基于查询优化的虚拟数据仓库模型

徐 强 李文烨 顾毓清
(中国科学院软件研究所 北京100080)

Query-Optimization Based Virtual Data Warehouse Model

XU Qiang LI Wen-Ye GU Yu-Qing
(Institute of Software, Chinese Academy of Sciences, Beijing 100080)

Abstract The technology of virtual data warehouses catches more and more interests of people. In this paper, a Query-Optimization Based Virtual Data Warehouse (QVDW) model is presented, which not only has all the advantages that a virtual data warehouse has, but also can change bottom layer data structures and the model itself according to users' various query requests. The aim is to offer customized service to OLAP (on-line analysis processing) users.

Keywords Virtual data warehouse, OLAP (on-line analysis processing)

1 虚拟数据模型概述

虚拟数据仓库技术因为其开放灵活的体系结构、以需求为驱动、无限的扩展性等优点而越来越引起人们的关注，相比传统数据仓库以供给为驱动的特点，虚拟数据仓库对有很多不同时期、不同构、复杂的数据源的大公司大企业来说有巨大的吸引力。本文在此技术的基础上，提出了一个基于查询优化的虚拟数据仓库模型，它使用多层次分布式的数据结构，在OLAP/DSS用户和底层的物理数据仓库之间建立了透明的元数据服务器(中间件)，使得OLAP/DSS用户不再关心底层数据存贮的具体结构、解决方案、存储策略；同时，数据仓库的物理结构以及虚拟结构则根据OLAP/DSS需求的不断变化自动地优化和调整而不需OLAP/DSS用户的额外干预，相对于一般的虚拟数据仓库技术，基于查询优化的虚拟数据仓库具有更大的灵活性和主动性。虚拟数据仓库淡化了数据仓库集成的特点，而本模型进一步淡化了数据仓库不可修改的特点，目的都是为了能够更灵活更方便地为OLAP/DSS提供服务。在文章的最后，重点分析了用户请求解决方案的选择和自动生成算法，该算法能够较好地响应OLAP/DSS用户的需求。

2 数据模型以及层次结构

基于查询优化的虚拟数据仓库模型为四层结构，自上而下分别为：应用层、虚拟元数据层、物理元数据层和物理数据层。

2.1 应用层

应用层位于模型的顶层，该层提供查询应用接口和转换机制，接收外部应用(XML、SQL、自然语言等各种格式)的调用，然后转换成符合虚拟元数据层要求格式的请求。它同时承担从虚拟元数据层返回数据的展示、输出任务。

2.2 虚拟元数据层

虚拟元数据层在整个虚拟数据仓库的第二层，也是所有数据仓库请求的唯一入口。它直接接收从应用层得到的请求，建立请求与物理元数据层数据间的映射，从而得到实际可以执行在物理数据层的解决方案。虚拟元数据层本身具有一定的数据接受、数据运算能力。

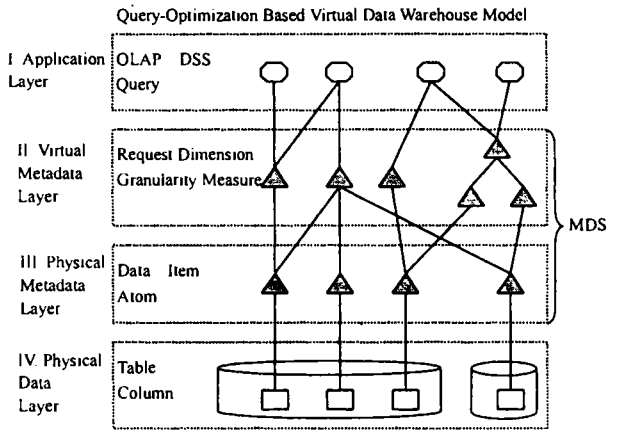

图1

请求 描述的是用户向数据仓库提交的数据访问请求，它由维度、粒度、度量等属性组成，可以理解为一个多维数据空间上的数据体。可以通过交、并、差、选择、衍生、笛卡儿乘积、维集聚等操作对问题进行加工、分解和演变。其中前5个操作不改变问题的维度空间，而后面2个操作可能会引起维度空间的变化。

维度 刻画用户问题中数据空间中的各个维度，即数据体上的边，例如时间、地域等。每个维之间是相互独立的，不存在交叉。

粒度 刻画用户问题中每个维度上的统计粒度。粒度必须依赖于某个维度才有意义，例如时间维度上的月、日等粒度，地域维度上的省、市等粒度。在同一维度的不同粒度间可能存在"偏序"关系，例如月可以由日汇总得到，年可以从月汇总得到等。这种"偏序"关系在维度的各个粒度之间构成了一个有向无环图。

度量 刻画用户问题中的数据，即数据体上内部的点，例如销售额、人口数、平均年龄等。度量必须描述在一定的维度上才有其意义，例如某商场的销售额，某省的人口数等。度量集合通过函数R依赖维度：

$R: DOM(d_1) \times DOM(d_2) \times \cdots \times DOM(d_n) \rightarrow DOM(m_1) \times DOM(m_2) \times \cdots \times DOM(m_k)$

规则 与度量相关的还有一个规则的概念，规则描述了

度量之间的组合、衍生关系，比如人口数以及国民生产总值可以派生出人均国民生产总值度量。度量有直接度量(Direct Measure)和间接度量(Indirect Measure)两种，直接度量在物理元数据库中存在具体的数据项(Item)对应，而间接度量则需要通过 Rule 由直接度量衍生。一个度量可以既是直接度量又是间接度量，表示它既在物理元数据库中存在对应的数据项(Item)，也可以通过其他的度量衍生得到。

$$\text{Request}=\{r_1,r_2,\cdots,r_s\}$$
$$\text{Dimension}=\{d_1,d_2,\cdots,d_n\}$$
$$\text{Measure}=\{m_1,m_2,\cdots,m_k\}$$
$$\text{Granularity}=\{g_{11},g_{12},\cdots,g_{21},g_{22},\cdots,g_{n1},g_{n2},\cdots\}$$
$$(\text{Dimension},\text{Granularity})=\{(d_1,g_1),(d_2,g_2),\cdots,(d_n,g_n)\}$$
$$\text{Request}=((\text{Dimension},\text{Granularity}),\text{Measure})$$
$$=(d_1,g_1,d_2,g_2,\cdots,d_n,g_n,m_1,m_2,\cdots,m_k)$$

请求的同构(Homotype)　请求 $R=(d_{r1},g_{r1},d_{r2},g_{r2},\cdots,d_{rn},g_{rn},m_{r1},m_{r2},\cdots,m_{rs})$，$Q=(d_{q1},g_{q1},d_{q2},g_{q2},\cdots,d_{qn},g_{qn},m_{q1},m_{q2},\cdots,m_{qt})$ 同构，当且仅当两个请求的维度、粒度可以一一对应，即在同一个问题的空间。如果给数据仓库中的维度进行编号，按照维度编号的升序排列，则 R,Q 同构当且仅当 $d_{r1}=d_{q1},g_{r1}=g_{q1},\cdots,d_{rn}=d_{qn},g_{rn}=g_{qn}$。记为：$R\backsim Q$。请求的交、并、差等操作必须在同构的前提下才能进行，不同构的请求必须先通过笛卡儿乘积、维集聚等操作转化成同构请求，才能进行交、并、差操作。

请求交　如果请求 R,Q 同构，则：$R\cap Q=\{S|S$ 同时满足 R 以及 $Q\}$，$R\cap Q$ 仍然和 R,Q 同构，同时度量为 R,Q 度量的合并，也可以记为 INT(R,Q)。

请求并　如果请求 R,Q 同构，则：$R\cup Q=\{S|S$ 满足 R 或者满足 $Q\}$，$R\cup Q$ 仍然和 R,Q 同构，同时度量为 R,Q 度量的合并，也可以记为 UNI(R,Q)。

请求差　如果请求 R,Q 同构，则：$R-Q=\{S|S$ 满足 R 同时 S 不满足 $Q\}$，$R-Q$ 仍然和 R,Q 同构，同时度量为 R,Q 度量的合并，也可以记为 SUB(R,Q)。

请求的选择　SEL(R,Precondition)＝{S|S 满足 R 同时其维度粒度度量满足条件 Precondition}，SEL(R,Precondition)的维度粒度度量均与 R 相同。

请求的衍生　DEV(R,Rule)＝{S|S 维度粒度均与 R 相同，同时度量通过 Rule$(d_1,g_1,\cdots,m_k)$演变得到}。

请求的笛卡儿乘积　$R\times Q=\{S|S$ 的维度粒度是 R,Q 维度粒度的叉乘，同时度量为 R,Q 度量的合并$\}$，也可以记为 MUL(R,Q)。

请求的维集聚　$R=(d_1,g_1),(d_2,g_2),\cdots,(d_n,g_n),m_1,m_2,\cdots,m_k$，假设粒度 d_i 上存在偏序 $g_i\rightarrow q_i$，即粒度 g_i 可以归并到粒度 q_i，例如：时间维上日粒度可以归并到月粒度，则：COL(R,d_i,g_i,q_i)＝{S|S 的其他维度粒度均与 R 相同，(d_i,g_i)归并为(d_i,q_i)，相应的度量也进行 sum,avg…等归并操作。逻辑上 COL(R,d_i,g_i,q_i)是在统计粒度上的合并，引起了统计粒度的变粗，也有可能引起某个维度的简化，从而改变了请求的维度空间。

2.3　物理元数据层

物理元数据层介于虚拟元数据层和物理数据层之间，它本身依附于每一个具体的物理数据仓库，逻辑地描述每个物理数据仓库所能实现的数据。在虚拟数据仓库结构中，可能存在多个物理的数据仓库(也可能是其他任何形式的分布式数据源)，每个物理的数据仓库则对应着一个用来描述它的物理元数据层。

数据项　对数据仓库中每一项数据的逻辑描述，它由原数据构成，例如数据项地域可能由省编号、市编号、县编号3项原数据组成。同时，数据项之间也存在组合的关系，复杂的数据项可以分解成简单的数据项。

原数据(Atom)　对数据仓库表格中的每一个数据列的直接逻辑描述，它与数据仓库中表格的列是一一对应的关系。

数据　对数据仓库中表格的直接逻辑描述，它由数据项 Item 组成，与数据仓库中的表格是一一对应的关系。

$$\text{Item}=\{i_1,i_2,\cdots,i_n\}$$
$$\text{Atom}=\{a_1,a_2,\cdots,a_n\}$$
$$\text{Data}=\{d_1,d_2,\cdots,d_n\}$$
$$\text{Data}=D\ (\text{Item})$$
$$\text{Item}=I\ (\text{Item})$$

Item＝I (Atom)

2.4　物理数据层

物理数据层对应着具体各个物理的数据仓库(也可能是其他任何形式的分布式数据源)，也就是物理数据实际存储的地方。

表格　实际描述数据仓库中的数据表格。

数据列　实际描述数据仓库中表格的各个数据列。

$$\text{Table}=\{t_1,t_2,\cdots,t_n\}$$
$$\text{Column}=\{c_1,c_2,\cdots,c_n\}$$
$$\text{Table}=T\ (\text{Column})$$

2.5　层次结构以及数据映射

虚拟元数据层和物理元数据层合称为元数据服务器，它为用户提供两种方式的服务。一种是直接作为用户访问的代理，用户对元数据服务器提出请求，并通过它获得物理数据层的分布式数据源的结果，这种方式称为代理模式(DB Agent Mode)；一种是用户对元数据服务器提出请求，元数据返回用户访问物理数据层的分布式数据源的操作对象(DB Operation Object)，用户根据操作对象(Object)直接对底层的物理数据进行操作，这种方式称为引用模式(DB Reference Mode)。两种模式见图2和图3。

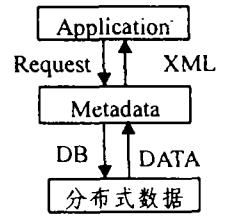

图2　DB Agent Mode

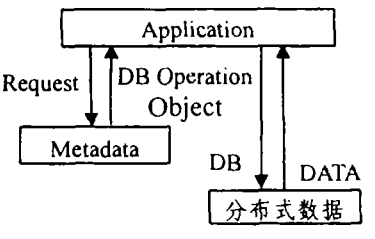

图3　DB Reference Mode

·用户请求与请求间的映射属于多对多映射，即一个用户请求可能被描述成几个不同的请求，或者说一个用户请求可以通过很多种解决方案来解决。

·请求与数据间的映射属于多对多映射，即一个请求的解决可能需要多个数据来提供，甚至这些数据可能来自不同的物理数据仓库。同样一个数据也可以为多个请求提供解决方

案。

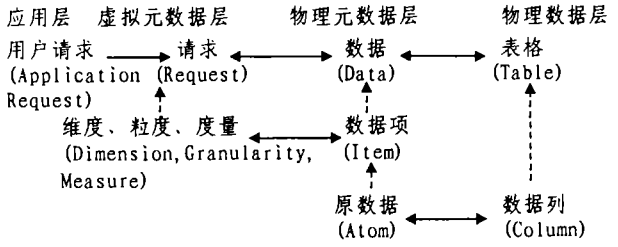

图4

·数据与表格间的映射属于一对一映射，即一个数据逻辑地描述一个数据仓库中的表格。这里说的表格指的是事实表，而不是维表。

·维度、粒度、度量等与数据项间的映射属于多对多映射。对于一个数据项而言，可能是一个维度、粒度，也可能是一个度量。

·原数据(Atom)和数据列间的映射属于一对一映射，即一个原数据(Atom)具体描述一个数据表格中的数据列。

·请求可以细化成维度、粒度、度量的组合，即一个请求可以用一组维度、粒度、度量来描述。

·数据可以细化成数据项的组合，即一个数据可以用一组数据项来描述。

·数据项可以细化成原数据(Atom)的组合，即一个数据项可以用一组原数据(Atom)来描述。

表格可以细化成数据列的组合，即一个表格可以用一组数据列来描述。

3 查询优化

查询优化主要体现在两方面：

A. 用户请求解决方案的选择以及自动生成

1)对于可以直接映射到多个物理元数据层数据的请求，选择较好(体现在系统代价小、响应时间短等)的解决方案为其提供数据。

2)对于不能直接映射到物理元数据层数据的请求，通过交、并、差、选择、笛卡尔乘积、衍生、维集聚等数据操作，使之可以映射到一个或者多个物理元数据层数据，从而生成一个新的解决方案。

B. 各层结构以及数据根据用户请求日志的调整

1)对于用户经常提交的请求，如果物理层没有直接对应的数据，则在物理层实现对应的数据，避免重复的中间计算操作。

2)对于用户经常提交的请求，将物理层对应的数据存放到较快的物理服务器上，以提高响应速度；同样，用户不经常提交的请求，将物理层对应的数据存放在较差的服务器上，以免影响整体速度。

3)根据日志，合理调整虚拟元数据层的元数据，对于经常作为条件访问但不是维度粒度的数据项，抽象成维度粒度；对于经常作为查询结果访问但不是度量的数据项，抽象成度量；反之，对于几乎不访问的维度、粒度、度量，可以从虚拟元数据层删除。

4)尽量将经常并发的请求分放在不同的物理数据仓库中，以提高虚拟数据仓库中各个物理数据仓库的使用率。

4 算法—用户请求解决方案的选择和自动生成

4.1 优化原则

用户请求解决方案的选择和自动生成算法的最终目的是为用户提交的请求寻找合适的(可能的或者说较好的)解决方案。在解决方案的选择上，有如下的一些原则：

·如果请求可以由某个物理元数据库(一个物理元数据库对应一个物理的数据仓库)提供解决方案，也可以由多个物理元数据库组合在一起提供解决方案，则优先选择只用一个物理元数据仓库情况，避免跨平台的数据操作，这样可以充分利用底层物理数据仓库解决问题的能力。

·如果请求可以由某个物理元数据层中一个数据提供解决方案，也可以由某个物理元数据库中多个数据提供解决方案，则优先选择只用一个数据的情况，避免跨表格的数据操作。

·如果请求的演算(例如请求的交、并等)可以在顶层的虚拟元数据层完成，也可以在底层的物理层通过数据库操作实现，则优先选择通过底层运算实现。只有必须涉及多个数据库之间操作的情况，才在虚拟层实现。这样可以充分利用底层物理数据仓库解决问题的能力。

4.2 辅助概念

·*快表* 对于用户经常提交的一部分请求为其编号并存放在快表中。当用户的请求和快表中记录匹配时，就直接可以找到物理元数据层相应的为其提供解决方案的数据(Data)。

·*主题* 为物理元数据层的数据建立主题索引。当请求寻找匹配的数据时，提供检索索引以提高效率以及限制目标集合的大小。

·*匹配系数* 表示一个数据能够为一个请求提供解决方案的可能性，匹配系数为0表示完全匹配，即请求可以由数据直接解决；如果缺少某个维度、或者缺少某个度量、或者某个粒度需要归并等，则相应的匹配系数就会增加。匹配系数是选择较好的解决方案的主要依据之一。

·*数据性能* 衡量某个数据的性能的参数，包括其所在物理数据仓库的数据处理能力，数据量大小，网络流量，数据的索引情况，数据维度粒度汇总能力等。数据性能是选择较好的解决方案的主要依据之一。

4.3 算法描述

1 确定请求的元组：$G=(d_1,g_1,d_2,g_2,\cdots,d_n,g_n,m_1,m_2,\cdots,m_k)$。同时也包括选择条件，度量衍生规则等的确定。

2 首先在快表中查找是否有匹配(元组完全相同)的记录。如果有，则可以直接获得相应的提供解决方案的数据集合，跳到算法的第9步。

3 根据请求涉及的主题，确定可能提供解决方案的物理元数据层的数据集合，依次计算相应的匹配系数 F(R,D)。如果 Data 中的 Item 全部匹配元组 $G=(d_1,g_1,d_2,g_2,\cdots,d_n,g_n,m_1,m_2,\cdots,m_k)$，则匹配系数为0；如果 Data 中的 Item 中不含有 d_i,g_i 或者不含有 m_j，匹配系数加某个值；如果 Data 中的 Item 包含了更多的维度、粒度或存在(d_i,g_i,q_i)维度粒度合并操作，则匹配系数加某个值。

4 检索完毕后，依次选择匹配系数最小的 Data。

5 如果 F(R,D)为0，表示请求 R 可以直接从 D 获得解决方案。从这些 Data 中，选择数据性能最好的数据作为解决方案的提供者，然后跳到算法的第9步。

6 如果最小 F(R,D)不为0，即没有直接匹配的数据，则如果 D 需要进行维度粒度的归并(维集聚)，则选择数据性能

(下转第86页)

参 考 文 献

1 Cho J, Shin H. Design issues for multimedia information servers in mobile computing environment. in: Proc of the 4th Intl. Workshop on Mobile Multimedia Communications, Sep. 1997. 336～339

2 Alonso R, et al. Managing video data in a mobile environment. SIGMOD Record, 1995, 24(4): 28～33

3 Bahl P. Supporting digital video in a managed wireless network. IEEE Communications Magazine, June 1998. 94～102

4 Vass J, et al. Mobile video communications in wireless environment. In: Proc. of IEEE Workshop on Multimedia Signal Processing, Copenhagen, Denmark, Sept. 1999. 45～50

5 Vass J, et al. Error resilient video communications in wireless ATM. IEEE Journal on Selected Areas in Communications, June 2000

6 Vass J, et al. 3DSLCCA-A high scalable very low bit rate software-only wavelet video codec. In: Proc. of IEEE Workshop on Multimedia Signal Processing, Dec. 1998

7 Bahl P, Chlamtac I. H. 263 based video codec for real-time visual communications over wireless radio network. In: Proc. IEEE Int. Conf. on Universal Personal Communications, San Diego, CA, Oct. 1997

8 Gall D L. MPEG: A video compression standard for multidedia applications. Communications of ACM, 1991, 34(4): 46～58

9 Imielinski T, Badrinath B R. Mobile woreless computing: Challenges in data management. Communications of ACM, 1994, 37(4): 38～47

10 Filippini L. MPEG-FAQ. http: // www. crs4. it/HTML/LUIGI/MPEG/mpegfaq. html, July, 1995

11 Ozer J. Video compression for multimedia. Academic Press, 1995

12 Shannon C E. Communications in the presence of noise. In: Proc. IRE, vol. 37, Jan, 1949

(上接第65页)

最好的数据作为解决方案的提供者，然后跳到算法的第9步。

7 请求需要数据笛卡儿乘积操作才可能获得解决，则根据 D 中 Item 对应的元组 H，找到另外一个数据 D′的元组 H′，使得 H×H′=G。

8 递归执行该算法以获得 D′。如果仍然需要 D″，则继续递归，直到获得能够满足请求的数据集合，如果搜索了所有的相关的数据仍然不能满足需要，则请求无解，算法结束。在选择 D′，D″等的时候，要遵循尽量在一个物理元数据库的原则。

9 找到了满足请求的数据集合(D, D′, D″…)，如果请求本身含有交操作，则试图通过数据的 and 操作来完成；同理还有或、差、选择、衍生等操作用物理级的 or、not、where 等代替。

10 在物理层实现每个 Data 的演算，在同一个物理数据源的笛卡儿乘积等操作则在物理层通过表格的连接操作(join)实现。最后把 XML 格式的 Data 集合提交到虚拟层。

11 在虚拟层完成底层无法完成的一些请求演算。

12 最后将规范的请求结果集合(XML 格式)提交到应用层。

13 将问题的解决方案写入虚拟数据仓库日志，以便于数据模型的自身优化。

4.4 算法分析

·算法的动态有穷性 本算法是机械执行的，并且是在一个有限的数据集合上进行请求解决方案的搜索，所以总是能够终止的。

·算法的有效性 算法可以回答虚拟数据仓库可以回答的所有问题，因为依靠维度、粒度、度量、主题等概念，算法从最可能的、效果最好的数据开始搜索一直到数据集穷举完毕。并且，虽然算法不能保证总给出最优解，但算法总是能给出一个相对稳定的较好的解。

·算法的优越性 基于查询优化的虚拟数据仓库算法比传统的虚拟数据仓库算法而言的优势在于其底层的数据操作更加灵活地根据需求的变化而变化。虚拟数据仓库淡化了数据仓库集成的特点，而本模型进一步淡化了数据仓库不可修改的特点，目的都是为了能够更灵活更方便地为 OLAP/DSS 提供服务。

5 相关工作的比较以及继续研究的问题

1. 关于数据仓库的多维数据模型

在数据仓库的构建过程中，数据模型的设计至关重要，因而也成为了数据仓库研究的核心问题之一。文[1]以“偏序”和“映射”为基础，给出了一种多维数据模型以及表达数据仓库的复杂数据结构和语义，并提供了一个以 OLAP 操作为核心的操作代数。

2. 关于虚拟数据仓库

需求驱动的数据仓库是作为供给驱动型的物理数据仓库的替代物而产生的。在虚拟数据仓库中，其自身并不实际集中存储大量的数据，而仅仅是由一层(或多层)用于将多种不同的数据系统连接在一起的中间件所构成的，这些中间件还负责将数据呈现给用户，通常是通过 Web 界面。这样，无论最终实现的是集中式的数据仓库，数据集市或者其他，虚拟数据仓库都能提供访问各个存储地点所存放的数据的方法。对用户来讲，并不需要去关心物理数据的实际存放位置，因而是“虚拟”了的。文[2]给出了虚拟数据仓库的特点，并解释了虚拟数据仓库与物理数据仓库的关系。

3. 关于元数据模型

在物理上分布的数据的集成过程中，元数据模型发挥着至关重要的作用，管理数据的关键主要来自于对元数据的管理。文[3]提供了一种用于分布式信息集成的模型驱动的体系结构，这种体系结构主要基于元数据模型。

目前，我们正在本所提出的基于查询优化的虚拟数据仓库模型的基础上实现面向 Web 的数据查询优化和处理方法。

参 考 文 献

1 李建中，高宏. 一种数据仓库的多维数据模型. 软件学报，2000，11(7)：908～917

2 Bischoff J. Data Warehouse - Practical Advice From The Experts. 1998. 196～202

3 A model-driven architecture for Distributed Information Integration. http: // www. omg. org

4 Inmon W H. Building the Data Warehouse Second Edition. 2000. 116～143

5 Zhang Yulong, Li Shanpinge. Web-based XML Document Management. In: Proc. of APWEb' 2000. Xi'an, China, 2000, 10

6 Lou Agosta. 数据仓库技术指南. 人民邮电出版社，2001