MD-SQL:一种基于 MDX 的多维数据查询语言*)

张佳文　乐嘉锦
(东华大学计算机科学与技术学院　上海 201620)

摘　要　多维数据分析技术在建立企业级数据仓库的过程中起着关键的作用,同时也是面向联机分析处理(OLAP)的核心。本文在分析了传统多维数据查询语言 MDX(Multi-Dimensional eXpression)不足的基础上,提出了一种改进的多维数据查询的语言模型 MD-SQL,并给予实现,最后对模型提出了优化方法,通过相关实验证明 MD-SQL 语言性能优于 MDX 语言。

关键词　多维数据分析,MDX,MD-SQL,OLAP

MD-SQL: A Query Language for Multi-dimensional Data Analysis Based on MDX

ZHANG Jia-Wen　LE Jia-Jin
(College of Computer Science and Technology, Donghua University, Shanghai 201620)

Abstract　As the core of OLAP, multi-dimensional data analysis plays an important role in the establishment of enterprise data warehouse. Based on the shortcomings of the traditional MDX, we propose a new SQL language model, MD-SQL, to support the OLAP computing and compare the traditional MDX with it. One kind of optimization is presented in the latter part. The experiment shows that MD-SQL is better than traditional MDX in terms of efficiency.

Keywords　Multi-dimensional data analysis, MDX, MD-SQL, OLAP

1　引言

近十年来,各数据库厂商为了使自己的产品拥有决策支持的功能,对各自的产品倾注了巨大的投资。Top N[1],Ranking[2],Grouping sets[3],Data cube[4]等都是已经开发出来的产品。决策支持基于联机分析处理(On Line Analytical Processing,缩写为 OLAP),而 OLAP 的核心是多维数据分析。多维数据分析是指对以多维方式组织起来的数据采取多种分析动作,进行数据分析,以便用户最终能从多个角度、多个侧面观察数据库中的数据,从而深入了解包含在数据中的信息和内涵。多维分析是分析企业数据最有效的方法,是 OLAP 的灵魂。遗憾的是,传统的 SQL 语言不能有效地支持多维视图、复杂分组及聚集操作。虽然微软开发的 SQL Server 2000 Analysis Services 系统中的 MDX(Multi-Dimensional eXpression)语言能够进行多维查询,但是 MDX 语法仍旧过于复杂,而且其他数据库系统中没有类似的 MDX 语言。因此,如何开发一种通用的多维数据查询语言以适应日趋成熟的 OLAP 系统,这个问题引起了广泛的关注[5]。

面向单元格的数据表示方法,比如电子表格,提供了友好的用户界面,用户可以输入业务数据,定义公式,旋转数据,计算给定单元格的数据集合等等;除此之外,电子表格还可以提供灵活的用户图形界面和满足不同需求的报表,更适合表示 OLAP 查询的结果[6]。

基于这种现状,本文在分析传统 MDX 针对 OLAP 应用不足的基础上,提出了一种基于电子表格的多维查询语言模型 MD-SQL,来支持多维数据查询。实验分析表明:该语言性能优于传统 MDX,更适合 OLAP 应用。本文的贡献如下:1)针对多维数据查询,提出传统 MDX 的不足之处;2)提出改进的 MD-SQL 语言模型,通过实验与 MDX 语言进行比较性分析。

2　MD-SQL 语言模型

2.1　用 MDX 语言进行多维数据查询

基本的 MDX 语句包含一个 SELECT 子句和一个 FROM 子句,以及一个可选的 WHERE 子句。SELECT 子句用来选择要返回的维度和成员,称之为轴维度。WHERE 子句用来将返回的数据限定为特定维度和成员条件,称之为切片维[7]。

MDX 的基本语法结构如下所示:

```
SELECT
[〈axis_specification〉[, 〈axis_specification〉…]]
FROM[〈cube_specification〉]
    [WHERE[〈slicer_specification〉]]
```

基本的 MDX 语句包含一个 SELECT 子句,一个 FROM 子句,以及一个可选的 WHERE 子句。其中,axis_specification 表示轴维度,cube_specification 表示多维数据源,slicer_specification 表示切片维度。SELECT 子句决定 MDX SELECT 语句的轴维度,FROM 子句决定当析取数据以填充 SELECT 语句的结果集时将使用哪个多维数据源,可选用的 WHERE 子句决定哪个维度或成员用作切片维度。

我们通过实例来讨论 MDX 的基本语法的各个部分。查询 1 基于数据立方实例 Sales(t, r, p, s, c),其中包含 3 个维:时间(t),地区(r)和产品(p)以及 2 个度量:销售额(s)和成本(c);对应的 3 个维表分别是:时间维表 Dimtime(TimeID, year, month, day),地区维表 Dimregion(RegionID, city)和产品维表 Dimproduct(ProductID, ProdutName)。

查询 1　查询 2000 年和 2001 年,ibm 和 hp 产品分别在

*)本项研究工作得到了上海市科委的资助,资助课题编号为 05DZ11C06。**张佳文**　硕士研究生,主要研究方向:数据库和数据仓库。

上海的销售量。

```
SELECT
{[Dimtime].[2000],[Dimtime].[2001]} ON COLUMNS,
{[Dimproduct].[ibm],[Dimproduct].[hp]} ON ROWS,
FROM Sales
WHERE ([Dimregion].[Shanghai]);
```

SELECT 子句定义 MDX 语句的轴维度。查询 1 中的 SELECT 子句定义了两个轴维度 COLUMNS，ROWS。通过成员、元组和集合来定义轴维度的内容。成员是在不同层次上取值的组合，如[Time].[2000]。MDX 允许用户指定 128 个轴，前 5 个轴分别用 COLUMNS，ROWS，PAGES，SECTIONS 和 CHAPTERS 表示，后面的轴用数字来表示，但一般情况下用户不会使用 5 个以上的轴。

FROM 子句决定当析取数据以填充结果集时将使用哪个多维数据源。在查询 1 中多维数据源是多维数据集 Sales。

WHERE 子句决定哪个维度及成员用作切片维度，将数据的析取限制于特定维度或成员。切片维度只接受可评估为单个元组的表达式：如果切片表达式提供元组集合，则 MDX 将尝试顺着集合聚合各个元组中的结果单元来评估集合。查询 1 根据地区维度对数据立方进行切片。

通过分析，我们发现 MDX 主要有两点不足：

(1)语法过于复杂，而且在报表支持方面仍有欠缺。

(2)由于稀疏性问题，用 MDX 查询的结果中会有很多空值，应该依据显示需要，在设计阶段把需要默认值的地方用默认值代替空值。

2.2 用 MD-SQL 语言进行多维数据查询

2.2.1 MD-SQL 语言的设计目标

为了弥补 MDX 的不足，MD-SQL 语言在 MDX 的语言基础上做了一些改进，下面列出它的四点改进目标。

(1)提供基本的查询分析功能。将数据立方划分成互不相交的子集，由用户指定层次进行上卷、下转分析，进行切片、切块分析。

(2)由用户指定结果显示方式，并将多维查询的结果处理到二维空间(电子表格)上进行显示，支持报表应用。

(3)语法格式简单易用，语义表达简洁清楚。

(4)解决稀疏性问题，避免空值的产生。

2.2.2 MD-SQL 基本语法结构

根据多维数据模型的特点，MD-SQL 语言将关系型的属性划分为维和度量，在查询中将各个字段分别定义为 PARTITION，DIMENSION 和 MEASURES，对应的缩写分别为 PBY，DBY 和 MEA。查询根据 PBY 中的字段将关系划分为不相交的分组；定义为 DBY 的字段用来在每个分组中唯一确定一行，即用作度量字段的数组下标；定义为 MEA 的字段确定要通过公式计算的表达式。MODEL 子句中有一系列公式，每个公式描述了在 DBY 指定的数组单元格上的一个计算。MD-SQL 语法的巴科斯范式如下所示：

```
MD-SQL Statement::=
"SELECT"〈column〉{[,column…]} "ON" "PBY" ","
〈column〉{[,column…]} "ON" "DBY" ","
〈column〉{[,column…]} "ON" "MEA"
"FROM" 〈FactTable〉
"MODEL"
"("
    〈fomula〉[,〈fomula〉… ] ";"
")"
```

其中，SELECT 子句用来选择要返回的维度和成员，选定的字段用 ON 关键字来表示，定义为 PBY，DBY 或 MEA。如果 SELECT 子句中未指定任何度量值，则以默认的形式返回数据立方定义中度量值列表中的第一个度量值。

MD-SQL 查询语言可以分成两个部分：SELECT 语句部分(不含 MODEL 子句)和 MODEL 子句部分。MODEL 子句在 SELECT 语句之后，操作的对象是 SELECT 语句的执行结果集。

我们仍旧使用 2.1 中的销售数据立方作为范例，查询 1 用 MD-SQL 语言表示如下：

```
SELECT r ON PBY,
       p, t ON DBY,
       s ON MEA
FROM Sales
MODEL(s['ibm',2000], s['ibm',2001];)
```

上述查询首先根据地区 r 将数据划分成互不相交的分区，然后以产品 p 和时间 t 为维度，销量 s 为度量进行查询。其中，s['ibm',2000]表示 2000 年 ibm 台式机的销量。与 2.1 节中的 MDX 语言相比，MD-SQL 语言不仅形式简洁，而且语义清晰，通过 MODEL 子句提高了查询的效率。

查询 2：将 Sales 事实表按地区 r 进行划分，2002 年 ibm 的销售量是 2000 年 ibm 的销售量和 2001 年 ibm 的销售量的总和。2002 年 hp 的销售量是 hp 从 1993 年到 2001 年销售量的平均值。

```
SELECT r ON PBY,
       p, t ON DBY,
       s ON MEA
FROM Sales
MODEL
(s['ibm',2002]=s['ibm',2000]+s['ibm',2001],
s['hp', 2002]=avg(s)['hp',1992<t<2002];
)
```

查询 2 中，avg(s)['hp',1992<t<2002]表示 1992 年到 2002 年 hp 台式机销量的平均值。

由于 MD-SQL 首先根据 PBY 中的字段进行划分，减少了聚集次数，而且查询结果显示在电子表格中，减少了 NULL 值的产生，缓解了数据稀疏性问题。

2.3 MD-SQL 编译器的设计与实现

MD-SQL 编译器用来对 MD-SQL 语句进行语法分析和语义转换，实现从多维查询语句到关系代数表达式的转化。MD-SQL 编译器由以下两部分组成：语法分析器和转换处理器，如图 1 所示。语法分析器负责将一条 MD-SQL 语句转化为语法树的结构。转换处理器对语法树进行语义检查，并转化为关系型的预处理结构。我们采用 Visual Parse++编译器生成工具来实现词法分析和语法分析。语法分析部分检查单词序列是否符合 MD-SQL 的语法规范，同时生成一棵语法树，称为 MD-SQL 语法树，如图 2 所示。从语法分析器得到 MD-SQL 语法树后，转换处理器将进行从语法树到关系代数表达式的转换。

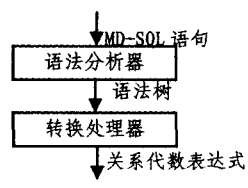

图 1 MD-SQL 编译器的结构图

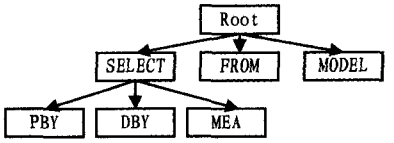

图 2　MD-SQL 语法树

3　MD-SQL 语言优化

作为 MDX 的改进模型，MD-SQL 不但支持原来 MDX 的功能特性，而且能够以电子表格的形式显示查询结果，为报表计算也提供了接口。为了获得更好的查询效率，本文接下来对 MD-SQL 语言模型进行优化。

MD-SQL 的 MODEL 子句中通常包含一系列公式，当这些公式表示的形式相同时，我们考虑将其按以下方式优化。假设查询中的 MODEL 子句描述如下：

```
MODEL
(x[1, 1] = x[1, 2]+x[1, 3]+x[1, 4],
x[2, 1] = x[2, 2]+x[2, 3]+x[2, 4],
...
x[n, 1]=x[n, 2]+x[n, 3]+x[n, 4]);
```

该规则表示一共有 n 行数据，每行的第一列的值等于第二列的值，第三列的值和第四列的值的总和，为了提高效率，无需写 n 句赋值语句，只需要用一个表达式表达就可以了。有两种表示方法，如表 1 所示。

表 1　公式优化

for 循环	表达式
MODEL (x[for row from 1 to n,1]=x [CV(row), 2]+x[CV(row), 3]+x[CV(row), 4]);	MODEL (x[1<=row<=n,1]=x[CV (row),2]+x[CV(row), 3]+x [CV(row),4]);

4　实验与性能分析

为了验证 SQL 扩展模型的性能，我们构造了一系列实验进行测试。实验环境是一台 Intel Pentium Ⅲ 1.7GHz 处理器和 512M 内存的 PC 机，软件环境为 Windows XP Professional SP2，Oracle 9i。

4.1　MDX 查询和 MD-SQL 查询的性能对比

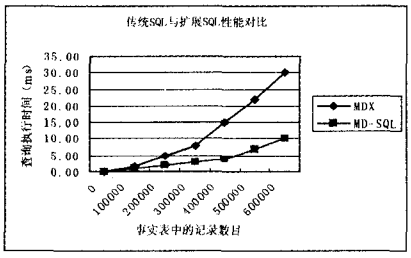

图 3　MDX 查询和 MD-SQL 查询的性能对比图

实验比较用传统 MDX 查询和用 MD-SQL 查询的性能，得到的数据如图 3 所示。其中事实表记录数分别为 10,000，20,000,30，000,40,000,50,000 和 60,000。传统 MDX 查询所花费的时间明显比 MD-SQL 模型查询要多，使用 MD-SQL 模型性能可以提高约 3 倍，并且随着事实表记录数的增加，该模型性能提高得更多。

4.2　公式优化性能测试

实验比较进行公式优化前后的查询性能，事实表中的数据量为 100000 行，得到的数据如图 4 所示。但公式数目在 10 以内时，优化前和优化后的性能差不多，但随着公式数目增加，优化后的优势逐渐体现出来，没有经过优化的查询执行时间几乎呈线性增长。

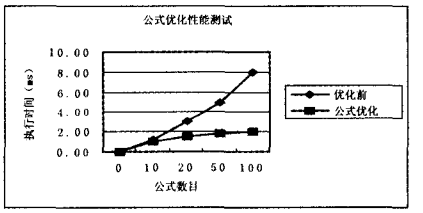

图 4　公式优化性能测试

结束语　由于传统多维数据查询语言 MDX 在支持 OLAP 查询方面存在一定的不足，很多商业智能软件厂商投入大量的资金开发各自的 OLAP 引擎来支持数据库产品。我们研究的出发点来自于传统多维数据查询语言针对 OLAP 应用的不足，本文的创新在于对现有 MDX 进行了改进，提出了一种基于 MDX 的 MD-SQL 语言模型并给予了实现；该语言不但与主流数据库兼容，而且与 MDX 相比，能更好地支持 OLAP 应用；实验结果表明在性能上优于传统 MDX，提高了 OLAP 查询的效率。

参 考 文 献

1　Carey M L, Kossmann D. Processing Top N and Bottom N Queries [C]. In: Proc. of IEEE Data Engineering Bulletin, 1997, 20 (3): 12～19

2　Geerts F, Mannila H, Terzi E. Relational Link-based Ranking [C]. In: Proc. of VLDB, Toronto, Canada, August 2004. 552～563

3　ISO/IEC. SQL 1999. 9075-1:1999

4　Gray J, Bosworth A, Layman A, Pirahesh H. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total [C]. In: Proc. of IEEE ICDE, New Orleans, Louisiana, February 1996. 152～159

5　Niemi T, Hirvonen L, Jarvelin K. Multidimensional Data Model and Query Language for Informetrics [J]. Journal of the American Society for Information Science and Technology, 2003, 54 (10): 939～951

6　Witkowski A, et al. Spreadsheets in RDBMS for OLAP [C]. In: Proc. of ACM SIGMOD, San Diego, June 2003. 52～63

7　曹忠升，黄宇殊，等. 多维查询语言 DM_MDX 编译器的设计与实现[J]. 计算机工程与应用，2004，40(9)：109～110