Proceedings of the 6th ACM SIGSOFT International Symposium on Foundations of Software Engineering. Orlando, Florida, USA: ACM Press, 1998.

[3] 单锦辉. 面向路径的测试数据自动生成方法研究[D]. 哈尔滨: 国防科学技术大学, 2002.

[4] Davis M. Hilbert's Tenth Problem is Unsolvable[J]. The American Mathematical Monthly, 1973, 80(3): 233-269.

[5] Weyuker E J. The Applicability of Program Schema Results to Programs[J]. International Journal of Computer Information Sciences, 1979, 8(5): 387-403.

[6] 侯　睿. 基于约束满足问题的空间方向关系推理[D]. 重庆: 重庆大学, 2005.

(上接第 81 页)

的实用价值。但随着分销环境的动态变化及移动技术的发展会出现新的问题及技术解决方法，将不断完善该信息系统，使分销信息管理过程更快、更准、更便捷，实现供应链管理中物流与信息流的一致。

参考文献

[1] 中兴网络有限公司. Sybase 零售制造业移动商务解决方案[EB/OL]. (2004-09-03). http://www3.it168.com/solution/20040903/200409034546.pdf.

[2] 叶晓俊, 海　翔. 企业信息移动发布平台的设计与开发[J]. 计算机集成制造系统, 2003, 9(11): 972-975.

[3] 周宽久, 曾琳铖曦, 李　瑶. 一个语音信息门户的设计与实现[J]. 计算机工程, 2006, 32(9): 101-103.

[4] 仲萃豪. SOA 的十大技术理论体系[J]. 中国计算机用户, 2006, (15): 28-29.

[5] Pastore S. The Service Discovery Methods Issue: A Web Services UDDI Specification Framework Integrated in a Grid Environment[J]. Journal of Network and Computer Applications, 2006, 31(2): 93-107.

[6] Min Junki, Lee Chunhee, Chung Chinwan. XTRON: An XML Data Management System Using Relational Databases[J]. Information and Software Technology, 2007, 50(5): 462-479.

[7] Baker M, Masayasu I, Shinichi M, et al. XHTML Basic 1.0[EB/OL]. (2000-12-19). http://www.w3.org/TR/2000/REC-xhtml-basic-20001219.