[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 17/30 (2006.01)

[12] 发明专利说明书

专利号 ZL 200380106455.1

[45] 授权公告日 2008 年 10 月 8 日

[11] 授权公告号 CN 100424686C

[22] 申请日 2003.12.17
[21] 申请号 200380106455.1
[30] 优先权
[32] 2003. 1.13 [33] US [31] 10/341,763
[86] 国际申请 PCT/GB2003/005490 2003.12.17
[87] 国际公布 WO2004/063942 英 2004.7.29
[85] 进入国家阶段日期 2005.6.17
[73] 专利权人 国际商业机器公司
地址 美国纽约
[72] 发明人 内森·格瓦尔德·克洛斯
威廉·厄尔利·玛洛伊
米亚·哈米德·匹拉赫史
克雷格·雷格纳尔德·托姆林
[56] 参考文献
US 5918232 A 1999.6.29
WO 01/09768 A 2001.2.8
US 2002/095430 A1 2002.7.18
CN 1347529 A 2002.5.1
An Integrative and Uniform Model for Metadatamanagementin Data Warehousing Environments. STOHR T ET AL. PROCEEDINGS OF THE INTERNATIONAL WORKSHOP ON DESIGN AND MANAGEMENT OF DATA WAREHOUSES DMDW99. 1999
审查员 张 雯
[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 康建峰

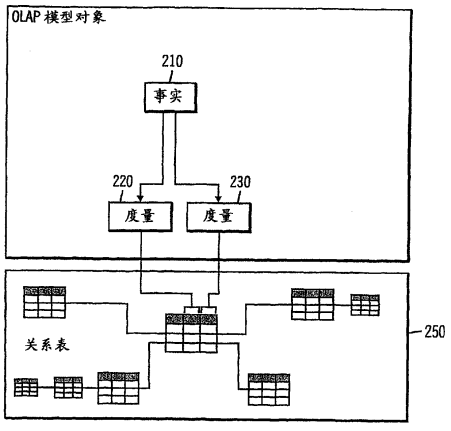
权利要求书 2 页 说明书 64 页 附图 42 页

[54] 发明名称

规定用于关系 OLAP 引擎的多维计算

[57] 摘要

本发明公开了一种用于规定多维计算的系统、方法和程序。 接收从事实元数据对象和一个或多个维元数据对象生成的立方体模型元数据对象的子集的选择。 事实元数据对象引用一个或多个度量元数据对象。 使用立方体模型元数据对象和度量元数据对象中的元数据生成用于检索多维信息的语句，其中度量元数据对象的每一个规定一个或多个聚合。



1. 一种用于自动生成单个结构化查询语言语句的计算机实现方法，其规定涉及对称或非对称度量集的多维计算，其中对称度量具有单个聚合操作符，并且没有特定聚合次序，而非对称度量具有多个聚合操作符，该方法包括以下步骤：

接收从事实元数据对象和一个或多个维元数据对象生成的立方体模型元数据对象的子集的选择，其中事实元数据对象引用一个或多个度量元数据对象；以及

使用立方体模型元数据对象和该一个或多个度量元数据对象中的元数据生成用于检索多维信息的语句，其中度量元数据对象的每一个规定一个或多个聚合，

其中所述生成用于检索多维信息的语句的步骤包括：

分离在该一个或多个度量元数据对象中定义的对称度量和非对称度量；

为对称度量生成结构化查询语言语句；

为非对称度量生成结构化查询语言语句；以及

将对称和非对称度量的结构化查询语言语句组合成用于检索多维信息的单个结构化查询语言语句。

2. 如权利要求 1 所述的方法，其中组合包括使用连接。

3. 一种用于自动生成单个结构化查询语言语句的设备，其规定涉及对称或非对称度量集的多维计算，其中对称度量具有单个聚合操作符，并且没有特定聚合次序，而非对称度量具有多个聚合操作符，该设备包括：

用于接收从事实元数据对象和一个或多个维元数据对象生成的立方体模型元数据对象的子集的选择的装置，其中事实元数据对象引用一个或多个度量元数据对象；以及

用于使用立方体模型元数据对象和该一个或多个度量元数据对象中的元数据生成用于检索多维信息的语句的装置，其中度量元数据对象的

每一个规定一个或多个聚合，

其中，所述用于生成用于检索多维信息的语句的装置包括：

用于分离在该一个或多个度量元数据对象中定义的对称度量和非对称度量的装置；

用于为对称度量生成结构化查询语言语句的装置；

用于为非对称度量生成结构化查询语言语句的装置；以及

用于将对称和非对称度量的结构化查询语言语句组合成用于检索多维信息的单个结构化查询语言语句的装置。

规定用于关系 OLAP 引擎的多维计算

技术领域

本发明涉及规定用于关系(relational)在线分析处理(on-line analytical processing，OLAP)引擎的多维计算。

背景技术

在线分析处理(OLAP)日益变得普及。代替查阅成堆的打印在绿栏纸上的静态报告，OLAP 分析者可以交互地探查商业结果，动态地调整数据视图和询问问题，并且几乎即时地得到答案。从静态答案到固定时间表上的固定问题的这一自由允许商业分析者更有效地工作并且实现商业操作上的改善。

Nigel Pendse 引入了术语“FASMI”来表现 OLAP 系统的特征。FASMI 特征是：快速、分析、共享、多维和信息。关于进一步的信息，参见 N. Pendse, “What is OLAP?”,, The OLAP Report, http://www.olapreport.com/fasmi.htm。

关于快速，与 OLAP 中“O”的精神保持一致，该系统需要非常迅速地提供结果，这通常是在仅仅几秒内，并且很少超过 20 或 30 秒。这一性能级别对于允许分析者有效地工作而不会分心是关键的。

关于分析，考虑 OLAP 中的“A”，OLAP 系统通常以最小量的编程来提供适于给定应用的丰富分析功能。

关于共享，OLAP 系统通常是共享资源。这意味着要求 OLAP 系统提供适当的安全性和完整性特性。最终，这可意味着对数据库的不同单元提供不同的访问控制。

关于多维，多维性是 OLAP 系统的主要要求。OLAP 产品在多维框架中呈现其数据。维是该系统的数据值的相关标识符或属性(attribute)(例如，产品、市场、时间、渠道、场景或客户)的集合。属于

特定维的此集合的标识符(例如，“The Lord of the Rings-DVD”，“San Jose, California”,“2002”,“Retail Rental” 和“John Q. Public”)通常具有某种结构，例如分级结构。有时，对于这些标识符，存在多于一个自然结构。

多维特征意味着 OLAP 系统可以在各个维方向之间以及在一维的各个子集和结构排列之间快速地切换。由于 OLAP 系统的多维性质，它们所实现的数据集合被称作立方体(cube)。关于信息，OLAP 系统存储并计算信息。OLAP 系统的数据经常来自一个或多个操作系统。对这些数据应用分析模型，并且在该系统中存储或者在查询时生成结果。特定 OLAP 系统可管理的信息量是该系统的一个特征。

很多年来，企业一直是使用星形或雪花模式(schema)在关系数据库中存储多维数据。随着时间的过去，关系数据库厂商对这些模式增加了增强查询性能的优化。在二十世纪九十年代期间，开发了很多专用数据库，其可以处理增加的计算复杂度，并且其性能通常好于关系引擎的性能。

多维 OLAP(MOLAP)是指这样的 OLAP 系统系列，其中使用专用文件系统或索引来存储立方体数据。Express Web Publisher, Essbase™, TM1 和 Pilot Suite 是基于专用存储和索引建立技术的产品的少数例子。Microsoft 的 OLAP 出售品也包括 MOLAP 引擎。这些系统经常是只读系统，其周期性地被装载基本数据，然后对派生结果进行计算、存储和索引建立。MOLAP 系统的可伸缩性经常受限于在其内计算和存储派生结果的批窗口的大小。为了改善可伸缩性，该系统经常具有用于将一些派生结果的计算推迟到查询时候的手段。

对于关系 OLAP(ROLAP)，很多年来都是使用星形模式作为用于在关系数据库中表示多维数据的手段。很多商业软件开发公司，例如 MicroStrategy, Brio, Business Objects, Metacube, Hyperion 和 Metaphor 为关系星形模式开发了批方式或交互式多维报告和探查接口。这些系统全都是在将超级聚合(aggregate)操作符添加到结构化查询语言(SQL)语言定义之前设计和实现的。

具体地说，直到几年前为止，关系数据库对于每个查询仅允许在单个级别上进行聚合计算。例如，一个带 GROUP BY 子句的 SELECT 语句将用来在季度的级别上(即，针对季度的集合)检索结果集，而另一个带 GROUP BY 子句的 SELECT 语句将用来在月的级别上(即针对月的集合)检索结果集。这迫使关系 OLAP 系统对数据库运行多个查询，以便在不同的级别上计算单元。

为了帮助 OLAP 类型查询的创建，并且提供更高级的优化，可从国际商业机器公司获得的 DB2®关系数据库管理系统(RDBMS)实现了三个新的超级聚合操作符：ROLLUP、CUBE 和 GROUPING SETS，其被添加到 SQL 标准，以便允许单个查询生成多个聚合。这些超级聚合操作符是对 GROUP BY 子句的扩展，并且规定在多个级别上生成聚合。例如，一个 SELECT 语句可用来获得多个级别(例如，季度和月)上的聚合计算的结果集。

注意，这些超级聚合操作符不仅仅是用于生成多个编组集合(grouping set)的简化表示。由于在单个语句中请求多个编组集合，因此 DB2® RDBMS 可以构建执行规划，其中该执行规划以对于计算所需的每个输入行仅被引用一次的方式生成所有编组集合。这可导致多个数量级的性能改善，尤其是当输入行集合未装入(fit in)缓冲池(即高速缓存)时，更是如此。

现有技术的系统被设计成通过发出多个查询来产生表示具有不同粒度级别的结果的多维报告。为该多个查询获得多个结果集，并且合并这些结果集来形成单个报告。这样的系统取决于对星形模式中表和列的角色的某种描述(元数据)，其用于生成必要的 SQL 语句来检索数据以便产生多维报告。确切的元数据随着产品而不同。

多维在线分析处理(OLAP)系统(例如，出自诸如 Hyperion, Cognos 和 Microsoft 的公司)被设计成，当被提供多维立方体每条边的成员集时，自然地返回多维结果集。该多维 OLAP 系统还被设计成预先计算任何查询的一些或全部结果。

自从引入了关系数据库以来一直是使用 SQL 进行多维分析，但是

关系 OLAP 系统不能自然地返回多维结果集，或者预先计算查询的一些或全部结果。

因此，在本技术领域内需要一种改进的关系 OLAP 系统。

发明内容

从而，在第一方面，本发明提供了一种用于规定多维计算的方法，包括：接收从事实(facts)元数据对象和一个或多个维(dimension)元数据对象生成的立方体模型元数据对象的子集的选择，其中事实元数据对象引用一个或多个度量(measure)元数据对象；以及使用立方体模型元数据对象和该一个或多个度量元数据对象中的元数据生成用于检索多维信息的语句，其中度量元数据对象的每一个规定一个或多个聚合，其中所述生成用于检索多维信息的语句的步骤包括：分离在该一个或多个度量元数据对象中定义的对称度量和非对称度量；为对称度量生成结构化查询语言语句；为非对称度量生成结构化查询语言语句；以及将对称和非对称度量的结构化查询语言语句组合成用于检索多维信息的单个结构化查询语言语句。

本发明还提供了一种用于自动生成单个结构化查询语言语句的设备，其规定涉及对称或非对称度量集的多维计算，其中对称度量具有单个聚合操作符，并且没有特定聚合次序，而非对称度量具有多个聚合操作符，该设备包括：用于接收从事实元数据对象和一个或多个维元数据对象生成的立方体模型元数据对象的子集的选择的装置，其中事实元数据对象引用一个或多个度量元数据对象；以及用于使用立方体模型元数据对象和该一个或多个度量元数据对象中的元数据生成用于检索多维信息的语句的装置，其中度量元数据对象的每一个规定一个或多个聚合，其中，所述用于生成用于检索多维信息的语句的装置包括：用于分离在该一个或多个度量元数据对象中定义的对称度量和非对称度量的装置；用于为对称度量生成结构化查询语言语句的装置；用于为非对称度量生成结构化查询语言语句的装置；以及用于将对称和非对称度量的结构化查询语言语句组合成用于检索多维信息的单个结构化查询语言语句的装置。

优选地，该语句是结构化查询语言语句。

优选地，度量元数据对象的每一个规定一个或多个结构化查询语言表达式。

优选地，结构化查询语言表达式的每一个包括用于构建查询语言表达式的模板。

优选地，该模板使用从列、属性和度量的列表中引用特定列、属性或度量的标记表示法(token notation)。

优选地，结构化查询语言表达式的每一个包括列、属性和度量的列表。

优选地，基于每一个度量元数据对象中所规定的一个或多个聚合而生成结构化查询语言语句。

优选地，聚合列表包括聚合函数和对应的维集的列表。

优选地，维集为对应的聚合函数规定 NULL，以便包括除了聚合列表内的另一个聚合函数中所规定的维之外的所有其它可用维。

优选地，度量元数据对象规定一个或多个结构化查询语言表达式，并且其中使用结构化查询语言表达式作为对聚合列表中的聚合的输入。

优选地，使用分级结构元数据对象中的元数据生成语句来构建 ROLLUP 子句。

优选地，度量元数据对象引用另一个度量元数据对象。

优选地，生成结构化查询语言语句还包括：生成用于总计(grand total)查询的 SELECT 语句。

优选地，生成结构化查询语言语句还包括：生成用于立方体模型元数据对象的子集切片(slice)的 SELECT 语句。

优选地，生成结构化查询语言语句还包括：生成用于立方体模型元数据对象的子集的 SELECT 语句。

优选地，该方法还包括：分离在该一个或多个度量元数据对象中定义的对称度量和非对称度量；为对称度量生成结构化查询语言语句；为非对称度量生成结构化查询语言语句；以及将对称和非对称度量的结构化查询语言语句组合成单个结构化查询语言语句。

优选地，该组合包括使用连接(join)。

优选地，该方法还包括：确定度量是否与一个或多个度量兼容；并且如果度量与一个或多个度量不兼容，则确定是否可以改写这些度量的任一个；并且如果可以改写这些度量的任一个，则改写这些度量。

优选地，该方法还包括：基于立方体模型元数据对象的子集的选择而生成立方体元数据对象，包括生成用于创建立方体视图的结构化查询语言语句，其中从该一个或多个度量元数据对象中的元数据生成结构化查询语言语句。

优选地，该方法还包括：在应用程序的控制之下，使用立方体模型元数据对象和一个或多个度量元数据对象，生成结构化查询语言语句来检索多维信息。

在第二方面，本发明提供了一种用于规定多维计算的系统，包括具有至少一个可编程组件以运行至少一个用于以下操作的程序的计算机系统：接收从事实元数据对象和一个或多个维元数据对象生成的立方体模型元数据对象的子集的选择，其中事实元数据对象引用一个或多个度量元数据对象；以及使用立方体模型元数据对象和该一个或多个度量元数据对象中的元数据生成用于检索多维信息的语句，其中度量元数据对象的每一个规定一个或多个聚合。

优选地，该语句是结构化查询语言语句。

优选地，度量元数据对象的每一个规定一个或多个结构化查询语言表达式。

优选地，结构化查询语言表达式的每一个包括用于构建查询语言表达式的模板。

优选地，该模板使用从列、属性和度量的列表中引用特定列、属性或度量的标记表示法。

优选地，结构化查询语言表达式的每一个包括列、属性和度量的列表。

优选地，基于每一个度量元数据对象中所规定的一个或多个聚合而生成结构化查询语言语句。

优选地，聚合列表包括聚合函数和对应的维集的列表。

优选地，维集为对应的聚合函数规定 NULL，以便包括除了聚合列表内的另一个聚合函数中所规定的维之外的所有其它可用维。

优选地，度量元数据对象规定一个或多个结构化查询语言表达式，并且其中使用结构化查询语言表达式作为对聚合列表中的聚合的输入。

优选地，使用分级结构元数据对象中的元数据生成语句来构建 ROLLUP 子句。

优选地，度量元数据对象引用另一个度量元数据对象。

优选地，该至少一个程序还包括：生成用于总计查询的 SELECT 语句。

优选地，该至少一个程序还包括：生成用于立方体模型元数据对象的子集的 SELECT 语句。

优选地，该至少一个程序还包括：生成用于立方体模型元数据对象的子集的 SELECT 语句。

优选地，该至少一个程序还包括：分离在该一个或多个度量元数据对象中定义的对称度量和非对称度量；为对称度量生成结构化查询语言语句；为非对称度量生成结构化查询语言语句；以及将对称和非对称度量的结构化查询语言语句组合成单个结构化查询语言语句。

优选地，该组合包括使用连接。

优选地，该至少一个程序还包括：确定度量是否与一个或多个度量兼容；并且如果度量与一个或多个度量不兼容，则确定是否可以改写这些度量的任一个；并且如果可以改写这些度量的任一个，则改写这些度量。

优选地，该至少一个程序还包括：基于立方体模型元数据对象的子集的选择而生成立方体元数据对象，包括生成用于创建立方体视图的结构化查询语言语句，其中从该一个或多个度量元数据对象中的元数据生成结构化查询语言语句。

优选地，该至少一个程序还包括：在应用程序的控制之下，使用立方体模型元数据对象和一个或多个度量元数据对象，生成结构化查询语言语句来检索多维信息。

在第三方面，本发明提供了一种计算机程序，其包括当被装载到计算机系统中并且在其上执行时使该计算机系统执行根据第一方面的方法的所有步骤的计算机程序代码。

该计算机程序可被实施在包括用于规定多维计算的程序的产品中，其中该程序导致执行多个操作，这些操作包括：接收从事实元数据对象和一个或多个维元数据对象生成的立方体模型元数据对象的子集的选择，其中事实元数据对象引用一个或多个度量元数据对象；以及使用立方体模型元数据对象和该一个或多个度量元数据对象中的元数据生成用于检索多维信息的语句，其中度量元数据对象的每一个规定一个或多个聚合。

优选地，该语句是结构化查询语言语句。

优选地，度量元数据对象的每一个规定一个或多个结构化查询语言表达式。

优选地，结构化查询语言表达式的每一个包括用于构建查询语言表达式的模板。

优选地，该模板使用从列、属性和度量的列表中引用特定列、属性或度量的标记表示法。

优选地，结构化查询语言表达式的每一个包括列、属性和度量的列表。

优选地，基于每一个度量元数据对象中所规定的一个或多个聚合而生成结构化查询语言语句。

优选地，聚合列表包括聚合函数和对应的维集的列表。

优选地，维集为对应的聚合函数规定 NULL，以便包括除了聚合列表内的另一个聚合函数中所规定的维之外的所有其它可用维。

优选地，度量元数据对象规定一个或多个结构化查询语言表达式，并且其中使用结构化查询语言表达式作为对聚合列表中的聚合的输入。

优选地，使用分级结构元数据对象中的元数据生成语句来构建 ROLLUP 子句。

优选地，度量元数据对象引用另一个度量元数据对象。

优选地，用于生成结构化查询语言语句的操作还包括：生成用于总计查询的 SELECT 语句。

优选地，用于生成结构化查询语言语句的操作还包括：生成用于立方体模型元数据对象的子集切片的 SELECT 语句。

优选地，用于生成结构化查询语言语句的操作还包括：生成用于立方体模型元数据对象的子集的 SELECT 语句。

优选地，这些操作还包括：分离在该一个或多个度量元数据对象中定义的对称度量和非对称度量；为对称度量生成结构化查询语言语句；为非对称度量生成结构化查询语言语句；以及将对称和非对称度量的结构化查询语言语句组合成单个结构化查询语言语句。

优选地，该组合包括使用连接。

优选地，这些操作还包括：确定度量是否与一个或多个度量兼容；并且如果度量与一个或多个度量不兼容，则确定是否可以改写这些度量的任一个；并且如果可以改写这些度量的任一个，则改写这些度量。

优选地，这些操作还包括：基于立方体模型元数据对象的子集的选择而生成立方体元数据对象，包括生成用于创建立方体视图的结构化查询语言语句，其中从该一个或多个度量元数据对象中的元数据生成结构化查询语言语句。

优选地，这些操作还包括：在应用程序的控制之下，使用立方体模型元数据对象和一个或多个度量元数据对象，生成结构化查询语言语句来检索多维信息。

这样，优选地，提供了一种用于规定多维计算的方法、系统和程序。接收从事实元数据对象和一个或多个维元数据对象生成的立方体模型元数据对象的子集的选择。事实元数据对象引用一个或多个度量元数据对象。使用立方体模型元数据对象和度量元数据对象中的元数据生成用于检索多维信息的语句，其中度量元数据对象的每一个规定一个或多个聚合。

本发明的所述实现提供了一种在关系 OLAP 系统中规定多维计算的方法、系统和程序。

附图说明

现在参照附图，其中相同的附图标记始终表示对应的部件：

图 1 以方框图的形式示出了根据本发明特定实现的计算环境。

图 2 示出了根据本发明的特定实现事实元数据对象和度量元数据对象与关系数据相关。

图 3 示出了根据本发明特定实现的样例星形连接模式。

图 4 示出了根据本发明的特定实现从关系表构建维元数据对象。

图 5 示出了根据本发明的特定实现，元数据对象一起装入(fit in)立方体模型，并且映射到关系表的关系星形模式。

图 6 示出了根据本发明的特定实现将概念元数据对象分类成三层。

图 7 示出了根据本发明的特定实现创建与基本/关系层对应的元数据对象。

图 8 示出了根据本发明特定实现的来自基本/关系层的附加元数据对象。

图 9 示出了根据本发明特定实现的基于星形连接模式而创建的多维层元数据对象。

图 10 示出了根据本发明特定实现的用来定义立方体的元数据对象的实例。

图 11 示出了根据本发明的特定实现创建在线分析处理(OLAP)层中每个元数据对象的一个实例。

图 12 示出了根据本发明特定实现的平衡分级结构的例子。

图 13 示出了根据本发明特定实现的非平衡分级结构的例子。

图 14 示出了根据本发明特定实现的不规则(ragged)分级结构。

图 15 示出了根据本发明特定实现的网络分级结构。

图 16 示出了根据本发明特定实现的一些元数据对象之间的一些关系。

图 17 示出了根据本发明特定实现的由两个维表和一个事实表组成的星形模式。

图 18A-18E 示出了根据本发明特定实现的可以针对星形模式生成的元数据对象实例的可能集合以及元数据对象的一些属性(property)。

图 19 示出了根据本发明特定实现的表 A，其表示基本数据。

图 20 示出了根据本发明特定实现的表 B，其表示具有聚合:SUM(Market)和 Min(Time)的度量。

图 21A-21D 示出了根据本发明特定实现的表 C，其表示具有聚合:SUM(Product)(即产品之和)、AVG(Time)(即时间上的平均值)和 Max(Market)(即市场的最大值)的度量。

图 22 示出了根据本发明特定实现的两个全加性度量元数据对象的创建。

图 23 示出了根据本发明特定实现的半加性度量的创建。

图 24 示出了根据本发明特定实现的具有聚合的复合度量的创建。

图 25 示出了根据本发明特定实现的没有聚合的复合度量的创建。

图 26 示出了根据本发明特定实现的具有 OLAP 函数的度量的创建。

图 27 示出了根据本发明特定实现的具有聚合和多个输入的度量。

图 28 示出了根据本发明一些实现的来自图 22-27 的所有定义的度量元数据对象。

图 29A、29B、29C、29D 和 29E 示出了根据本发明特定实现的用于为一个或多个度量元数据对象生成 SQL 语句的逻辑。

图 30 示出了根据本发明特定实现的表 D，其列出一些度量并且表示哪些度量是对称的或者非对称的。

图 31 示出了根据本发明特定实现的表 E，其列出一些聚合函数并且表示哪些聚合函数是分布式的(distributive)而哪些是非分布式的(non-distributive)。

图 32 示出了根据本发明特定实现的表 F，其列出多个度量，以及如何可将聚合步骤分解成用于这些度量的多个聚合步骤。

图 33 示出了计算机系统体系结构的一种实现。

具体实施方式

在下面描述中，参考形成本文一部分且示出了本发明的若干实现的附图。应当理解，在不脱离本发明的范围的情况下，可以采用其它实现并且可以进行结构和操作上的改变。

A. 多维元数据介绍

在特定实现中，本发明提供了多维元数据对象以及用于使用多维元数据对象的技术。为便于引用起见，本发明的优选实施例在此将被称作“OLAP 多维元数据系统 100”，并且度量元数据对象将被称作“元数据对象”。

在特定实现中，OLAP 多维元数据系统 100 在从国际商业机器公司获得的 DB2®通用数据库(UDB) RDBMS 中实现。虽然本说明书描述了使用 IBM 的 DB2® UDB RDBMS 软件，但是本领域的技术人员应当认识到本发明可使用其它 RDBMS 软件，例如从 Oracle、IBM Informix、Sybase 获得的 RDBMS 软件。另外，本发明可以在使用各种操作系统例如 IBM z/OS®、IBM AIX®、Microsoft Windows® 2000、Microsoft Windows® XP、Linux、Solaris、HP-UX 的计算机上运行。

图 1 以方框图的形式示出了根据本发明特定实现的计算环境。关系数据库管理系统(RDBMS)110 包括多维元数据软件 120(例如，存储过程应用编程接口(API))和用户接口 150。RDBMS 110 访问多维元数据对象 130 和关系数据库 140。在特定实现中，多维元数据对象 130 和关系数据库 140 中的数据可以被存储在单个数据库中。

OLAP 多维元数据系统 100 包括多维元数据软件 120(例如，存储过程应用编程接口(API))、用户接口 150 和多维元数据对象 130。多维元数据软件 120 用来创建、存储和访问多维元数据对象 130。可选地，可以向用户或管理员提供用户接口 150，以向多维元数据软件 120 发送命令。用户可以通过经由用户接口 150 提交命令来创建、访问、修改或删除多维元数据对象 130。由多维元数据软件 120 接收并处理这些命令。例如，多维元数据软件 120 可以创建并存储多维元数据对象 130。

在特定实现中，OLAP 多维元数据系统 100 为 RDBMS 110 如

DB2®通用数据库(在此被称作 DB2® UDB)提供附加特性，其改善 RDBMS 110 执行 OLAP 处理的能力。本发明的优选实施例以流水线的方式进行(streamline)OLAP 解决方案的部署和管理，并且改善 OLAP 工具和应用的性能。

具体地说，OLAP 多维元数据系统 100 提供元数据对象。新的元数据对象被存储在例如数据库目录(例如，DB2® UDB 目录)中，其中该数据库目录描述现有关系数据的维模型和 OLAP 构造。数据库目录提供 OLAP 应用可以从其捕捉(capture)多维元数据的单个知识库(repository)。在特定实现中，元数据对象可以驻留在不同于数据库目录的数据存储库上，或者可以驻留在多个数据存储库之间。通过中央知识库中的信息，数据库优化器能够使用特定于星形模式的技术，以便优化查询的执行。

通过元数据对象，本发明可以通过在汇总表中聚合数据并且创建索引来优化 OLAP 查询性能。OLAP 多维元数据系统 100 还提供元数据编程接口。具体地说，OLAP 多维元数据系统 100 为 OLAP 工具和应用开发者提供基于 SQL 和基于可扩展标记语言(XML)的应用编程接口(API)。例如通过命令行接口(CLI)、开放数据库连接性(ODBC)、或 Java 数据库连接性(JDBC™)连接，或者例如通过使用 DB2® UDB 的嵌入 SQL，应用和工具可使用单个存储过程(即，多维元数据软件 120 的例子)来创建、修改和检索元数据对象。在特定实现中，多个存储过程可以提供用于创建、修改和检索元数据对象的功能性。

OLAP 多维元数据系统 100 的元数据对象描述作为智能 OLAP 结构的关系信息，但是由本发明提供的多维元数据对象不同于传统的 OLAP 对象。本发明的元数据对象存储元数据，这意味着元数据对象存储关于基本表中的数据的信息。元数据对象描述有关数据位于何处，并且还能描述基本数据内的关系。例如，事实元数据对象是存储关于相关度量、属性和连接的信息的特定元数据对象，但是不包括特定地来自基本事实表的数据。

元数据提供从其理解数据的新角度。在没有元数据对象的情况下，

数据库目录仅仅知道表和列名称，并且不能存储关于这些表和列的意义或者这些表和列如何彼此相关的信息。通过元数据对象，可以存储该信息。

每个元数据对象完成表示关系数据的意义是什么的总画面(the big picture)的一个片断。一些元数据对象担当通过对数据进行聚合或者直接对应于关系表中的特定列来直接访问关系数据的基础。其它元数据对象描述基本元数据对象之间的关系，并且将这些基本元数据对象链接在一起。最终，所有元数据对象都可以通过彼此之间的关系而被一起编组到被称作立方体模型的元数据对象中。立方体模型表示关系表的特定编组和配置。立方体模型的目的是向给定应用或工具描述 OLAP 结构。立方体模型往往针对正被分析的数据描述不同用户可能想要的所有立方体。立方体模型对维和事实进行编组，并且为维提供多个分级结构的灵活性。立方体模型传达生成对星形模式数据库的复杂查询的查询设计工具和应用所需的结构信息。

多维元数据对象模型被设计成描述在关系数据库中用来表示多维数据的模式。组织这样的数据的一种方法是通过使用星形或雪花模式(在雪花模式中，维表被规格化)。然而，该模型灵活得足以处理任何类型的模式(例如，更加规格化的模式)。

A.1 多维元数据概述

多维元数据使得能够维护关于存储在数据仓库中的 OLAP 结构的信息。该信息先前不能在数据库目录中获得，并且时常地不被数据仓库元数据知识库记载(document)。多维元数据帮助数据仓库设计者表示表及其列之间的结构关系。一旦该元数据存在于数据库目录中，则 RDBMS 110 的其它组件例如数据库优化器(例如，DB2® UDB 优化器)可以利用该结构信息，并且更快地对由这些新的 OLAP 元数据对象描述的数据执行查询。元数据对象还能通过提供生成对数据仓库的多维查询所需的基本结构信息来协助商业智能工具。为了捕捉 OLAP 结构信息，OLAP 多维元数据系统 100 定义若干新的元数据对象。这些元数据对象能够描述频繁地用来对 OLAP 数据进行建模的模式例如星形连接和雪花模式

的重要方面。

将元数据对象添加到数据库目录提供了完全的功能性以及与其它数据库组件的集成。新的元数据对象以与常规表相同的方式由模式拥有。元数据对象的另一个设计点是它们的大部分单独地是有用的。也就是，元数据对象提供关于底层关系模式、元数据对象是否包括在更复杂的多维结构中的信息。

立方体模型可以以多种方式来构造，但是经常被构建成表示关系星形模式或雪花模式。基于简单星形模式的立方体模型围绕着中央事实元数据对象来构建，其中该中央事实元数据对象描述来自事实表的聚合关系数据。度量元数据对象描述根据关系表中的多列的数据计算，并且被连接在一起以创建事实元数据对象。图 2 示出了根据本发明的特定实现事实元数据对象 210 和度量元数据对象 220、230 与关系数据 250 相关。

在立方体模型中维元数据对象连接到事实元数据对象，就像是在星形模式中维表连接到事实表一样。来自关系表的数据的列由连接在一起以构成维元数据对象的属性元数据对象表示。

图 3 示出了根据本发明特定实现的样例星形连接模式。星形连接模式具有连至中央 Sales(销售)事实表 300 的 Time(时间)310、Product(产品)320 和 Region(地区)330 维表。为关系表中的相关维和事实表 300、310、320、330 列创建属性。每个维表 310、320、330 具有维键属性，如 TimeID(时间 ID)、ProductID(产品 ID)或 RegionID(地区 ID)。地区维表 330 还具有 City(城市)和 City_Population(城市人口)属性以及名称为“CityProp AR”的属性关系。该属性关系表达 City 属性中的每一个值确定 City_Population 属性中的对应值的函数相关性。在事实表内，存在 Sales(销售)和 Costs(成本)的两个度量以及三个维键属性 TimeID、ProductID 和 RegionID。

三个连接通过对应的维键属性将每个维表 310、320、330 连接到中央事实表 300。在本例中，维表 310、320、330 基于 TimeID、ProductID 或 RegionID 属性与事实表 300 相连。图 4 示出了根据本发

明的特定实现从关系表 450 构建维元数据对象 406、410。例如，在元数据对象中，维元数据对象 406 基于属性元数据对象 408 而构建，并且属性元数据对象 408 连接到关系表中的属性 452。维元数据对象 410 基于属性元数据对象 412、414 和连接元数据对象 416 而构建。这些属性元数据对象连接到关系表 450 中的属性 454 和 456。

分级结构存储关于维内的属性如何彼此相关和被结构化的信息。作为元数据对象，分级结构提供对维进行计算和导航的方法。每维具有对应的分级结构，其具有针对每个成员属性而定义的级别。例如，Region 维具有 RegionH 分级结构，其具有针对 State(州)和 City(城市)属性而定义的级别，并且还引用 CityPop AR 属性关系。在立方体模型中，每维可具有多个分级结构，但是本示例星形模式对于每维只定义了一个分级结构。

在星形模式中，所有维元数据对象以星形连接到中央事实元数据对象，以便创建立方体模型。连接元数据对象可以连接表以创建事实元数据对象或维元数据对象。元数据连接还能通过将事实元数据对象连接到维元数据对象而担当立方体模型内的胶粘剂(glue)。维元数据对象具有关于所有其组成分级结构、属性、属性关系和相关连接的信息。事实元数据对象具有关于所有其组成度量、属性和相关连接的信息。

图 5 示出了根据本发明的特定实现元数据对象 500 一起装入立方体模型，并且映射到关系表 550 的关系星形模式。立方体模型元数据对象 510 基于维元数据对象 512、514、连接元数据对象 516、518 以及事实元数据对象 520 而构建。

立方体模型元数据对象是灵活的元数据对象，其组成部分可以被复用以针对特定的应用创建更精确的立方体元数据对象。例如，立方体模型元数据对象可以具有 37 个事实，但是从该立方体模型元数据对象生成的一个立方体元数据对象可以去除一个或多个维元数据对象、维元数据对象的一个或多个级别和/或一个或多个度量元数据对象。

除了立方体模型元数据对象之外，还存在被称作立方体元数据对象的更特定元数据对象。立方体元数据对象是与 OLAP 概念立方体最接

近的元数据对象。立方体元数据对象是立方体模型元数据对象的特定实例或子集。立方体元数据对象具有从父立方体模型元数据对象派生的特定一组类似但更受限制的元数据对象，其包括：立方体维、立方体分级结构和立方体事实。例如，RegionCubeDim 是作为从 Region 维派生的属性的子集的立方体维。RegionCubeDim 引用 State 和 City 属性，但是不引用 City_Population 属性或 CityPop AR 属性关系。RegionCubeDim 引用其界定(scope)的 Region 维，并且所有结构信息包括连接信息与立方体模型 Region 维滞留在一起。

在特定实现中，立方体元数据对象具有按照每个立方体维而定义的一个立方体分级结构，而维元数据对象可以具有针对立方体模型元数据对象而定义的多个分级结构。立方体元数据对象与立方体模型元数据对象之间的这一结构差异允许采用单个 SQL 语句来检索立方体元数据对象。

图 6 示出了根据本发明的特定实现将概念元数据对象分类成三层。这些层是基本/关系层 600、多维层 610 和 OLAP 层 620。基本/关系层 600 向其它元数据对象提供基本底层结构，并且封装关系数据库的概念。多维层 610 包括引用基本/关系层 600 中的元数据对象以提供关系数据库之上的多维抽象的元数据对象。OLAP 层 620 包含表示 OLAP 结构的高级元数据对象。通过对来自其它层的元数据对象进行编组，OLAP 层 620 提供具有不同复杂度的 OLAP 立方体。

为了更好地理解本实施例，提供了一个例子。本例基于在数据集市(data mart)中使用的常见结构，即星形连接模式。对于星形连接模式，基于基本/关系层、多维层和 OLAP 层而创建元数据对象的实例。图 3 示出了根据本发明特定实现的简单星形连接模式，其包括事实表 300 即 Fact、以及三个维表 Time 310、Product 320 和 Region 340。

现有的数据库目录典型地存储表和列名称。关于这些表和列扮演什么角色以及这些表和列如何彼此相关的信息是不存在的。然而，采用 OLAP 多维元数据系统 100，通过创建元数据对象来捕捉该信息。

图 7 示出了根据本发明的特定实现创建与基本/关系层对应的元数据

对象。为所有维表列和在连接中使用的事实表列创建属性。为事实表中的每个事实列创建一个度量元数据对象。由三个连接元数据对象捕捉在该星形连接模式中使用的连接。连接元数据对象规定如何连接事实表和维表的对应属性。创建 Region 维表中的一个属性关系，以表示 City 和 City_Population 之间的关系，以及 City 属性中的每个值确定 City_Population 属性中的值的事实。

图 8 示出了根据本发明特定实现的来自基本/关系层的附加元数据对象。创建三个分级结构 800、810 和 820，其表示相关属性之间的关系。这些分级结构 800、810 和 820 通过多维在多维层中使用，以便创建用来对维进行计算和导航的装置。在 RegionH 分级结构 820 中，引用 CityPop AR 属性关系。捕捉应用于给定分级结构的所有属性关系。还对于每个分级结构创建一个立方体分级结构 850、860、870，以便在立方体上下文中使用。立方体分级结构 850、860、870 用来界定(scope)对于给定立方体感兴趣的分级结构的级别。立方体分级结构 850、860、870 还捕捉应用于其的属性关系。

图 9 示出了根据本发明特定实现的基于星形连接模式而创建的多维层元数据对象。为事实表 Fact 创建一个事实元数据对象 900。SalesFacts 元数据对象 900 包括可用的度量以及在事实到维连接中所需的属性。为作为星形连接模式的一部分的每个维表创建一个维元数据对象 910、920、930。维元数据对象对高度相关的属性进行编组，在本例中这些属性来自于单个维表。维元数据对象还引用应用于维属性上的分级结构。维可以具有多个所定义的分级结构，但是在本例中，对于每维，仅定义了一个分级结构。

图 10 示出了根据本发明特定实现的用来定义立方体的元数据对象 1000、1010、1020 的实例。立方体事实、立方体维和立方体分级结构元数据对象用来界定作为立方体的一部分的属性和度量。这些元数据对象的每一个引用正被界定的元数据对象，并且所有结构信息如连接保留在主(即父)元数据对象中。所有立方体特定对象保存对从其定义它们的主对象的引用。例如，立方体分级结构元数据对象具有对从其定义该立方

体分级结构元数据对象的分级结构元数据对象的引用。在特定实现中，对于多个立方体维，分配一个分级结构。在本例中，创建立方体事实SalesCubFacts 1000，并且它列出在该立方体中使用的度量(Sales)。

OLAP 层由立方体模型和立方体元数据对象组成。立方体模型元数据对象描述对于给定应用感兴趣的事实和维。立方体模型元数据对象的维可具有多个所定义的分级结构，这使立方体模型元数据对象是非常灵活的结构。立方体元数据对象从立方体模型元数据对象派生，从而所有立方体维、立方体分级结构和立方体事实元数据对象从立方体模型元数据对象派生。立方体模型元数据对象和立方体元数据对象之间的差异在于在立方体元数据对象中每维定义一个分级结构，这使得有可能采用单个 SQL 语句检索立方体元数据对象。

图 11 示出了根据本发明的特定实现创建 OLAP 层中每个元数据对象的一个实例。在本例中创建的立方体模型捕捉从图 3 的示例星形连接模式生成的一个可能的立方体模型 1100。基于立方体维 TimeCubeDim、ProductCubeDim、RegionCubeDim 和立方体事实 SalesCubeFacts 来创建立方体 1150。

A.2 元数据对象属性

每个元数据对象具有一组通用属性，以及元数据对象特定属性。通用属性用来标识元数据对象实例，描述元数据对象实例的使用或角色，并且跟踪元数据对象实例变化。在特定实现中，以对其它数据库元数据对象命名的相同方式，使用模式对元数据对象命名。当不希望缺省用户名模式时，可能需要元数据对象的完全限定(qualification)。

表 1 描述了根据本发明特定实现的对于所有元数据对象都存在的通用属性。

表1

属性	描述
名称	元数据对象的名称
模式	拥有元数据对象的模式
商业名称	向最终用户呈现的名称。该名称可以作为对于最终用户更有意义的名称而在图形用户界面上使用
注释	关于元数据对象性质或使用的文本描述或注释
创建时间	创建元数据对象的时间
创建者	定义了元数据对象的用户(模式)
修改时间	最后修改元数据对象的时间
修改者	执行了修改的用户(模式)

除了由所有元数据对象共享的公共一组通用属性之外，每个元数据对象具有一组元数据对象特定属性。这些元数据对象特定属性描述定义元数据对象的组成部分和特性(quality)。立方体模型是逻辑星形模式的表示。立方体模型是围绕着中央事实元数据对象的相关维元数据对象的编组。每维可以具有多个分级结构，这提高了立方体模型的灵活性。关于如何连接由事实和维元数据对象使用的表的结构信息被存储在立方体模型中。另外，存储在立方体模型中的是检索 OLAP 数据的足够信息。理解立方体模型并且可处理特定维的多个分级结构的其它报告和 OLAP 工具可以受益于立方体模型的使用。

立方体模型定义复杂的一组关系，并且可用来选择性地向应用暴露相关事实和维。将维连接到中央事实元数据对象的每个连接元数据对象作为集合与对应的维存储在一起。立方体模型组成部分的子集可以由用于不同分析目的很多立方体使用。

可以创建没有事实元数据对象或任何维的空立方体模型。然而，在创建对应立方体之前完成立方体模型。OLAP 多维元数据系统 100 通过确保立方体模型包括事实元数据对象、至少一维和现有事实与维之间的连接，以及所有属性引用合法的表来验证立方体模型。认为立方体模型完整不要求分级结构，但是，为了能够从立方体模型定义立方体，定义

每维的至少一个分级结构。

每个元数据对象具有一组元数据对象特定属性，其描述定义元数据对象的组成部分和特性。根据本发明的特定实现，表 2 描述了立方体模型的元数据对象特定属性。

表 2

属性	描述
事实	在立方体模型中使用的事实
(维、连接)集	在立方体模型中使用的维及其对应的连接

事实元数据对象编组对于给定应用感兴趣的相关度量。多个关系事实表可以通过特定属性来连接，以便映射附加的相关度量。事实元数据对象存储关于在事实到维连接中使用的属性、以及用来跨越多个数据库表映射附加度量的属性和连接的信息。因此，除了一组度量之外，事实元数据对象存储一组属性和一组连接。事实元数据对象在立方体模型中用作星形模式的中心。

事实元数据对象在星形模式中扮演事实表的角色。如同事实表所做的一样，事实元数据对象收集在数据库目录中通过度量表示的测量实体。它们不需要来自相同的表，从而允许设计者根据任何 OLAP 应用的要求来对度量进行编组。

根据本发明的特定实现，在表 3 中描述了事实元数据对象的元数据对象特定属性。

表 3

属性	描述
度量集	事实元数据对象中所有相关度量的集合
属性集	在事实元数据对象中使用的所有属性的集合
连接集	连接所有规定的度量和属性所需的所有连接的集合

维元数据对象扮演星形模式中的维表的角色。维对相关属性进行编组，其中这些相关属性一起描述一个或多个度量的某方面。因此，维元

数据对象提供对一起描述度量的一方面的一组相关属性进行分类的方法。在立方体模型中使用维来根据诸如 Region、Prouduct 或 Time 的逻辑类别来组织事实元数据对象中的数据。在维元数据对象特定属性中定义相关属性和将这些属性编组在一起所需的连接。

维引用一个或多个分级结构。分级结构描述维属性的关系和结构，并且可用来驱动维的导航和计算。

维还具有描述维是否是面向时间的类型。例如，被称作 Time 的维可能包含诸如 Year(年)、Quarter(季度)和 Month(月)的属性，并且它将是时间类型。被称作 Region 的另一维可能包含诸如 Country(国家)、State(州)、City(城市)和 Population(人口)的属性，并且它将是常规类型。类型信息可以由应用用来智能且适当地执行时间相关函数。

根据本发明的特定实现，在下表 4 中描述了维元数据对象的元数据对象特定属性。

表 4

属性	描述
属性集	在该维中使用的所有属性的集合
连接集	连接所有规定的属性所需的所有连接的集合。这里规定连接维表所需的连接
分级结构集	应用于该维的分级结构的集合
类型[常规，时间]	维类型

分级结构定义立方体模型的给定维内的一个或多个属性的集合之间的关系。定义这些关系提供了遍历(traverse)给定维的导航和计算装置。可以为立方体模型的维定义多个分级结构。分级结构元数据对象还引用一组将该分级结构中的属性链接到其它相关属性的属性关系。通过属性关系直接相关的属性可以作为分级结构的一部分来查询。例如，Region 维的分级结构可具有 City 属性，并且属性关系可以将 City 链接到 City_Population 属性。该分级结构可以在包括 City 的查询中包括 City_Population 信息。

分级结构描述了属性之间的父子关系。该信息由维参考(refer to)，以表示可以如何浏览维成员、以及如何聚合该维中的数据。

分级结构类型描述了该分级结构内的属性之间的关系。支持以下四种分级结构类型：平衡、非平衡、不规则(ragged)和网络。

图 12 示出了根据本发明特定实现的平衡分级结构 1200 的例子。平衡分级结构是这样一种分级结构，其有意义级别和分支具有一致的深度。每个属性的逻辑父位于直接在该属性之上的级别。平衡分级结构 1200 表示每个级别例如年 1210、季度 1220 和月 1230 的意义和深度是一致的时间。

图 13 示出了根据本发明特定实现的非平衡分级结构 1300 的例子。非平衡分级结构是这样一种分级结构，其级别具有一致的父子关系，但是对于特定级别中的所有成员具有不一致的语义意义。另外，分级结构分支具有不一致的深度。

非平衡分级结构可以表示组织图。例如，非平衡分级结构 1300 在该分级结构的最高级别表示 CEO，并且在其下可以分支出至少两种人，包括运营总裁(chief operating officer)和执行秘书。运营总裁也分支出更多的人，但是执行秘书不这样。在 CEO 与向 CEO 汇报的所有人之间都存在一致的父子关系。然而，直接在 CEO 之下的级别的语义意义由于该级别中不同的雇员类型而是不一致的。

不规则分级结构是这样一种分级结构，其中每个级别具有一致的意义，但是这些分支具有不一致的深度，因为分支级别中的至少一个成员属性未被填充。不规则分级结构可以表示地理分级结构，其中一致地使用每个级别的意义如城市或国家，但是该分级结构的深度不同。图 14 示出了根据本发明特定实现的不规则分级结构 1400。不规则分级结构 1400 表示定义有洲、国家、省/州和城市级别的地理分级结构。一个分支具有作为洲的北美、作为国家的美国、作为省/州的加利福尼亚、以及作为城市的旧金山。然而，当一个成员没有在所有级别上都具有条目时，分级结构 1400 变得不规则。例如，另一个分支具有作为洲的欧洲、作为国家的希腊以及作为城市的雅典，但是对于省/州级别没有条

目，因为这一级别不适用于希腊。在本例中，希腊和美国分支降至不同的深度，从而创建了不规则分级结构 1400。

网络分级结构是这样一种分级结构，其中不规定级别的次序(order)，但是级别确实具有语义意义。图 15 示出了根据本发明特定实现的网络分级结构 1500，其描述诸如 Color(颜色)、Size(尺寸)和 PackageType(包装类型)的产品属性。由于这些属性级别不具有内在的父子关系，因此级别的次序可以改变。装饰品公司可能具有诸如这样的成员条目，即针对颜色的白色、针对尺寸的小以及针对包装类型的热缩塑料包。第二成员条目可能具有针对颜色的红色、针对尺寸的大以及针对包装类型的盒装。

分级结构(平衡、非平衡、不规则或网络)还规定用于该分级结构的部署机制。部署机制定义如何解释分级结构的属性。支持下面两个部署机制：标准和递归。

标准部署机制使用分级结构的级别定义，其中该分级结构中的每个属性定义一个级别。例如，Time 维的平衡分级结构将由包括年、季度和月的每个所定义级别组织。标准部署可以用于所有四个分级结构类型。表 5 示出了根据本发明的特定实现使用标准部署来组织 Time 维的一些平衡分级结构属性。

表 5

年	季度	月
2001	第 1 季度	一月
2001	第 1 季度	二月
2001	第 1 季度	三月
2002	第 1 季度	一月
2002	第 1 季度	二月
2002	第 1 季度	三月

递归部署机制使用分级结构的属性之间的内在父子关系。使用递归部署的非平衡分级结构以父子属性对表示。例如，表 6 示出了根据本发

明特定实现的非平衡分级结构的属性对，其描述图 13 所示的组织图。这些父子属性对包括：执行总裁和执行秘书、执行总裁和运营总裁、运营总裁和通信总监(director of communications)、通信总监和通信专家。递归部署可以用于非平衡分级结构。

表 6

父属性	子属性
执行总裁	执行秘书
执行总裁	运营总裁
运营总裁	通信总监
通信总监	通信专家

根据本发明的特定实现，在下表 7 中描述了分级结构元数据对象的元数据对象特定属性。

表 7

属性	描述
属性列表	从分级结构的顶部到底部属性的有序列表。在递归分级结构的情况下，两个属性用作父和子
属性关系集	将分级结构属性链接到其它属性的所有属性关系的集合。
类型[平衡、非平衡、不规则、网络]	分级结构类型
部署[标准，递归]	分级结构部署

度量元数据对象定义测量实体，并且用于事实元数据对象中。度量在维的上下文内变得有意义。例如，300 的总收益本身没有意义。当将收益度量置于诸如 Region 的维的上下文中，该度量变得有意义。例如，佛蒙特州的收益是 300。度量元数据对象的常见例子是 Revenue(收益)、Cost(成本)和 Profit(利润)。

度量对象使测量实体的存在变得显式。度量由一个或多个 SQL 表达式定义，其中该 SQL 表达式可以简单到映射至一个表列，或者可以涉及多个列或者其它度量或属性。对于每个度量，定义聚合的列表以便在立方体模型或立方体的上下文中计算。该列表中的每个聚合规定诸如

SUM、COUNT、MIN、MAX 的聚合函数和应用聚合函数的维列表。聚合中的空维集表示要使用在该度量中未被显式引用的所有其余维。当所使用的第一聚合函数需要多于一个输入例如 CORRELATION 时，度量将具有多于一个 SQL 表达式模板。当度量具有单个 SQL 表达式模板时，度量可具有空的聚合列表，并且它仅参考其它度量。在这种情况下，发生被引用度量的聚合。度量和属性共享相同的名称空间，这意味着当由模式完全限定时名称在度量和属性之间必须是唯一的。度量的常见例子是 Sales(销售额)、Costs(成本)、Profit(利润)等。

度量由 SQL 表达式的聚合来定义。表、列、属性和度量被映射到模板，以构建 SQL 表达式(即“SQL 表达式模板”)。然后，使用所得到的 SQL 表达式作为对该度量的第一聚合函数的输入。如果度量具有多于一个聚合，则按照所列次序执行聚合函数，其中每个后继聚合以前一聚合的结果作为输入。如果度量元数据对象的 SQL 表达式仅引用其它度量，则由于被引用的度量描述任何所需聚合而省略聚合函数。

度量的 SQL 表达式通过两个属性的组合来创建：SQL 表达式模板以及列、属性和度量的列表。SQL 表达式模板使用标记表示法，其中{$$n}是标记，并且 n 从列表中引用特定列、属性或度量。列、属性和度量的列表是有序的，并且列、属性或度量的列表中的位置对应于标记“n”值。

SQL 表达式用作对第一聚合的输入。每个聚合规定应用于对应维集的函数。聚合函数可以是由底层数据库支持的任何聚合函数，例如包括 SUM、COUNT、MIN、MAX 和 CORRELATION。在特定实现中，每维由度量元数据对象聚合一次。如果维集为空，则将聚合函数应用于未被该列表中的另一个聚合特定使用的立方体或立方体模型中的所有维。在特定实现中，聚合函数是由 RDBMS 110 支持的用户定义的聚合函数。

简单度量的例子是 Revenue(收益)。可以为具有三维：Product(产品)、Market(市场)和 Time(时间)的立方体模型创建 Revenue 度量。Revenue 具有 SQL 表达式模板(template = “{$$1}”)，其表示到在列、属

性和度量的单项目列表中规定的列的简单映射，其中 list ="Column Fact.Rev"。聚合列表是(SUM, <NULL>)，其中 SUM 是聚合函数，并且<NULL>是空维集。该 SQL 表达式用作对 SUM 聚合函数的输入，从而产生这样的 SQL: SUM(Fact.Rev)。

更复杂的度量 Profit(利润)可能具有 SQL 表达式模板(template = "{$$1}-{$$2}")，其中属性、列和度量的列表是 list = "Measure Revenue，Column Fact.Cost"。通过以正确的引用替换标记，SQL 表达式变成: "Revenue-Fact.Cost"。通过将收益度量引用扩展成其列引用，SQL 表达式变成: "Fact.Rev-Fact.Cost"。Profit 度量的聚合列表是: (SUM, <NULL>)。使用利润 SQL 表达式作为 SUM 聚合函数的输入，Profit 度量的 SQL 是: SUM(Fact.Rev-Fact.Cost)。

如果度量具有需要两个或更多参数的聚合函数例如 CORRELATION，则该度量将具有这一数目的该函数要求其作为输入的 SQL 表达式。也就是，参数的数目匹配 SQL 表达式的数目。例如，如果 CORRELATION 需要两个参数，则将存在两个 SQL 表达式。

度量还具有基于 SQL 数据类型的数据类型。OLAP 多维元数据系统 100 自动地确定度量的数据类型。另外，度量和属性共享相同的名称空间。因此，当由模式完全限定时，每个名称在度量和属性之间是唯一的。根据本发明的特定实现，在下表 8 中描述了度量元数据对象的元数据对象特定属性。

表 8

属性	描述
SQL 表达式模板的列表	用作度量的第一聚合函数的输入的 SQL 表达式模板的列表。这些模板通过使用‘{$$n}’表示法来引用列、属性和度量。在模板中，n 是对应于列、属性和度量的列表的序号。
列、属性和度量的列表	对于每个 SQL 表达式模板，提供列、属性和度量的有序列表。如在 SQL 表达式模板中所规定的那样应用这些列、属性和度量。
聚合的列表(聚合函数、维集的列表)	规定如何计算度量的聚合的列表。每个聚合由 SQL 聚合函数和应用该函数的可选维集组成。
数据类型(模式、名称、长度、标度(scale))	确定属性的数据类型。基于 SQL 数据类型，并且由数据类型模式、名称、长度和标度组成。

属性表示数据库表列的基本抽象。属性由 SQL 表达式定义，其中该 SQL 表达式可以是到一个表列的简单映射，可以涉及多个列和其它属性，并且可以涉及底层数据库的所有功能性，例如用户定义的函数。在特定实现中，当在定义的 SQL 表达式中使用其它属性时，其它属性不能形成属性引用环。例如，如果属性 A 引用属性 B，则属性 B 不能引用属性 A。

属性的 SQL 表达式定义通过组合两个属性：SQL 表达式模板以及列和属性的列表来创建。SQL 表达式模板使用标记表示法，其中{$$n}是标记，其中 n 从该列表中引用特定列或属性。列和属性的列表是有序的，并且列或属性的列表中的位置对应于标记“n”值。

例如，SQL 表达式模板(template = “{$$1} || ' ' || {$$2}”)可以与对应的列表例如 list = “Column CUSTOMER.FIRSTNAME, Attribute LastName”一起使用以串接客户的名和姓，其中在名和姓之间留有一个空格。通过以正确的列表引用替换 SQL 表达式模板标记，SQL 表达式

是: “Customer.FirstName || ' ' || LastName”。属性引用进一步被扩展为列引用以形成 SQL 表达式: “Customer.FirstName || ' ' || Customer.LastName”。

在数据仓库或数据集市的设计中，属性可以担当多个角色。属性可担当的角色是: 级别、描述、维属性、维键或键。

级别属性用于分级结构中。常见的级别属性的例子是: 年和季度、州和城市。描述属性用于描述类型的属性关系中，并且将附加的描述信息关联到另一个属性。例如，被称作 Product(产品)的表可能具有产品代码的属性以及文本描述的描述属性。维属性用于维类型的属性关系中，并且定义另一个属性的特定特征和特性。常见的维属性的例子是 Population(人口)、Size(尺寸)和 Weight(重量)。维键属性用来连接事实和维元数据对象，并且表示维表中的主键，或者要在事实表中使用的来自维表的外键。键属性用来连接事实或维元数据对象内的表。键属性经常用于雪花模式中。

属性和度量共享相同的名称空间。因此，当由模式完全限定时，每个名称在属性和度量之间是唯一的。属性和度量元数据对象是关系数据库列的抽象。然而，它们由可包括多列的 SQL 表达式定义。度量比属性更特殊化(specialized) - 它们包括用来从较低级别数据计算较高级别汇总的聚合函数(列函数)。

表 9 描述了根据本发明特定实现的定义属性元数据对象的元数据对象特定属性。

表 9

属性	描述
SQL 表达式模板	定义属性的 SQL 表达式。SQL 表达式模板通过使用{$$n}表示法来引用列、和属性，其中，n 是对应于列和属性的列表的序号。
用于 SQL 表达式的列和属性的列表	组成属性的所有列和属性的有序列表。如在 SQL 表达式模板中所规定的那样应用这些列和属性。
数据类型(模式、名称、长度、标度)	确定属性的数据类型。基于 SQL 数据类型，并且由数据类型模式、名称、长度和标度组成。
角色[级别、描述、维属性(DIMATTR)、维键(DIMKEY)、键]	属性担当的角色

属性关系一般地描述属性的关系。这些关系由左和右属性、类型、基数(cardinality)以及关系是否确定函数相关性来描述。该类型定义右属性相对于左属性是什么角色。例如，ProductName(产品名称)右属性描述 ProductCode(产品代码)左属性。ProductName 与 ProductCode 之间的关系类型是 DESCRIPTION(描述)。基数描述左和右属性的实例如何相关，并且基于基数来解释它。在 1:1 基数中，对于每个右属性实例，存在最多一个左属性实例。在 1:N 基数中，对于每个右属性实例，存在最多一个左属性实例，而对于每个左属性实例，存在任意数目的右属性实例。在 N:1 基数中，对于每个右属性实例，存在任意数目的左属性实例，而对于每个左属性实例，存在最多一个右属性实例。在 N:N 基数中，对于每个右属性实例，存在任意数目的左属性实例，并且对于每个左属性实例，存在任意数目的右属性实例。

函数相关性属性(property)告诉该属性关系是否还能用作函数相关

性。函数相关性定义两个属性之间的函数关系。例如，可以在诸如City(城市)和 Mayor(市长)或者 Product(产品)和 Color(颜色)的属性之间定义函数相关性。函数相关性告诉每一个 City 值确定 Mayor 值，或者每一个 Product 值确定 Color 值。这意味着在该关系中描述的基数由设计者设置，这对于查询优化是有用的。

属性关系的一个使用在维中的分级结构的上下文内。与分级结构属性直接相关的属性可以作为分级结构的一部分来查询。这允许分级结构的每个级别定义补充给定级别的信息的属性。例如，分级结构可以具有City(城市)属性。City 属性可以通过属性关系与 City_Population (城市人口)属性相关。通过该属性关系信息，City_Population 信息可以包括在包括 City 的查询中。

根据本发明的特定实现，在下表 10 中描述了定义属性关系元数据对象的元数据对象特定属性。

表 10

属性	描述
左属性	在该关系中使用的左属性。
右属性	在该关系中使用的右属性。
类型 [描述，关联]	由属性关系描述的关系类型。该类型用来确定属性扮演什么角色。
基数[1:1，1:N，N:1，N:N]	在连接中预期的基数。
函数相关性[是，否]	确定该属性关系是否还是函数相关性。

连接元数据对象连接由两个元数据对象引用的关系表。两个元数据对象可以通过一对或多对映射到关系表列的属性元数据对象来连接。在事实到维连接中，连接元数据对象连接来自事实元数据对象的属性与来自维元数据对象的属性。在复合连接中，属性对的集合来自相同的表集合。例如，为了连接具有复合键 FirstName(名)和 LastName(名)的关系表 1 与具有复合键 FName 和 Lname 的关系表 2，使用一个具有两个连

接谓词(predicate)的关系连接；一个连接谓词用于 Table1.FirstName 和 Table2.Fname，第二连接谓词用于 Table1.LastName 和 Table2.LName。关于该复合连接的信息被存储在一个连接元数据对象中。

连接元数据对象由左属性、右属性和连接操作符的列表定义。另外，规定了连接类型和预期基数。可以在两个事实、两维或一个事实和一维之间使用连接。连接元数据对象被立方体模型、事实和维对象参考。

根据本发明的特定实现，在下表 11 中描述了定义连接元数据对象的元数据对象特定属性。

表 11

属性	描述
(左属性，右属性，操作符)的列表	左属性：连接左侧的属性。右属性：连接右侧的属性。操作符：在连接中预期的操作符[=，<，>，<>，>=，<=]。
类型[内部，完全外部、左外部、右外部]	预期的连接类型。
基数[1:1，1:N，N:1，N:N]	在连接中预期的基数。

立方体是可以使用单个 SQL 语句表达(deliver)的 OLAP 立方体的非常精确定义。每个立方体从单个立方体模型派生。立方体事实和立方体维的列表是被引用的立方体模型中立方体事实和立方体维的列表的子集。立方体视图名称也被定义，其表示数据库中的立方体。因为立方体维允许每个立方体维存在一个立方体分级结构，所以立方体适于不使用多个分级结构的工具和应用。

立方体的目的是定义 OLAP 结构的标准关系视图。除了关系视图之外，立方体提供按照多维描述其列的角色的扩展描述(例如，XML 文档)。在定义立方体的过程中，设计者选择可能元素的子集，为每维选

择单个分级结构。这确保了立方体无歧义地定义单个关系结果集。立方体的简单性使得立方体有用于较不复杂的 OLAP 应用，例如由万维网("Web")服务支持的便携式设备。

根据本发明的特定实现，在下表 12 中描述了立方体元数据对象的元数据对象特定属性。

表 12

属性	描述
立方体模型	从其派生立方体的立方体模型。
立方体事实	在立方体中使用的立方体事实。立方体事实从立方体模型中的事实元数据对象派生。
立方体维的列表	在立方体中使用的立方体维的有序列表。立方体维从立方体模型中的维派生。一个立方体分级结构与每个立方体维相关联。
立方体视图	数据库中表示立方体的视图
扩展描述	按照多维模型描述列的角色及其关系的 XML 文档

立方体事实元数据对象具有来自特定事实元数据对象的有序列表中的度量的子集。立方体事实元数据对象向立方体提供了界定立方体模型的事实的灵活性。从父事实元数据对象引用诸如连接和属性的结构信息。根据本发明的特定实现，在下表 13 中描述了定义立方体事实元数据对象的元数据对象特定属性。

表 13

属性	描述
事实	从其派生立方体事实的事实。
度量列表	在立方体中使用的度量的有序列表。所有度量是从其派生立方体事实的事实的一部分。

立方体维元数据对象用来界定要在立方体中使用的维。立方体维元数据对象引用从其派生它的维以及给定立方体的相关立方体分级结构。在特定实现中，一个立方体分级结构可以应用于立方体维。从维定义中引用应用于立方体维的连接和属性。根据本发明的特定实现，在下表 14

中描述了定义立方体维元数据对象的元数据对象特定属性。

表 14

属性	描述
维	从其派生立方体维的维。
立方体分级结构	应用于立方体维的立方体分级结构。

立方体分级结构元数据对象是分级结构的被界定(scoped)版本，并且在立方体中使用。立方体分级结构引用从其派生它的分级结构，并且可具有来自父分级结构的属性的子集。另外，立方体分级结构元数据对象引用应用于立方体上的属性关系。在特定实现中，可以为立方体的立方体维定义一个立方体分级结构。立方体分级结构元数据对象具有与从其派生该立方体分级结构元数据对象的分级结构相同的分级结构类型和部署机制。

立方体分级结构非常类似于分级结构；然而，立方体维参考单个立方体分级结构。这允许单个 SELECT 语句计算立方体的各个单元。

根据本发明的特定实现，在下表 15 中描述了定义立方体分级结构元数据对象的元数据对象特定属性。

表 15

属性	描述
分级结构	从其派生立方体分级结构的分级结构。
属性列表	从立方体分级结构的顶部到底部所有属性的有序列表。属性的次序应当与父分级结构中的相同。
属性关系集	将立方体分级结构属性链接到其它属性的所有属性关系的集合。

图 16 示出了根据本发明特定实现的一些元数据对象之间的一些关系。箭头表示元数据对象引用另一个元数据对象。例如，立方体元数据对象 1610 引用立方体模型元数据对象 1600。根据本发明的特定实现，在下表 16 中示出了更详细的元数据对象的关系描述。

表 16

元数据 元数据对象 1	引用	元数据 元数据对象 2
立方体模型	零或一	事实
立方体模型	零或更多	维/连接
立方体	一	立方体模型
立方体	一	立方体事实
立方体	一或更多	立方体维
事实	一或更多	度量
事实	零或更多	属性
事实	零或更多	连接
维	一或更多	属性
维	零或更多	连接
维	零或更多	分级结构
立方体事实	一	事实
立方体事实	一或更多	度量
立方体维	一	维
立方体维	一或更多	属性
立方体维	一	立方体分级结构
分级结构	一或更多	属性
分级结构	零或更多	属性关系
立方体分级结构	一	分级结构
立方体分级结构	一或更多	属性
立方体分级结构	零或更多	属性关系
度量	零或更多	度量
度量	零或更多	属性
度量	零或更多	维
属性	零或更多	属性
属性关系	二	属性
连接	二的倍数(二的最小值)	属性

根据特定实现，存在用于命名的元数据对象命名约定和规则。可以使用与在此所述不同的命名约定和规则而不脱离本发明的范围。存在两个不同的命名约定来对对象命名：普通和定界(delimited)。对于元数据对象，由于其灵活性，当对对象进行命名和引用数据库表和列时，使用定界约定。定界约定允许混合大小写名称、空格和特殊字符如国家语言字符。完整的字符集由对象所在数据库的代码页确定。

在特定实现中，除了命名约定之外，还对对象中的不同标识符应用一些规则。例如，模式具有 1-30 字节的长度，并且模式名称不以'SYS'开头；名称具有 1-128 字节的长度；商业名称具有 1-128 字节的长度；注释具有 0-254 字节的长度；表模式(在引用列时使用)具有 1-128 字节的长度；表名称(在引用列时使用)具有 1-128 字节的长度，并且列名称(在引用列时使用)具有 1-128 字节的长度。

除了所施行(enforce)的关系之外，还为每个元数据对象描述附加规则。也就是，每一个元数据对象具有其自己的规则集，并且如果元数据对象遵循该元数据对象的所有元数据对象规则，则元数据对象的实例是合法的。这些规则分成三个类别：基本规则、立方体模型完整性规则和优化规则。下面对特定规则的讨论提供了用于本发明特定实现的规则集。在其它实现中，可以修改用于一个或多个元数据对象的规则集而不脱离本发明的范围。

立方体模型元数据对象的基本规则是：(1) 立方体模型元数据对象参考零个或一个事实元数据对象；(2) 立方体模型元数据对象参考零个或更多个维；(3) 维-连接对具有维和连接两者；(4) 如果在事实元数据对象的属性列表中找到连接一侧的所有属性并且在维元数据对象的属性列表中找到所有另一侧属性，则与维相关联的连接是合法的；以及(5)对于在立方体模型的事实中引用的每个度量，由立方体模型引用度量聚合中的所有显式维引用。当立方体模型引用至少一维时，具有空维集的聚合匹配前面未被引用的立方体模型中的至少一维。

立方体元数据对象的基本规则是：(1) 立方体元数据对象参考一个立方体事实；(2) 立方体元数据对象参考至少一个立方体维；(3) 立方

体事实从在立方体模型中使用的事实派生；以及(4) 立方体维从在立方体模型中使用的维派生。

事实元数据对象的基本规则是：(1) 事实元数据对象参考至少一个度量；(2) 由事实引用的所有属性和度量是可连接的；(3)在事实元数据对象上下文中，可以在两个给定表之间定义单个连接；(4) 在事实元数据对象中不存在连接环；以及(5) 由事实元数据对象引用的所有连接参考事实元数据对象属性。

维元数据对象的基本规则是：(1) 维元数据对象参考至少一个属性；(2) 由维引用的属性是可连接的；(3) 不存在连接环；(4) 在维上下文中，在任何两个给定表之间定义单个连接；(5) 由维引用的分级结构参考该维的属性；(6) 由维的分级结构引用的属性关系参考该维的属性；以及(7) 由维引用的连接参考该维的属性。

立方体事实元数据对象的基本规则是：(1) 立方体事实元数据对象参考至少一个事实；(2) 立方体事实元数据对象参考至少一个度量；以及(3) 由立方体事实元数据对象引用的度量是事实元数据对象的一部分。

立方体维元数据对象的基本规则如下：(1) 立方体维元数据对象参考一维；(2) 立方体维元数据对象参考立方体分级结构；以及(3) 由立方体维元数据对象引用的立方体分级结构从由立方体维元数据对象的维引用的分级结构派生。

分级结构元数据对象的基本规则是：(1) 分级结构元数据对象参考至少一个属性；(2) 对于递归部署，要求两个属性；(3) 分级结构内的每一个属性关系具有左属性作为该分级结构的一部分；(4) 该分级结构内的每一个属性关系具有 1:1 或 N:1 的基数；以及(5) 根据本发明的特定实现如表 17 所示允许分级结构类型和分级结构部署的特定组合。

表 17

类型/部署	标准	递归
平衡	X	
不规则	X	
非平衡	X	X
网络	X	

立方体分级结构元数据对象的基本规则是：(1) 立方体分级结构元数据对象参考一个分级结构；(2) 立方体分级结构元数据对象参考至少一个属性；(3) 由立方体分级结构元数据对象引用的属性是该分级结构的一部分；(4) 立方体分级结构元数据对象中属性的次序与该分级结构中的相同(被定义为网络的分级结构除外)；(5) 分级结构内的每一个属性关系具有左属性作为该分级结构的一部分；以及(6) 在立方体分级结构元数据对象中引用的属性关系也在定义立方体分级结构的分级结构中被引用。

度量元数据对象的基本规则是：(1) 度量元数据对象可以具有属性、列、度量，或者不具有它们中的任一个，作为每个 SQL 表达式模板的参数；(2) 用作 SQL 模板参数的属性和度量不能在属性和/或度量之间产生相关环；(3) 在度量元数据对象中定义的每一个 SQL 模板不是空字符串；(4) SQL 模板不使用聚合函数；(5) 如果引用至少一个度量以及仅仅引用度量，则不需要聚合；(6) 如果存在聚合，则 SQL 模板的数目匹配第一聚合函数的参数数目；(7) 具有多个 SQL 模板的度量元数据对象在聚合脚本中定义至少一个聚合步骤；(8) 如果度量元数据对象 A 参考定义多个 SQL 模板的度量元数据对象 B，则度量元数据对象 A 没有聚合脚本；该规则适用于度量引用树中的所有级别；(9) 多参数聚合函数用作第一聚合；(10) 如果度量元数据对象定义一个或多个聚合，则一个聚合可具有空维集；(11) 在度量元数据对象内，维不可以在聚合内或者跨越聚合被引用多于一次；(12) 在 SQL 表达式模板内，标记指示符(即{$$#})以 1 开始编号，并且是连续的而无编号间断；以及(13) 在 SQL 表达式内，每个列、属性和度量被引用至少一次。

属性元数据对象的基本规则是：(1) 属性元数据对象可以具有属性、列表或者不具有它们中的任一个作为 SQL 模板的参数；(2) 用作 SQL 模板参数的属性不能在属性之间产生相关环；(3) SQL 模板不能是空字符串或空白字符串；(4) 不允许聚合函数是 SQL 模板的一部分；(5) 在 SQL 表达式模板内，标记指示符(即{$$#})以 1 开始编号，并且是连续的而无编号间断；以及(6) 在 SQL 表达式内，每个列、属性和度量被引用至少一次。

属性关系元数据对象的基本规则是：(1) 属性关系元数据对象参考两个属性；以及(2) 属性关系元数据对象不能被定义为具有基数=N:N 和函数相关性=是。

连接元数据对象的基本规则是：(1) 连接元数据对象参考至少一个由左属性、右属性和操作符号组成的三元组；(2) 连接元数据对象中的所有左属性解析(resolve)成单个表的一个或多个列；(3) 连接元数据对象中的所有右属性解析成单个表的一个或多个列；以及(4) 连接元数据对象的每个三元组定义合法操作；左和右属性的数据类型，以及为它们定义的操作是兼容的。

立方体模型完整性规则扩展基本规则，以便确保立方体模型具有所需的与其它元数据对象的链接，以便允许形成有效的仓库 SQL 查询。立方体模型元数据对象的立方体模型完整性规则是：(1) 立方体模型元数据对象参考一个事实；(2) 立方体模型元数据对象参考一个或多个维。

优化规则扩展立方体模型完整性规则，以便确保可以执行仓库 SQL 查询的优化。

立方体模型元数据对象的优化规则是：(1) 在事实到维中使用的连接具有 1:1 或 N:1 的基数，并且将事实表连接到维的主表。

维元数据对象的优化规则是：(1) 考虑由维的连接形成的连接网络，存在至少一个表、主表，其中从该表辐射的所有连接具有 N:1 或 1:1 的基数。

连接元数据对象的优化规则是：(1) 在参与连接的列上定义有约

束；如果连接是自连接，即在等式(equality)的两侧都使用相同的列集，则定义匹配该列集的主键；在所有其它情况下，当一侧的列集不同于连接的另一侧，则主键匹配连接一侧的列，并且外键匹配其它列集以及引用该主键；(2) 连接基数是 1:1、N:1 或 1:N；如果连接是自连接，则基数是 1:1，在所有其它连接的情况下，基数在定义了主键的一侧为 1，而在定义了外键的一侧为 N；如果外键侧也具有在其上定义的主键，则使用 1 作为基数；(3) 在连接中使用的所有属性都解析成不可为空的 SQL 表达式；以及(4) 连接类型是内部连接。

A.4 元数据对象例子

图 17 示出了根据本发明特定实现的由两个维表 1710、1720 和事实表 1700 组成的星形模式。两条线 1730、1740 表示事实表 1700 与维表 1710、1720 之间的连接。在特定实现中，数据库设计者使用建模工具或用户接口 150 可以创建元数据对象 130 的元数据对象实例。如果元数据对象与新的模型重叠，则在生成多维元数据期间定义的大多数元数据对象 130 可以被复用于新的模型

图 18A-18E 示出了根据本发明特定实现的可以针对图 17 的星形模式生成的元数据对象实例的可能集合，并且为简单起见，示出了元数据对象的一些属性。具体地说，图 18A-18E 中的一些被省略的属性是常见属性。例如，图 18A-18E 示出了立方体模型元数据对象实例 1800、立方体元数据对象实例 1802、事实元数据对象实例 1804、立方体事实元数据对象实例 1806、度量元数据对象实例 1808、1810、维元数据对象实例 1812、1814、立方体维元数据对象实例 1816、1818、分级结构元数据对象实例 1820、1822、1824、立方体分级结构元数据对象实例 1826、1828、连接元数据对象实例 1830、1832 以及属性元数据对象实例 1834-1848。

用户可以使用用户接口 150 来创建元数据对象。在创建空的立方体模型元数据对象之后，创建事实元数据对象和维元数据对象，并且通过创建适当的连接元数据对象将它们连接到立方体模型元数据对象。

可以修改在此讨论的元数据对象的属性，而不脱离本发明的范围。

B. 规定用于关系在线分析处理(ROLAP)引擎的多维计算

OLAP 多维元数据系统 100 使得能够创建度量元数据对象以协助多维计算。在特定实现中，度量元数据对象包括在表 8 中定义的特定属性。

度量元数据对象中的度量由 SQL 表达式的聚合来定义。具体地说，将表列、属性和度量映射到 SQL 表达式模板，以构建 SQL 表达式。然后，使用所得到的 SQL 表达式作为度量元数据对象的第一聚合函数的输入。如果度量元数据对象具有多于一个聚合，则按照所列次序执行聚合函数，其中每个后继聚合以前一聚合的结果作为其输入。如果度量元数据对象的 SQL 表达式仅引用其它度量，则省略聚合函数，这是因为被引用的度量描述任何所需的聚合。

在计算度量时使用的 SQL 表达式通过组合两个属性来创建：SQL 表达式模板的列表以及列、属性和度量的列表。SQL 表达式模板使用标记表示法，其中{$$n}是标记，并且 n 引用该列表中的特定列、属性或度量。列、属性和度量的列表是有序的，并且列、属性或度量的列表中的位置对应于标记“n”值。由于大多数聚合函数接受单个表达式作为输入，因此对于大多数聚合函数，该列表中 SQL 表达式模板的数目为一。然而，当使用诸如 CORRELATION 的聚合函数时，SQL 表达式模板的数目匹配由聚合函数接受的输入参数的数目。

同样地，使用 SQL 表达式作为对第一聚合的输入。每个聚合规定应用于对应维集的函数。该聚合函数可以是由底层 RDBMS 110 支持的任何聚合函数，例如包括 SUM、COUNT、MIN、MAX 和 CORRELATION。在特定实现中，每维由度量元数据对象聚合一次。如果维集为空，则对未被该列表中的任何其它聚合特定使用的立方体或立方体模型中的所有维应用聚合函数。

多维元数据软件 120 使用度量元数据对象中的元数据，自动地生成用于生成立方体视图的 SQL 语句。

B.1 度量要求

该章节描述根据本发明特定实现的一些度量要求。

度量的一个要求是支持度量内的特定计算次序。由立方体模型元数据对象或立方体元数据对象引用的度量元数据对象集的计算次序无需相同-每个度量元数据对象可以规定与任何其它度量元数据对象的计算次序不同的计算次序。例如，出售量=SUM(收益/单价)，并且利润率=SUM(利润)/SUM(收益)。图 19 示出了根据本发明特定实现的表 A 1900，其表示基本数据。名称为“衣服”的成员是裤子 1902、衬衫 1904 和领带 1906 的父，并且衣服的出售量使用单价 1910 和收益 1912 来确定：(680/40)+(780/60)+(175/25)=17+13+7=37。衣服的利润率使用收益 1912 和利润 1914 来确定：(68+117+52.5)/(680+780+175)=237.5/1635=0.145。

度量的另一个要求是支持带多个输入参数的聚合函数，例如相关操作(例如，CORRELATION(Revenue, Profit))。度量对象需要为每个聚合函数输入定义独立的表达式。

度量的另一个要求是支持半加性(semi-additive)度量，例如快照度量(例如，Inventory(库存))。例如，对于 Market 和 Product 维，执行求和操作(例如，SUM(Inventory))。对于 Time 维，执行 MIN 操作(例如，MIN(Inventory))。

图 20 示出了根据本发明特定实现的表 B 2000，其表示具有聚合:(SUM, Market)和(Min, Time)的度量。在表 B 2000 中，不带星号(例如，*)或加号(例如，+)的数表示基本数据。带星号的数表示(SUM, Market)的聚合。带加号的数表示(Min, Time)的聚合。例如，针对加利福尼亚 2002 和 2004 年 1 月的(SUM, Market)是 28，并且针对加利福尼亚 2002 和 2006 年第 1 季度的(Min, Time)是 25。

度量的附加要求是对于每维支持一个聚合函数，支持跨越多维的不同计算次序；以及针对(targeting to)复杂应用(例如，通过(SUM, Product)、(AVG, Time), (MAX, Market)查找具有最大平均库存的市场位置)。针对复杂应用是半加性度量的更一般性表示，下面参照图 21 对此进行进一步的描述。另外，针对复杂应用的要求由下面进一步描述的非对称度量表示。非对称度量在聚合列表中定义多个聚合。

图 21A-21D 示出了根据本发明特定实现的表 C 2100，其表示具有聚合:(SUM, Product)(即产品之和)、(AVG, Time)(即时间上的平均值)和(Max, Market)(即市场的最大值)的度量。在表 C 2100 中，不带星号(例如，*)或加号(例如，+)或者短划线(例如，-)的数表示基本数据。也就是，在图 21A 中示出了基本数据。在图 21B 中，在表 C 2100 中，添加了(SUM, Product)的聚合数，并且它们以星号标识。例如，对于衣服 2102、洛杉矶 2104、2105 年 1 月，(SUM, Product)为 66。在图 21C 中，在表 C 2100 中，添加了(AVG, Time)的聚合数，并且它们以加号标识。例如，对于裤子 2106、圣何塞 2108、2109 年第一季度(QTR1)，(AVG, Time)为 11。在图 21D 中， 添加了(MAX, Market)的聚合数，并且它们由短划线(例如，-)标识。例如，对于衬衫 2110、加利福尼亚 2112、2105 年 1 月，(MAX, Market)为 36。

B.2 描述度量

在本发明的特定实现中，可以创建度量元数据对象，其包括表达式列表和聚合列表。在章节 A 中详细地讨论了度量元数据对象。为易于理解起见，在该章节中也将讨论度量元数据对象。度量元数据对象中的表达式列表包括用于每个表达式的 SQL 表达式模板以及用于每个 SQL 表达式模板的列、属性和度量的列表。聚合列表中的每个条目包括聚合函数和对应的维集。空的维集意味着所有剩余维要用于聚合函数。在特定实现中，对于度量元数据对象，只有一个聚合可具有空的维集。

图 22 示出了根据本发明特定实现的两个全加性度量元数据对象(Cost 2210 和 Revenue 2220)的创建。在图 22 的例子中，存在三个维，即 Product 2202、Market 2204 和 Time 2206。每个度量元数据对象 2210、2220 规定<NULL>维集，其意味着所有维都用于 SUM 的聚合函数。

为具有三维: Product 2202、Market 2204 和 Time 2206 的立方体模型创建 Cost 度量元数据对象 2210。Cost 度量元数据对象 2210 具有 SQL 表达式模板 2212 (template = “{$$1}”)，其表示到在列、属性和度量的单项目列表中规定的列的简单映射(list = “Column Fact.Cost”)。也

就是，对于 Cost 度量元数据对象 2210，表达式列表参考“{$$1}”，其是在生成 SQL 表达式时以列 Fact.Cost 替换的标记。聚合列表 2214 是(SUM, <NULL>)，其中 SUM 是聚合函数，并且<NULL>是空的维集。来自 SQL 表达式模板 2212 的 SQL 表达式用作 SUM 聚合函数的输入，从而产生 SQL: SUM(Fact.Cost)。

为具有三维：Product 2202、Market 2204 和 Time 2206 的立方体模型创建 Revenue 度量元数据对象 2220。Revenue 度量元数据对象 2220 具有 SQL 表达式模板 2222 (template = “{$$1}”)，其表示到在列、属性和度量的单项目列表中规定的列的简单映射，其中 list = “Column Fact.Rev”。也就是，对于 Revenue 度量元数据对象 2220，表达式列表参考“{$$1}”，其是在生成 SQL 表达式时以列 Fact.Rev 替换的标记。聚合列表 2224 是(SUM, <NULL>)，其中 SUM 是聚合函数，并且<NULL>是空的维集。来自 SQL 表达式模板 2222 的 SQL 表达式用作 SUM 聚合函数的输入，从而产生 SQL: SUM(Fact.Rev)。

图 23 示出了根据本发明特定实现的半加性度量的创建。Inventory(库存)度量元数据对象 2310 具有 SQL 表达式模板 2312(template = “{$$1}”)，其表示到在列、属性和度量的单项目列表中规定的列的简单映射，其中 list = “Column Fact.Inv”。在 Inventory 度量元数据对象 2310 中，聚合列表 2314 包括两个聚合 SUM 和 AVG(即平均)。Inventory 度量元数据对象 2310 最后聚合 Time 维，从而聚合列表规定用于 SUM 的<NULL>，其意味着在该列表中未被引用的所有维(即，不同于时间的其它维，在本例中其将是产品和市场)，并且规定用于 Time 的 AVG。首先执行 SUM 操作，并且使用该求和操作的结果来执行 AVG 操作。所得到的 SQL 表达式涉及多个聚合步骤。为易于理解起见，提供了所得到的 SQL 表达式的简单例子。例如，所得到的 SQL 表达式可以是 AVG(S1)，其中 S1 是 SUM(Fact.Inv)的结果。

图 24 示出了根据本发明特定实现的具有聚合的复合度量的创建。在这种情况下，Profit(利润)度量元数据对象 2410 使用预定义的度量，并且确定要执行的聚合。更复杂的度量例如 Profit 可能具有 SQL 表达

式模板 2412(template = “{$$1} - {$$2}”)，其中列、属性和度量的列表是 list = “Measure Revenue, Column Fact.Cost”。也就是，该表达式列表包括“{$$1} - {$$2}”，其表示以来自 Revenue 度量元数据对象 2220 的聚合结果替换第一标记{$$1}，并且以来自 Cost 度量元数据对象 2210 的聚合结果替换第二标记{$$2}。通过以正确的引用替换这些标记，SQL 表达式变成：“Revenue-Fact.Cost”。通过将收益度量引用扩展成其列引用，SQL 表达式变成：“Fact.Rev-Fact.Cost”。

Profit 度量元数据对象 2410 的聚合列表 2414 为：(SUM, <NULL>)。在聚合列表 2410 中，规定<NULL>维集，以表示用于 SUM 操作的所有维。使用利润 SQL 表达式作为对 SUM 聚合函数的输入，Profit 度量的 SQL 表达式是：SUM(Fact.Rev-Fact.Cost)。也就是，通过对从收益减去成本的所有减法结果进行求和来得到利润。

图 25 示出了根据本发明特定实现的没有聚合的复合度量的创建。在这种情况下，Profit Margin(利润率)度量元数据对象 2510 使用预定义的度量元数据对象(即 Revenue 度量元数据对象 2220 和 Profit 度量元数据对象 2410)，并且不要求执行任何聚合(在聚合列表 2414 中以<NULL>而非聚合表示)。在这种情况下，聚合来自基本度量。

Profit Margin 度量元数据对象具有 SQL 表达式模板 2512(template = “{$$1}/{$$2}”)。第一标记{$$1}以来自 Profit 度量元数据对象 2410 的聚合结果替换，而第二标记{$$2}以来自 Revenue 度量元数据对象 2220 的聚合结果替换。这样，所得到的用于 Profit Margin 度量的 SQL 表达式为 SUM(Fact.Rev-Fact.Cost)/SUM(Fact.Rev)。也就是，计算利润之和，计算收益之和，并且通过将利润除以收益来获得利润率。

图 26 示出了根据本发明特定实现的具有 OLAP 函数的度量的创建。Profit Rank(利润排名)度量元数据对象 2610 利用 RANK 函数(例如，可从 DB2® UDB RDBMS 获得)以对利润度量进行排名。新的 Profit Rank 度量元数据对象 2610 不需要任何聚合。具体地说，Profit Rank 度量元数据对象 2610 具有 SQL 表达式模板(template = “RANK( ) OVER (ORDER BY {$$1}”)，其表示以来自 Profit 度量元数据对象 2410

的聚合结果替换第一标记{$$1}。聚合列表 2614 通过<NULL>而非聚合函数来表示不存在聚合。所得到的用于 RANK 度量的 SQL 表达式是 RANK( ) OVER (ORDER BY SUM(Fact.Rev - Fact.Cost))。

图 27 示出了根据本发明特定实现的具有聚合和多个输入的度量。RevProfit Correlation(收益利润相关)度量元数据对象 2710 利用 Revenue 和 Profit 度量元数据对象 2220、2410。对于 RevProfit Correlation 度量元数据对象，它具有两个 SQL 表达式模板 2712 和 2713，每一个都具有以{$$1}表示的第一标记。对于 SQL 表达式模板 2712(template = "{$$1}")，以来自 Revenue 度量元数据对象 2220 的聚合结果替换第一标记{$$1}。对于 SQL 表达式模板 2713(template = "{$$1}")，以来自 Profit 度量元数据对象 2410 的聚合结果替换第一标记{$$1}。然后，执行相关。相关是提供两个数列彼此相关的程度的测量的统计函数。所得到的用于 Correlation 度量的 SQL 表达式是 CORRELATION(Fact.Rev, (Fact.Rev-Fact.Cost))。聚合列表 2714 规定 CORRELATION 操作和用来表示所有维的<NULL>维集。

图 28 示出了根据本发明某些实现的来自图 22-27 的所有定义的度量元数据对象。一些度量元数据对象基于其它度量元数据对象而构建。例如，RevProfit Correlation 度量元数据对象 2710 基于 Revenue 度量元数据对象 2220 和 Profit 度量元数据对象 2410 而构建，而 Profit 度量元数据对象 2410 基于 Cost 和 Revenue 度量元数据对象 2210、2220 而构建。

B.3 为以一个或多个度量元数据对象表示的度量生成 SQL 语句

多维元数据软件 120 生成计算由度量元数据对象表示的度量集的单个 SQL 语句。图 29A 示出了根据本发明特定实现的用于从一个或多个度量元数据对象生成 SQL 语句的逻辑。控制在块 2900 中以这样的操作开始：接收一个或多个度量元数据对象中的每一个的度量描述并且基于这些度量描述而生成该一个或多个度量元数据对象，例如图 22-28 所示的度量元数据对象中的一个或多个。在块 2902 中，接收引用所有度量元数据对象的事实元数据对象的事实描述，并且从该事实描述生成事实

元数据对象。另外，事实描述引用属性元数据对象和连接元数据对象。在块 2903 中，接收一个或多个维元数据对象中的每一个的维描述，并且从这些维描述生成该一个或多个维元数据对象。该维描述引用一个或多个属性元数据对象、零个或更多个连接元数据对象、以及一个或多个分级结构元数据对象。每个分级结构元数据对象包括用来构建 ROLLUP 子句的信息。具体地说，立方体模型元数据对象可以引用若干可能的分级结构，但是在本发明的特定实现中，SQL 生成要求选择单个分级结构。在块 2904 中，接收引用事实和一个或多个维元数据对象的立方体模型元数据对象的立方体模型描述，并且从该立方体模型描述生成立方体模型元数据对象。在块 2906 中，接收立方体模型元数据对象的子集的选择。在块 2908 中，根据该选择来生成立方体元数据对象。另外，从一个或多个度量元数据对象中的元数据生成用于创建立方体视图的 SQL 语句。SQL 语句的生成还可以使用其它元数据对象(例如，分级结构元数据对象)中的其它元数据。

具体地说，SQL 语句的生成可以从分级结构元数据对象中的元数据生成一个或多个 ROLLUP 操作符。ROLLUP 操作符，GROUP BY 子句的扩展，基于列的列表而生成多个小计编组子句。编组子句使用来自分级结构元数据对象的信息来生成。就 OLAP 而言，这在给定维中具有分级结构计算的相同效果。考虑诸如位置的维，其具有由国家、州和城市组成的分级结构。ROLLUP(country, state, city)子句生成表示该分级结构的计算的编组子句。n 个元素(c_1, c_2, …, c_{n-1}, c_n)的 ROLLUP 的一般规定等同于下面编组子句：

(c_1, c_2, …, c_{n-1}, c_n)

(c_1, c_2, …, c_{n-1})

…

(c_1, c_2)

(c_1)

注意，ROLLUP 子句中的 n 个元素翻译成(n+1)个编组子句。OLAP 应用可以具有多维(例如，在维元数据对象中定义)。用于每维的

ROLLUP 以关系方式返回表示 OLAP 立方体的结果。在单个语句中组合多于一个 ROLLUP 操作符产生为每个 ROLLUP 生成的编组子句的笛卡尔乘积。例如，在单个语句中组合下面 ROLLUP 操作符对 ROLLUP(country, state)、ROLLUP(year, month)导致生成下面编组子句，其是构成立方体的编组子句的集合:

(国家, 州, 年, 月)

(国家, 州, 年)

(国家, 州)

(国家, 年, 月)

(国家, 年)

(国家)

(年, 月)

(年)

( )

使用 ROLLUP 操作符的查询在单个结果集中包括所有生成的编组子句。因此，结果集包括所有编组子句列加上聚合列的联合。为了组合不同编组集的结果，在给定行不是成员的任何编组列中返回空(null)，如下面例子所示。对于单维的 ROLLUP 查询的结果，参见表 18。生成包括 ROLLUP 操作符的 SELECT 语句。例如，在下面 SELECT 语句中，从度量元数据对象生成“sum”操作符，并且从连接元数据对象生成连接。

```
SELECT country, state, sum(amt) AS revenue
FROM fact f, location l
WHERE f.lid = l.lid
GROUP BY ROLLUP(country, state)
```

表 18

Country(国家)	State(州)	Revenue(收益)
-	-	235329.23999999999
加拿大	-	35754.639999999999
加拿大	ON	35754.639999999999
美国	-	199574.60000000001
美国	CA	103910.41
美国	NY	94665.190000000002

在表 18 的例子中，以州列中的空(以短划线示出)表示具有美国的聚合收益的行。以国家和州列中的空表示具有所有国家和州的聚合收益的行。

虽然图 29A 描述了使用度量元数据对象来生成立方体视图，但是在另外的实现中，可以在不创建立方体视图的情况下使用度量元数据对象。也就是，立方体视图是使用度量元数据对象的定义以便规定计算的一种方式。使用度量元数据对象的定义的另一种方式是让应用读取度量定义和立方体模型元数据，并且从这些度量定义和立方体模型元数据直接生成 SQL 语句。

图 29B 示出了根据本发明特定实现的用于从一个或多个度量元数据对象和立方体模型元数据对象生成 SQL 语句的逻辑。控制在块 2910 中以这样的操作开始：接收一个或多个度量元数据对象中的每一个的度量描述，并且基于这些度量描述而生成该一个或多个度量元数据对象，例如图 22-28 所示的度量元数据对象中的一个。在块 2912 中，接收引用所有度量元数据对象的事实元数据对象的事实描述，并且从该事实描述生成事实元数据对象。另外，该事实描述引用属性元数据对象和连接元数据对象。在块 2913 中，接收一个或多个维元数据对象中的每一个的维描述，并且从这些维描述生成该一个或多个维元数据对象。该维描述引用一个或多个属性元数据对象、零个或更多个连接元数据对象、以及一个或多个分级结构元数据对象。每个分级结构元数据对象包括用来构建 ROLLUP 子句的信息。具体地说，立方体模型元数据对象可以引用

若干可能的分级结构，但是在本发明的特定实现中，SQL 生成要求选择单个分级结构。在块 2914 中，接收引用事实和一个或多个维元数据对象的立方体模型元数据对象的立方体模型描述，并且从该立方体模型描述生成立方体模型元数据对象。在块 2916 中，由应用程序对立方体模型元数据对象的子集进行选择。在块 2918 中，在该应用程序的控制之下，使用该立方体模型元数据对象和一个或多个度量元数据对象，生成用来检索多维信息的 SQL 语句。SQL 语句的生成还可以使用其它元数据对象(例如，分级结构元数据对象)中的其它元数据。

多维元数据软件 120 解决采用单个 SQL 语句计算多个度量中的关键问题(即度量的对称性、所涉及聚合函数的分布性、以及维在聚合脚本中出现的次序)。另外，多维元数据软件 120 处理各种查询类型(例如，总计查询、基于切片(Slice)的查询、以及完整立方体查询)。在总计切片中，仅返回所有维的总计。切片是子立方体，而完整立方体是整个立方体。

度量被表示为度量元数据对象。当在单个 SQL 语句中要计算多个度量时，本发明的实现确定这些度量是否兼容。兼容度量对于它们引用的维具有相同的聚合次序规定。如果一组度量是不兼容的，则本发明确定至少一种方式来计算要在单个 SQL 语句中组合的不兼容度量。在特定实现中，可以使用 JOIN 操作(也被称作“连接”)来组合不兼容的度量，并且该处理在图 29C 中作进一步的描述。在特定实现中，可以通过重构(restructure)聚合步骤使得它们兼容(也被称作“嵌套”)来组合不兼容的度量，并且该处理在图 29D 中作进一步的描述。在特定实现中，实现连接和嵌套的混合形式，其中连接一些度量集，并且嵌套一些度量集。

图 29C 示出了根据本发明特定实现的用于组合度量的逻辑的进一步详情。图 29C 的处理示出了用来生成用于计算所有度量的单个 SQL 语句的最高级逻辑。本发明的实现试图生成最少数目的兼容度量集。另外，当多个集合不可避免时，由于度量不兼容性，因此优选的集合组合是这样的组合，其具有对称度量集，所有对称度量位于其中。

在图 29C 中，控制在块 2940 中以访问度量集开始。度量集包括一

个或多个度量。通过引用一个或多个元数据对象来访问这些度量。在块2942中，将该一个或多个度量元数据对象中的度量集分成对称度量集和非对称度量集。对称度量具有单个聚合操作符，并且不规定任何特定的聚合次序。在很多应用中，这样的度量是常见的。所有对称度量是兼容的。可以通过使用每维的 ROLLUP 操作符来生成用来计算所有对称度量的单个 SQL 语句。例如，在前面例子中使用的 Revenue、Cost 和 Product 度量是对称的。一些度量元数据对象可以引用一个或多个其它度量。例如，Profit Margin 度量引用 Profit 和 Revenue 度量，其引用 Income(收入)和 Expense(开支)度量。由于 Income 和 Expense 度量是对称的，因此 Profit Margin 度量在对称度量集中。在块 2944 中，为对称度量集生成 SQL 语句。在块 2945 中，为非对称度量集生成 SQL 语句。可以独立地和同时地或者以任何次序处理对称和非对称度量集。在块 2946 中，组合对称度量集的 SQL 语句和非对称度量集的 SQL 语句，以形成用于检索多维信息的单个 SQL 语句。用来组合这两个语句的技术取决于非对称度量集的性质。当可以从公共的对称子立方体计算非对称集中的一些度量时，可以使用嵌套来通过将对称计算改写为基于公共对称子立方体的计算而构建的嵌套计算来组合这些度量和对称度量的计算。如果非对称集的任何度量要求以特定次序聚合多个维(即几乎没有或没有对称性)或者为立方体的多个维规定冲突的计算次序，则将这些非对称度量划分成共享相同计算次序的子集，并且生成通过内部连接而组合它们的 SQL 语句。然后，通过内部连接将这些不能通过嵌套而解析的具有不兼容性的非对称度量与对称度量和能够通过嵌套与对称度量组合的任何非对称度量进行组合。

图 29D 示出了根据本发明特定实现的用于使用嵌套来组合不兼容的度量的逻辑的进一步详情。控制在块 2950 中以访问度量集开始。在块 2952 中，以一个度量开始(例如，第一度量)，选择该度量集中的下一个度量。在块 2954 中，确定该度量是否与前面的度量兼容。对于被处理的第一度量，该确定是该度量兼容(因为对于第一度量，不存在前面度量)。如果该度量兼容，则处理继续到块 2960，否则处理继续到块

2956。在块 2956 中，确定是否可以改写一个或多个度量，使得所选的度量与前面的度量兼容。如果可以改写一个或多个度量，则处理继续到块 2958，否则，处理继续到块 2964。在块 2958 中，改写一个或多个度量。然后，确定是否存在要被处理的另一个度量。如果是，则处理循环回到块 2952，否则处理继续到块 2962。在块 2962 中，处理改写后的度量，以生成用于检索多维信息的 SQL 语句。如果所选度量与前面度量不兼容并且不能被改写，则在块 2964 中，通过参照图 29C 所述的技术连接这些度量。

图 29E 示出了根据本发明特定实现的为对称度量集和非对称度量集生成 SQL 语句的逻辑(块 2944、2945)的进一步详情。控制在块 2970 中以这样的操作开始：将度量元数据对象中的一个或多个 SQL 表达式模板的表达式输入到在度量元数据对象中定义的聚合列表中的第一聚合。在块 2972 中，以一个聚合(例如，第一聚合)开始，处理下一个聚合。处理聚合的结果是 SQL 语句。如果该度量包含多个聚合，则使用由第一聚合产生的 SQL 语句作为下一个聚合的输入。这被称作嵌套 SQL 语句。在块 2974 中，确定聚合列表中是否存在要处理的另外聚合。如果是，则处理循环回到块 2972，并且将前面聚合的处理输出输入到下一个聚合中。否则，处理继续到块 2976。在块 2976 中，输出检索多维信息的结构化查询语言语句。在特定实现中，可以将 SQL 语句编目(catalogue)为立方体视图，并且由立方体元数据对象引用。

就计算多个度量而言，由多维元数据软件 120 解决度量的对称性、所涉及聚合函数的分布性以及维出现在聚合脚本中的次序。

关于对称性，对称度量在聚合列表中定义单个聚合，而非对称度量在聚合列表中定义多个聚合。当度量没有定义聚合时，则对称性由基本度量定义。在这一情形下，如果所有其基本度量是对称的，则度量是对称的，而如果任何其基本度量是非对称的，则度量是非对称的。图 30 示出了根据本发明特定实现的表 D 3000，其列出一些度量并且表示哪些度量是对称的或者非对称的。例如，Cost 度量 3002 是对称的，而 Inventory 度量 3004 是非对称的。表 D 3000 没有提供度量的详尽列

表，并且其它度量也可以被定义为对称或非对称。

关于分布性(distributiveness)，当聚合函数可被分解成多个聚合步骤而不改变该聚合函数的结果时，该聚合函数是分布式的(distributive)。例如，对于 SUM，其是分布式的：单个聚合步骤=SUM(2, 8, 11)=21，并且该单个聚合步骤可以分成两个聚合步骤，例如：聚合步骤 1 具有步骤 1a=SUM(2, 8)和步骤 1b=SUM(11)；以及聚合步骤 2=SUM(step1a, step1b)=SUM(10, 11)=21。这示出了 SUM 是分布式函数，因为当单个步骤被分开时，其结果没有改变。当聚合函数不能在不改变聚合函数的结果的情况下分成多个聚合步骤时，该聚合函数是非分布式的。例如，平均(AVG)、标准偏差(STDDEV)和相关(CORRELATION)是非分布式的(non-distributive)。例如，对于 AVG：单个聚合步骤=AVG(2, 8, 11)=7，并且该单个聚合步骤可以分成两个聚合步骤，例如聚合步骤 1 具有步骤 1a=AVG(2, 8)和步骤 1b=AVG(11)；以及聚合步骤 2=AVG(step1a, step1b)=AVG(5, 11)=8。这示出了该平均是非分布式度量，因为当单个聚合步骤被分开时，其结果可能改变。

图 31 示出了根据本发明特定实现的表 E，其列出一些聚合函数，并且表示哪些聚合函数是分布式的以及哪些是非分布式的。例如，SUM 聚合函数 3102 是分布式的，而 AVG 3104 和 CORRELATION 3106 聚合函数是非分布式的。表 E 没有提供聚合函数的详尽列表，并且其它聚合函数可以被分类为分布式或非分布式。

关于维在聚合脚本中出现的次序，理想的是，让事实表的所有度量使用相同数目的聚合步骤，并且让每个聚合步骤计算相同的维集。如果两个聚合步骤使用相同的聚合函数，则可以组合这两个聚合步骤。另外，如果两个聚合函数是分布式的，则可以将该聚合步骤分成两个或更多个聚合步骤。图 32 示出了根据本发明特定实现的表 F 3200，其列出多个度量，以及如何可将聚合步骤分解成用于这些度量的多个聚合步骤。在表 D 3200 中，带星号的数表示计算度量的可选方式。例如，对于 Cost 度量 3202，第一选项是产品、市场和时间维之和，而第二选项是首先执行产品和市场维之和，然后加入时间维。

B.4 为对称度量集生成SQL语句

多维元数据软件 120 为对称度量生成 SQL 语句。为易于理解起见，在这一章节中提供了已经为一些对称度量生成的 SQL 语句。例如，可以在报告中存储执行为总计(Grand Total)查询或任意切片(Arbitrary Slice)查询生成的 SQL 语句的结果。然而，执行为完整立方体查询生成的SQL语句的结果是本身可被查询的立方体视图。

对于所有类型的查询，为每个度量生成 SQL 表达式遵循图 29E 所述的流程。

例如，对于具有对称度量的总计查询，多维元数据软件 120 可以生成以下 Select 语句：

```
select SUM(f.Cost) as Cost, SUM(f.Rev) as Revenue, SUM(f.Rev -
f.Cost) as Profit,
SUM(f.Revenue - f.Cost) / SUM(f.Rev) as "Profit Margin",
RANK () OVER (ORDER BY SUM(f.Rev - f.Cost)) as "Profit Rank",
CORRELATION(f.Revenue, f.Revenue - f.Cost) as "RevProfit
Correlation" from Fact f
```

对于具有对称度量的任意切片查询，多维元数据软件 120 可以生成以下 Select 语句：

```
select SUM(f.Cost) as Cost, SUM(f.Rev) as Revenue, SUM(f.Rev –
f.Cost) as Profit,
SUM(f.Revenue - f.Cost) / SUM(f.Rev) as "Profit Margin",
RANK () OVER (ORDER BY SUM(f.Rev - f.Cost)) as "Profit Rank",
CORRELATION(f.Revenue, f.Revenue - f.Cost) as "RevProfit
Correlation",
m.Country, m.State, t.Year
from Fact f, Market m, Time t
where f.marketid = m.marketid AND f.timeid = t.timeid
group by m.Country, m.State, t.Year
```

对于具有对称度量的完整立方体查询，多维元数据软件 120 可以生

成以下 Select 语句:

```
select SUM(f.Cost) as Cost, SUM(f.Rev) as Revenue, SUM(f.Rev –
f.Cost) as Profit,
SUM(f.Revenue - f.Cost) / SUM(f.Rev) as "Profit Margin",
RANK () OVER (ORDER BY SUM(f.Rev - f.Cost)) as "Profit Rank",
CORRELATION(f.Revenue, f.Revenue - f.Cost) as "RevProfit
Correlation",
m.Country, m.State, m.City,
t.Year, t.Quarter, t.Month,
p.Line, p.Group, p.Product,
from Fact f, Market m, Time t, Product p
where f.marketid = m.marketid AND f.timeid = t.timeid AND f.prodid =
p.prodid
group by   ROLLUP(m.Country, m.State, m.City),
           ROLLUP(t.Year, t.Quarter, t.Month),
           ROLLUP(p.Line, p.Group, p.Product)
```

B.5 为非对称度量集生成 SQL 语句

多维元数据软件 120 为非对称度量生成 SQL 语句。为易于理解起见，在这一章节中提供了已经为一些非对称度量生成的 SQL 语句。例如可以在报告中存储执行为总计查询或任意切片查询生成的 SQL 语句的结果。然而，执行为完整立方体查询生成的 SQL 语句的结果是本身可被查询的立方体视图。

通过使用嵌套 SELECT 语句来计算非对称度量集。每一个聚合步骤映射到在 SELECT 中嵌套的级别。在 SELECT 的最内嵌套级别计算第一聚合步骤，如在图 29E 的块 2970 中所述。使用每个级别的结果作为对直接外部嵌套 SELECT 的输入，如在图 29E 的块 2972 中所述。在这些级别的每一个中，对于尚未被计算且未被期望在给定级别中计算的维，在 GROUP BY 子句中包括粒度最小(最详细)的维属性。以这种方式，对于那些维，不发生聚合。关于聚合函数，每个级别具有以直接内

部级别中的度量结果作为输入的聚合函数。对于最低级别，聚合函数的输入是在度量中定义的 SQL 表达式，如在图 29E 的块 2970 中所述。

例如，对于具有非对称度量的总计查询，多维元数据软件 120 可以生成以下 Select 语句：

```
select AVG(s.Inventory) as Inventory
from (select SUM(f.Inv) as Inventory, f.timeid
from Fact f
group by f.timeid) s
```

对于具有非对称度量的任意切片查询，多维元数据软件 120 可以生成以下 Select 语句：

```
select AVG(s.Inventory) as Inventory,
s.Country, s.State, s.Year
from (select SUM(f.Inv) as Inventory,
F.timeid,
m.Country, m.State, t.Year
from Fact f, Market m, Time t
where f.marketid = m.marketid AND f.timeid = t.timeid
group by f.timeid, m.Country, m.State, t.Year) s
group by s.Country, s.State, s.Year
```

对于具有非对称度量的完整立方体查询，多维元数据软件 120 可以生成以下 Select 语句：

```
select AVG(s.Inventory) as Inventory,
s.Year, s.Quarter, s.Month,
s.Country, s.State, s.City,
s.Line, s.Group, s.Product
from (select SUM(f.Inv) as Inventory, f.timeid, m.Country,
m.State, m.City,
t.Year, t.Quarter, t.Month, p.Line, p.Group, p.Product
from Fact f, Market m, Time t, Product p
```

```
where f.marketid = m.marketid AND f.timeid = t.timeid AND
f.prodid = p.prodid
group by f.timeid, t.Year, t.Quarter, t.Month,
ROLLUP(m.Country, m.State, m.City),
ROLLUP(p.Line, p.Group, p.Product) ) s
group by ROLLUP(s.Year, s.Quarter, s.Month),
s.Country, s.State, s.City, s.Line, s.Group, s.Product
```

B.6 为不兼容的度量集生成 SQL 语句

这一章节描述如何将为多个度量集(例如，对称和非对称度量集)生成的多个 SQL 语句组合成单个 SQL 语句。

在表 F 的例子中，不可能找到维集的公共集。存在两个选项来计算所有(即对称和非对称)度量。每个选项创建两个单独的 SQL 查询，并且将它们合并在一起。例如，第一选项是获得(take)所有维("AllDim")的 Cost、Revenue、Profit、Profit Margin、Profit Rank 和 RevProfit Correlation、以及 AllbutTime(除了时间的所有维)、Time 的 Inventory。例如，第二选项是获得所有维("AllDim")的 RevProfit Correlation，以及 AllbutTime、Time 的 Inventory、Cost、Revenue、Profit、Profit Margin、Profit Rank。

当为相同的切片或立方体生成这些 SQL 语句时，为对称和非对称度量集生成的多个 SQL 语句共享相同的属性集。因此，属性实例在所有 SQL 语句中都将是相同的。组合不同度量集的 SQL 语句的技术包括连接两个 SQL 语句的结果。也就是，通过用内部连接(INNER JOIN)连接为每个度量集生成的 SQL 语句,将它们连接成单个 SQL 语句。在特定实现中，所使用的连接类型是对在 SQL 语句的 GROUP BY 子句中使用的属性的内部连接。

在多个 SQL 语句之间的内部连接中使用的子句取决于正被组合的 SQL 语句的类型(即，基于切片的相对于完整立方体)。对于基于切片的 SQL 语句，在内部连接中使用的子句将使用该切片中的所有属性的简单与(ANDed)等式。下面是针对第一选项由多维元数据软件 120 生成的基

于切片的 SQL 语句:

```
select r1.Inventory, r2.Cost, r2.Revenue, r2.profit,
                                                    r2.Margin,
                                                    r2."profit
Rank", r2."RevProfit Correlation",
r1.Country, r1.State, r1.Year
(select AVG(s.Inventory) as Inventory,
s.Country, s.State, s.Year
from (select SUM(f.Inv) as Inventory,
f.timeid, m.Country, m.State, t.Year
from Fact f, Market m, Time
where f.marketid = m.marketid AND f.timeid = t.timeid
group by f.timeid, m.Country, m.State, t.Year) s
group by s.Country, s.State, s.Year) r1
INNER JOIN
(select   SUM(f.Cost) as Cost, SUM(f.Rev) as Revenue,
                                                    SUM(f.Rev
 - f.Cost) as profit
SUM(f.Revenue - f.Cost) / SUM(f.Rev) as Margin,
RANK () OVER (ORDER BY SUM(f.Rev - f.cost)) as "Profit Rank",
CORRELATION(f.Revenue, f.Revenue - f.Cost) as "RevProfit
Correlation",
m.Country, m.State, t.Year from   Fact f, Market m, Time t
where f.marketid = m.marketid AND f.timeid = t.timeid
group by m.Country, m.State, t.Year) r2
ON r1.Country = r2.Country AND r1.State = r2.State AND
                                                    R1.Year =
r2.Year
```

在总计切片中，仅返回所有维的总计。如果正被查询的切片是总计

切片，则没有正被编组的属性，因此，在连接子句中使用暂时常量属性(transient constant attribute)。暂时常量属性与 Grand Total(总计)列相关联，并且描述是否为给定属性保存了聚合。下面是针对第一选项由多维元数据软件 120 生成的总计切片 SQL 语句：

```
select   r1.Inventory,
r2.Cost, r2.Revenue, r2.profit, r2.Margin, r2."profit Rank",
                                                    r2."RevProfit
Correlation"
(select AVG(s.Inventory) as Inventory, l as GrandTotal
from (select SUM(f.Inv) as Inventory, f.timeid
from Fact f
group by f.timeid) s
) r1
INNER JOIN
(select SUM(f.Cost) as Cost, SUM(f.Rev) as Revenue, SUM(f.Rev-
f.Cost) as profit,
SUM(f.Revenue - f.Cost) / SUM(f.Rev) as "Profit Margin",
RANK () OVER (ORDER BY SUM(f.Rev - f.cost)) as "Profit Rank",
CORRELATION(f.Revenue, f.Revenue - f.Cost) as "RevProfit
Correlation",
1 as GrandTotal
from Fact f) r2
ON r1.GrandTotal = r2.GrandTotal
```

对于完整立方体类型的 SQL 语句，连接子句还考虑属性实例也将包含聚合表示这一事实。由于此原因，将暂时属性添加到正被组合的基本 SQL 语句。这些新的暂时常量属性描述是否为给定属性保存了聚合。然后，当属性实例包含特定成员或者包含聚合时，连接子句连接这些属性实例。下面是针对第一选项由多维元数据软件 120 生成的基于立方体的 SQL 语句，其示出了使用以 AGG(即聚合)为后缀的暂时常量属

性:

```
select r1.Inventory, r2.Cost, r2.Revenue, r2.profit,
                                                    r2.Margin,
                                                    r2."profit
Rank", r2."RevProfit Correlation",
r1.Year, r1.Quarter, r1.Month, r1.Country, r1.state, r1.City,
                                                      r1.Line,
r1.Group, r1.Product
(select AVG(s.Inventory) as Inventory, s.Year, s.Quarter,
                                                   s.Month,
s.Country, s.State, s.City, s.Line, s.Group, s.Product,
      GROUPING(s.Year) as YearAgg, GROUPING(s.Quarter) as
QuarterAgg, GROUPING(s.Month) as MonthAgg,
s.CountryAgg, s.StateAgg, s.CityAgg, s.LineAgg, s.GroupAgg,
s.ProductAgg
from   (select SUM(f.Inv) as Inventory, f.timeid, t.Year,
                                                  t.Quarter,
t.Month, m.Country, m.State, m.City, p.Line, p.Group,
                                                  p.Product,
GROUPING(m.Country) as CountryAgg, GROUPING(m.State) as
                                                      StateAgg,
GROUPING(m.City) as CityAgg,
GROUPING(p.Line) as LineAgg, GROUPING(p.Group) as GroupAgg,
GROUPING(p.Product) as ProductAgg
            from   Fact f, Market m, Time t, Product p
            where   f.Marketid = m.marketid AND f.timeid =
                                                    t.timeid AND
f.pordid = p.prodid
group by f.timeid, t.Year, t.Quarter, t.Month,
```

```
ROLLUP(m.Country, m.State, m.City), ROLLUP(p.Line, p.Group,
p.Product) ) s
group by ROLLUP(s.Year, s.Quarter, s.Month), s.Country,
                                                    s.State,
s.City, s.Line, s.Group, s.Product,
s.CountryAgg, s.StateAgg, s.CityAgg, s.LineAgg, s.GroupAgg,
s.ProductAgg) r1
INNER JOIN
(select SUM(f.Cost) as Cost, SUM(f.Rev) as Revenue,
                                                SUM(f.Rev
 - .Cost) as Profit,
SUM(f.Revenue - f.Cost) / SUM(f.Rev) as "Profit Margin",
RANK () OVER (ORDER BY SUM(f.Rev - f.Cost)) as "Profit Rank",
                CORRELATION(f.Revenue, f.Revenue - f.Cost) as
                                                    "RevProfit
Correlation",
                m.Country, m.State, m.City, t.Year, t.Quarter,
                                                    t.Month, p.Line,
p.Group, p.Product,
          GROUPING(m.Country) as CountryAgg,
                                                GROUPING(m.State)
as StateAgg, GROUPING(m.City) as CityAgg,
           GROUPING(t.Year) as YearAgg, GROUPING(t.Quarter)
                                                    as
QuarterAgg, GROUPING(t.Month) as MonthAgg,
            GROUPING(p.Line) as LineAgg, GROUPING(p.Group) as
GroupAgg, GROUPING(p.Product) as ProductAgg
From Fact f, Market m, Time t, Product p
where f.marketid = m.marketid AND f.timeid = t.timeid AND
```

```
                                                    f.prodid =
p.prodid
group by ROLLUP(m.Country, m.State, m.City), ROLLUP(t.Year,
t.Quarter, t.Month), ROLLUP(p.Line, p.Group, p.Product)) r2
ON (r1.Country = r2.Country OR (r1.CountryAgg = 1 AND
r2.CountryAgg = 1)) AND
          (r1.State = r2.State OR (r1.StateAgg = 1 AND
                                                    r2.StateAgg = 1))
                                                    AND
(r1.City = r2.City OR (r1.CityAgg = 1 AND r2.CityAgg = 1)) AND
     (r1.Year = r2.Year OR (r1.YearAgg = 1 AND r2.YearAgg = 1))
                                                    AND
     (r1.Quarter = r2.Quarter OR (r1.QuarterAgg=1 AND
r2.QuarterAgg=1)) AND
     (r1.Month = r2.Month OR (r1.MonthAgg=1 AND
                                                    r2.MonthAgg=1))
AND
     (r1.Line = r2.Line OR (r1.LineAgg=1 AND r2.LineAgg=1))
                                                    AND
(r1.Group = r2.Group OR (r1.GroupAgg=1 AND r2.GroupAgg=1))
AND
     (r1.Product = r2.Product OR (r1.ProductAgg=1 AND
r2.ProductAgg=1))
```

虽然上面例子示出了两个度量集，但是本发明的实现可以组合多于两个度量集。而且，虽然这里的例子涉及 SQL 语句，但是可用来访问数据库的其它语句也在本发明的范围之内。

在特定实现中，不是组合为多个度量集生成的 SQL 语句，而是重构聚合集，使得它们是兼容的。例如，可以组合 Sales 和 Inventory 度量的计算。由于 Sales 度量对所有维使用 SUM，而 Inventory 度量对除

了时间之外的所有维使用 SUM，因此 Sales 度量的计算可以分成两个步骤。第一步骤是对除了时间之外的所有维的 SUM，并且最后步骤是对时间的 SUM(由于 SUM 是分布式的，因此这起作用)。Sales 的这一计算次序现在与 Inventory 所需的步骤兼容。

IBM、DB2、Z/OS 和 AIX 是国际商业机器公司在美国和/或其它国家的商标。Windows 是 Microsoft 公司在美国和/或其它国家的商标。Solaris 和 JDBC 是 SUN Microsystems 在美国和/或其它国家的商标。Linux 是 Linus Torvalds 在美国和/或其它国家的商标。HP-UX 是在美国和/或其它国家的 Open Group UNIX 95 有商标产品。Pilot Suite 是 Pilot Software 在美国和/或其它国家的商标。Express 是 Oracle 公司在美国和/或其它国家的商标。Essbase 是 Hyperion Solution 公司在美国和/或其它国家的商标。TM1 是 Applix 公司在美国和/或其它国家的商标。

另外的实现细节

所述的用于维护关于网络组件的信息的技术可被实现为使用标准编程和/或工程技术以产生软件、固件或其任意组合的方法、设备或产品。在此使用的术语“产品”是指在硬件逻辑电路(例如，集成电路芯片、可编程门阵列(PGA)、专用集成电路(ASIC)等等)或者计算机可读介质例如磁存储介质(例如，硬盘驱动器、软盘、磁带等等)、光存储装置(CD-ROM、光盘等)、易失性和非易失性存储器装置(例如，EEPROM、ROM、PROM、RAM、DRAM、SRAM、固件、可编程逻辑电路等等)中实现的代码或逻辑。计算机可读介质中的代码由处理器访问和执行。其中实现了优选实施例的代码还可通过传输介质或从网络上的文件服务器访问。在这种情况下，其中实现了该代码的产品可以包括传输介质如网络传输线、无线传输介质，通过空间传播的信号、无线电波、红外线信号等等。因此，该“产品”可以包括其中实施了代码的介质。另外，该“产品”可以包括在其中实施、处理和执行代码的硬件和软件组件的组合。当然，本领域的技术人员应当认识到，在不脱离本发明范围的情况下，可以对该配置进行各种修改，并且该产品可以包括在

本技术领域内公知的任何信息承载介质。

图 29A、29B、29C、29D 和 29E 的逻辑描述了以特定次序发生的特定操作。在可选的实现中，可以以不同的次序执行、修改或删除某些逻辑操作。而且，可以向上述逻辑添加步骤，并且仍然遵循所述实现。此外，可以顺序地发生在此描述的操作，或者可以并行处理某些操作，或者可以通过分布式过程执行被描述成由单个过程执行的操作。

图 29A 、29B、29C、29D 和 29E 的逻辑被描述成用软件实现。该逻辑可以是主机系统的操作系统的一部分或者应用程序。在另外的实现中，可以将该逻辑保持在存储区或者只读存储器或者其它硬连线类型的装置中。优选的逻辑可以用硬件或者可编程和不可编程门阵列逻辑电路实现。

图 33 示出了计算机系统 100 的体系结构的一种实现。计算机系统 100 可以实现具有处理器 3302(例如，微处理器)、存储器 3304(例如，易失性存储器装置)和存储装置 3306(例如，非易失性存储区，如磁盘驱动器、光盘驱动器、磁带驱动器等)的计算机体系结构 3300。存储装置 3306 可以包括内部存储装置或附连或网络可访问存储装置。以在本技术领域内公知的方式将存储装置 3306 中的程序装载到存储器 3304 中，并且由处理器 3302 执行。该体系结构还包括用来支持与网络的通信的网卡 3308。输入装置 3310 用来向处理器 3303 提供用户输入，并且可包括键盘、鼠标、输入笔、麦克风、触摸式显示屏或者在本技术领域内公知的任何其它激活或输入机制。输出装置 3312 能够呈现从处理器 3302 或其它组件传送的信息，例如显示监视器、打印机、存储装置等。

前面对本发明的优选实现的描述是为了示例说明和描述的目的而提供的。它并不旨在是详尽的或者将本发明局限于所公开的确切形式。根据上述教导，各种修改和变更是可能的。

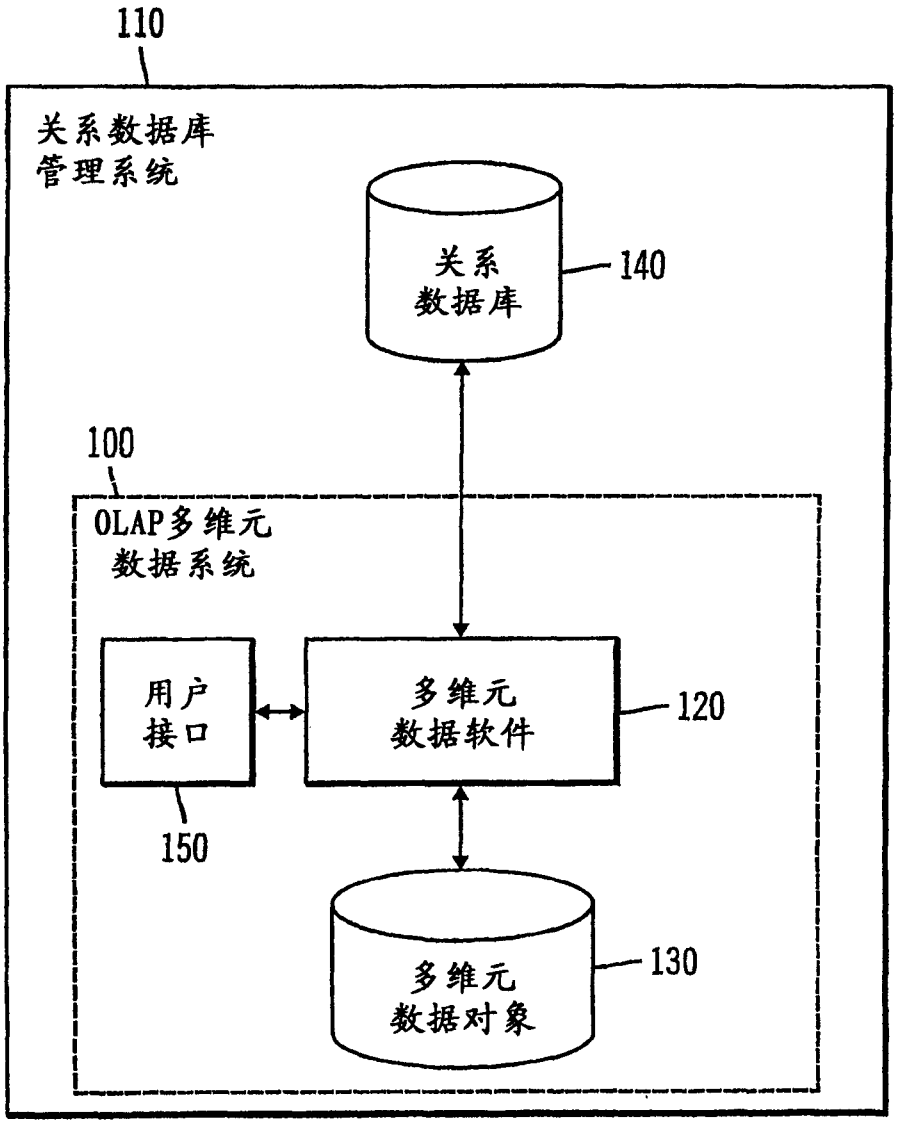

110
关系数据库
管理系统
关系
数据库
140
100
OLAP多维元
数据系统
用户
接口
多维元
数据软件
120
150
多维元
数据对象
130

图1

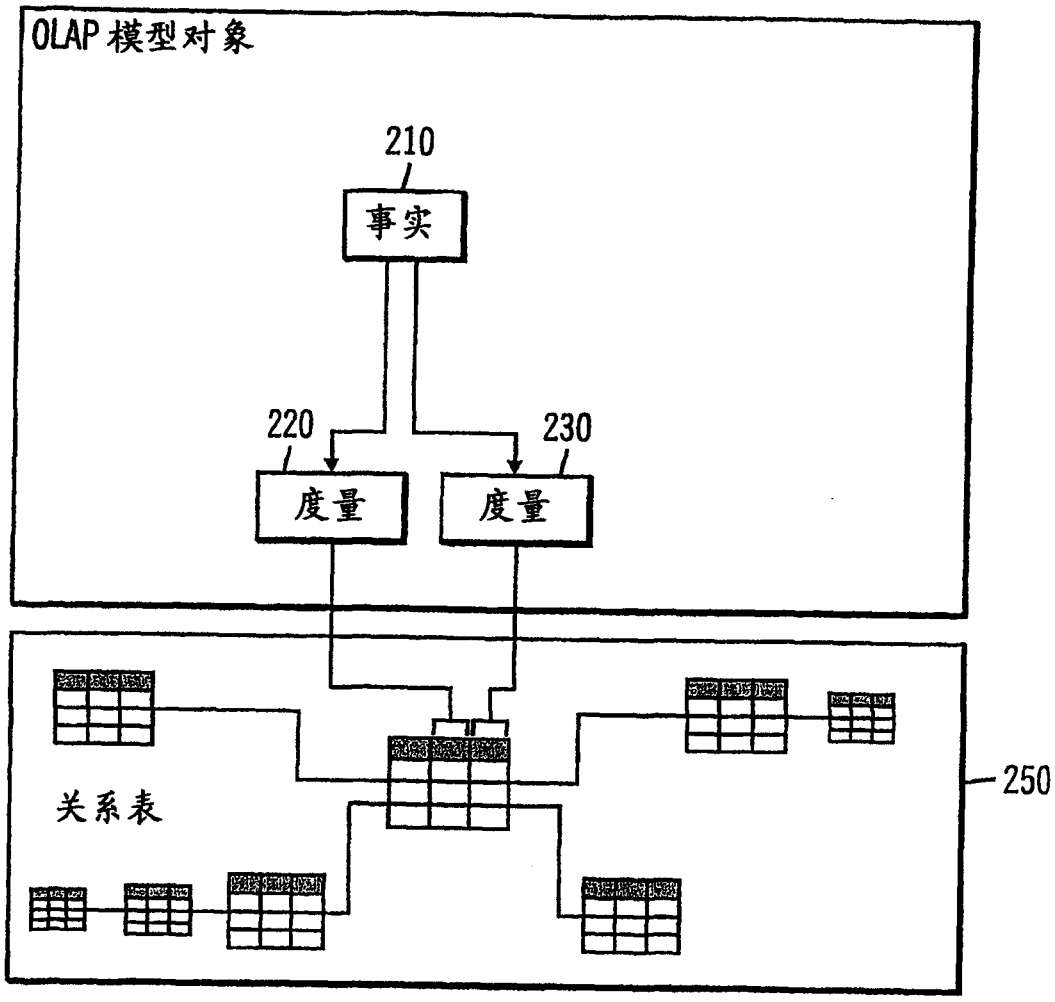


图 2

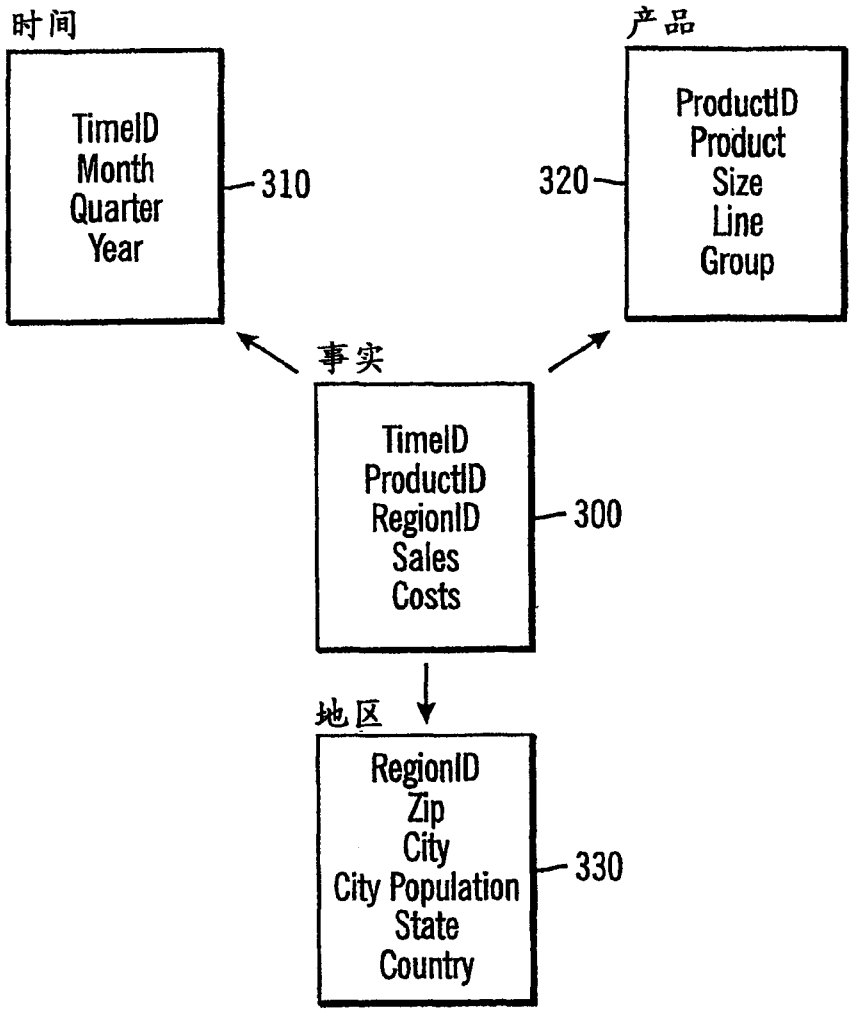

时间
TimeID
Month
Quarter
Year
310
产品
ProductID
Product
Size
Line
Group
320
事实
TimeID
ProductID
RegionID
Sales
Costs
300
地区
RegionID
Zip
City
City Population
State
Country
330

图 3

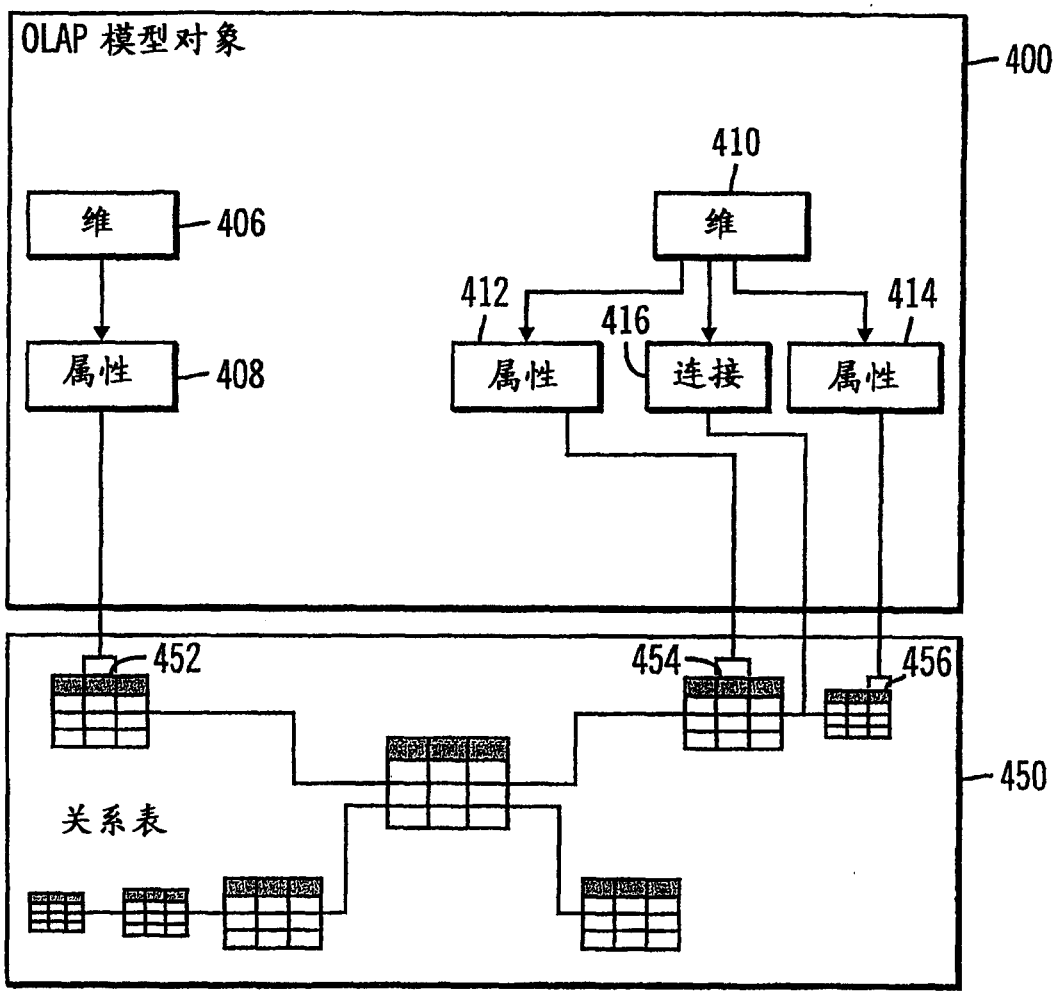

OLAP 模型对象
400
410
维
406
维
412
414
416
属性
408
属性
连接
属性
452
454
456
450
关系表

图4

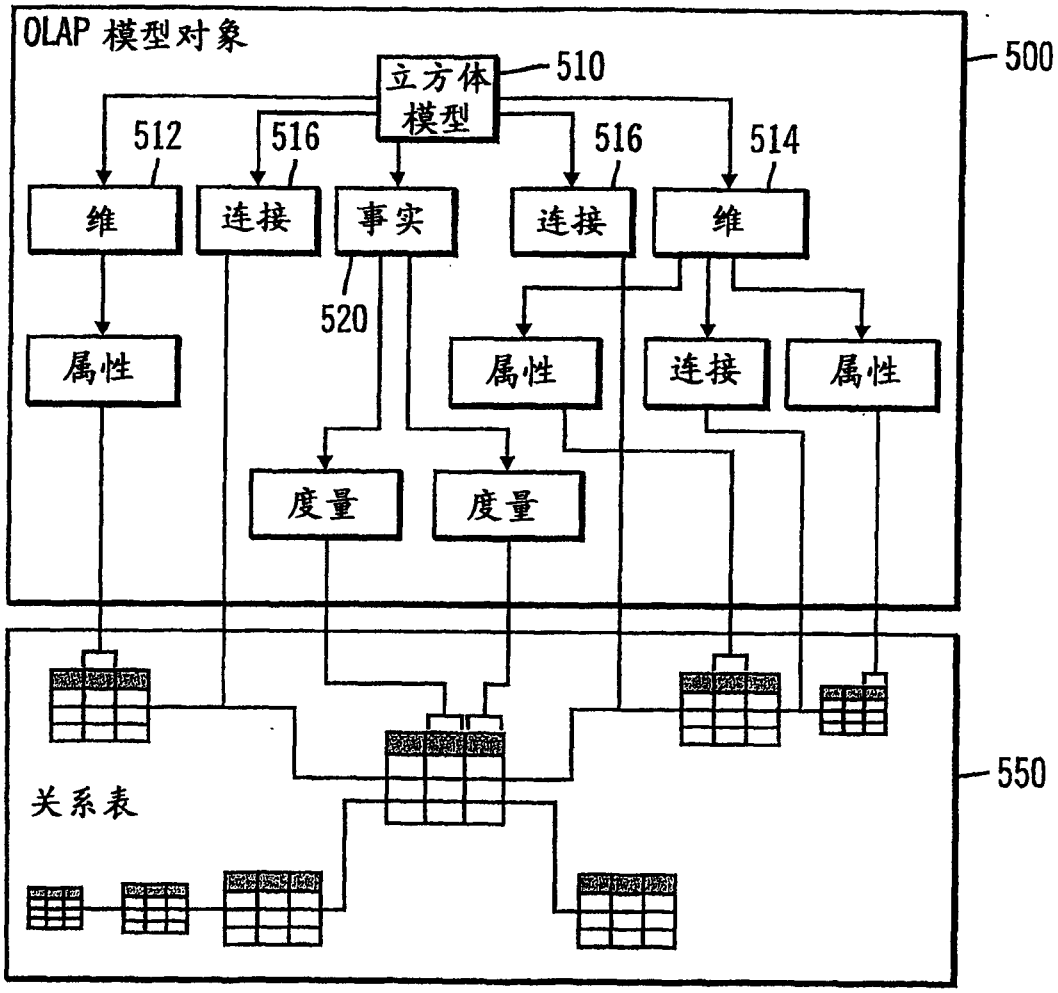

OLAP 模型对象
500
立方体模型
510
512
516
516
514
维
连接
事实
连接
维
520
属性
属性
连接
属性
度量
度量
关系表
550

图5

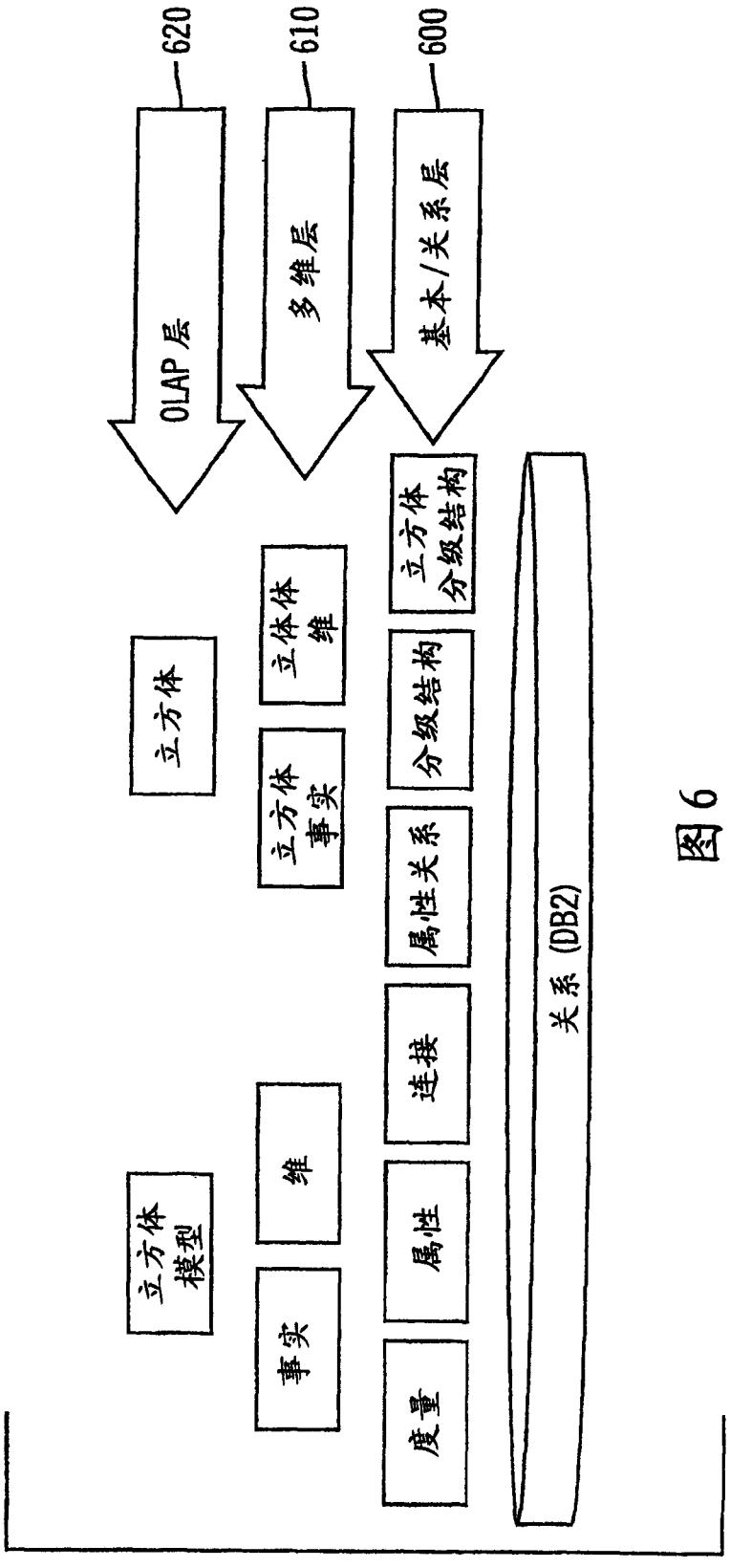

620
610
600
OLAP层
多维层
基本/关系层
立方体
立方体体维
立方体分级结构
立方体事实
分级结构
属性关系
立方体模型
事实
维
连接
属性
度量
关系 (DB2)

图6

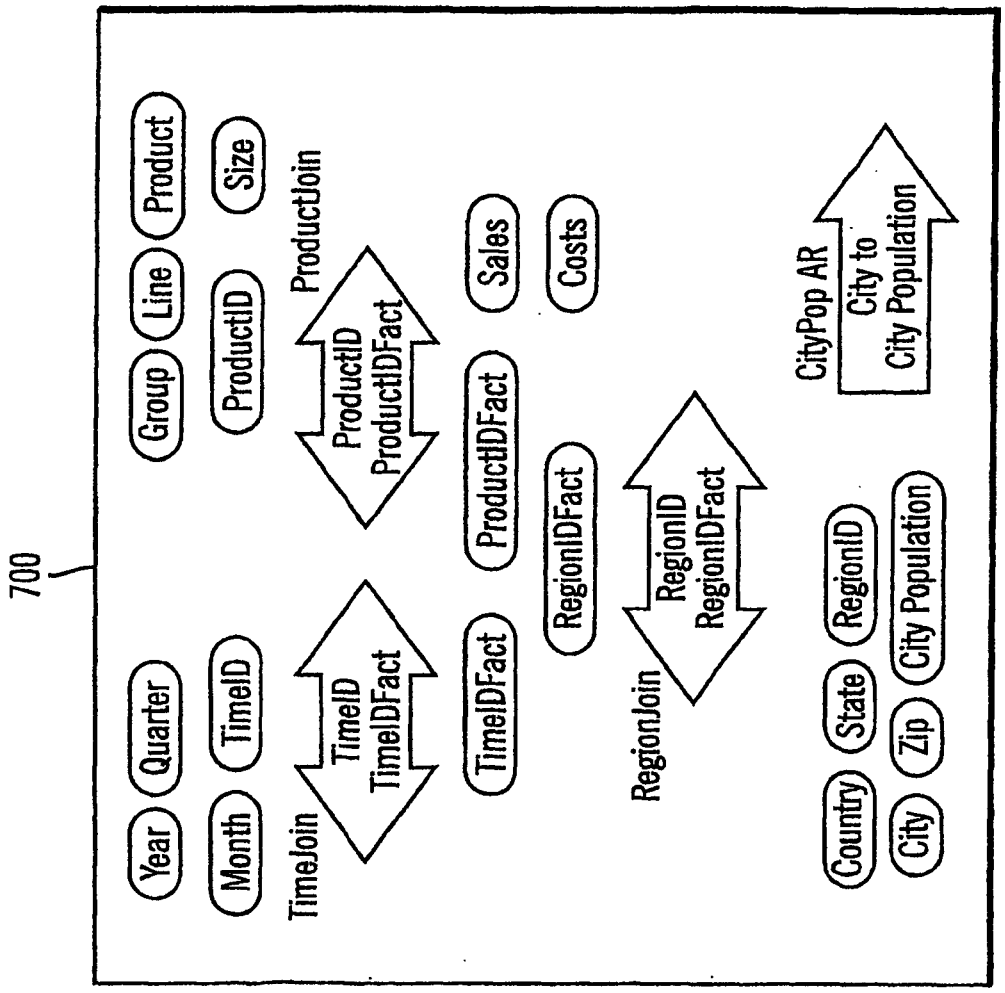

700
Year
Quarter
Month
TimeID
TimeJoin
TimeID
TimeIDFact
TimeIDFact
ProductIDFact
RegionIDFact
RegionJoin
RegionID
RegionIDFact
Group
Line
Product
ProductID
Size
ProductJoin
ProductID
ProductIDFact
Sales
Costs
CityPop AR
City to
City Population
Country
State
RegionID
City
Zip
City Population

图 7

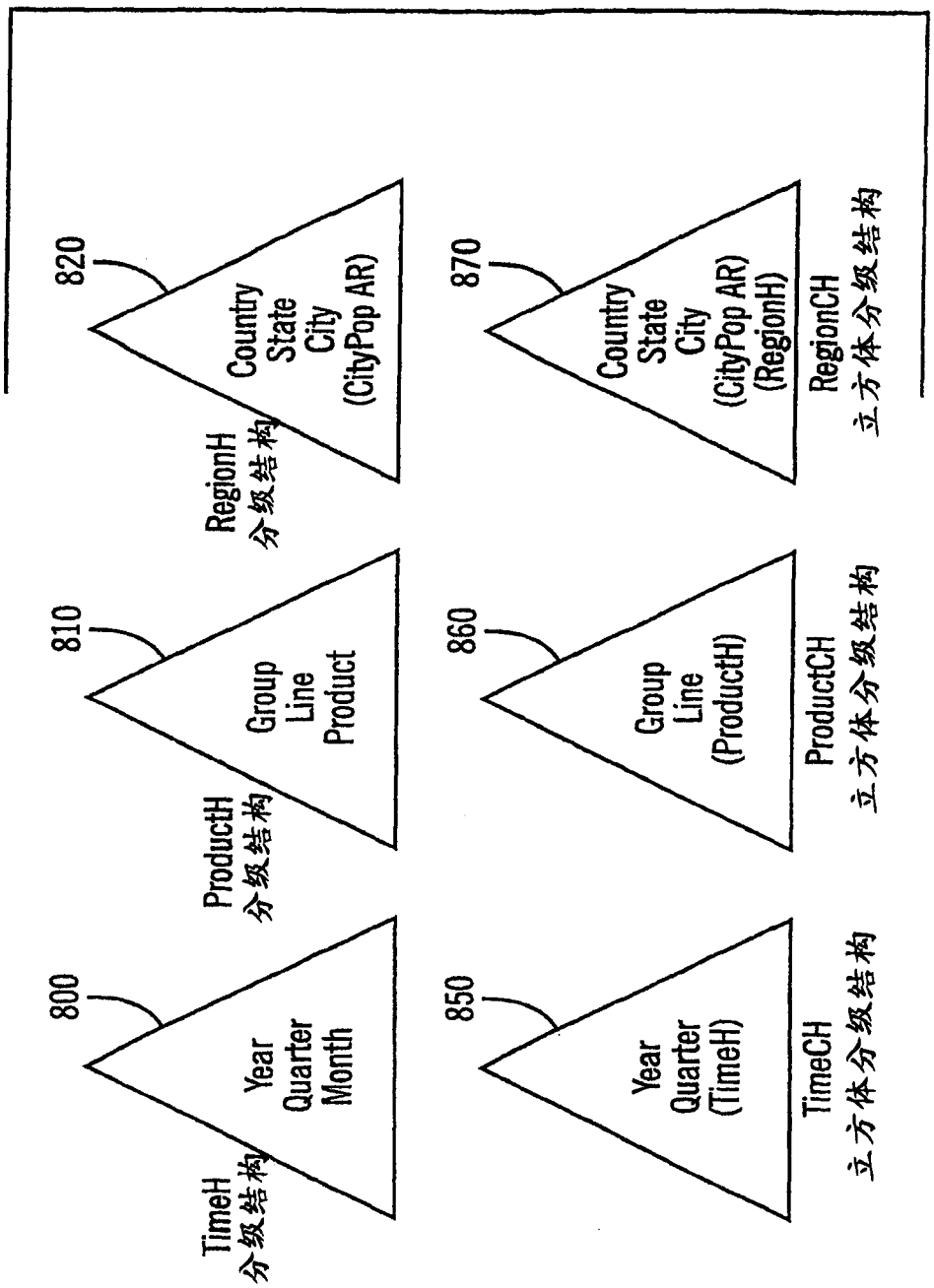

800
Year
Quarter
Month
TimeH
分级结构
810
Group
Line
Product
ProductH
分级结构
820
Country
State
City
(CityPop AR)
RegionH
分级结构
850
Year
Quarter
(TimeH)
TimeCH
立方体分级结构
860
Group
Line
(ProductH)
ProductCH
立方体分级结构
870
Country
State
City
(CityPop AR)
(RegionH)
RegionCH
立方体分级结构

图 8

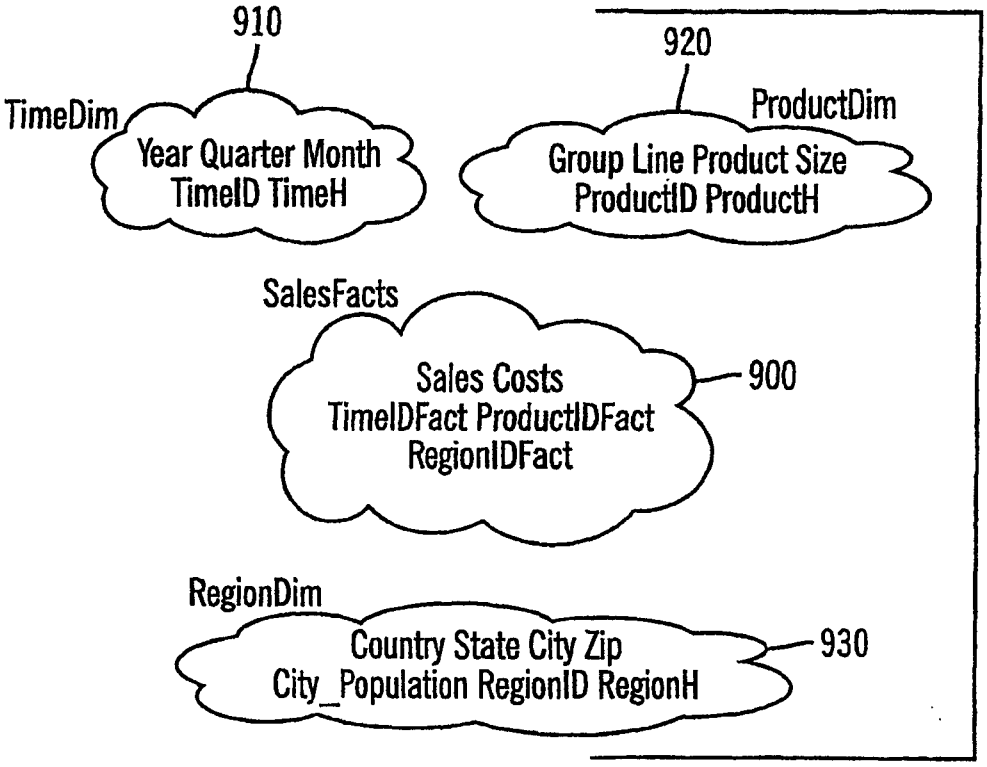

910
TimeDim
Year Quarter Month
TimeID TimeH
920
ProductDim
Group Line Product Size
ProductID ProductH
SalesFacts
Sales Costs
TimeIDFact ProductIDFact
RegionIDFact
900
RegionDim
Country State City Zip
City_Population RegionID RegionH
930

图9

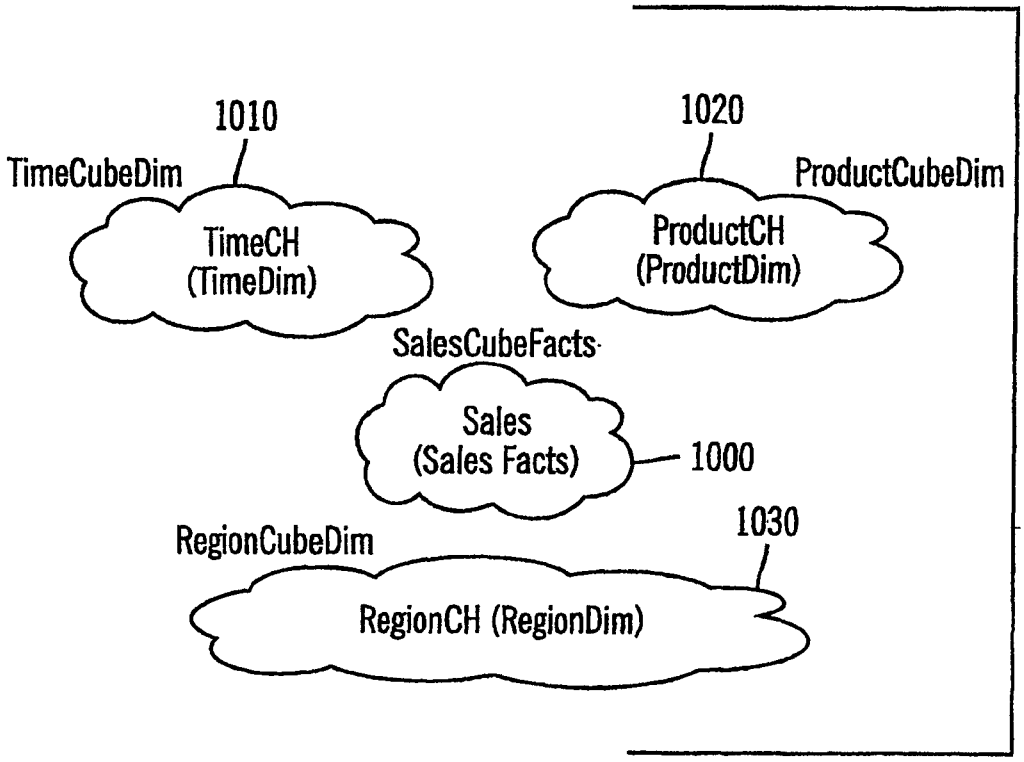

1010
TimeCubeDim
TimeCH
(TimeDim)
1020
ProductCubeDim
ProductCH
(ProductDim)
SalesCubeFacts
Sales
(Sales Facts)
1000
RegionCubeDim
1030
RegionCH (RegionDim)

图10

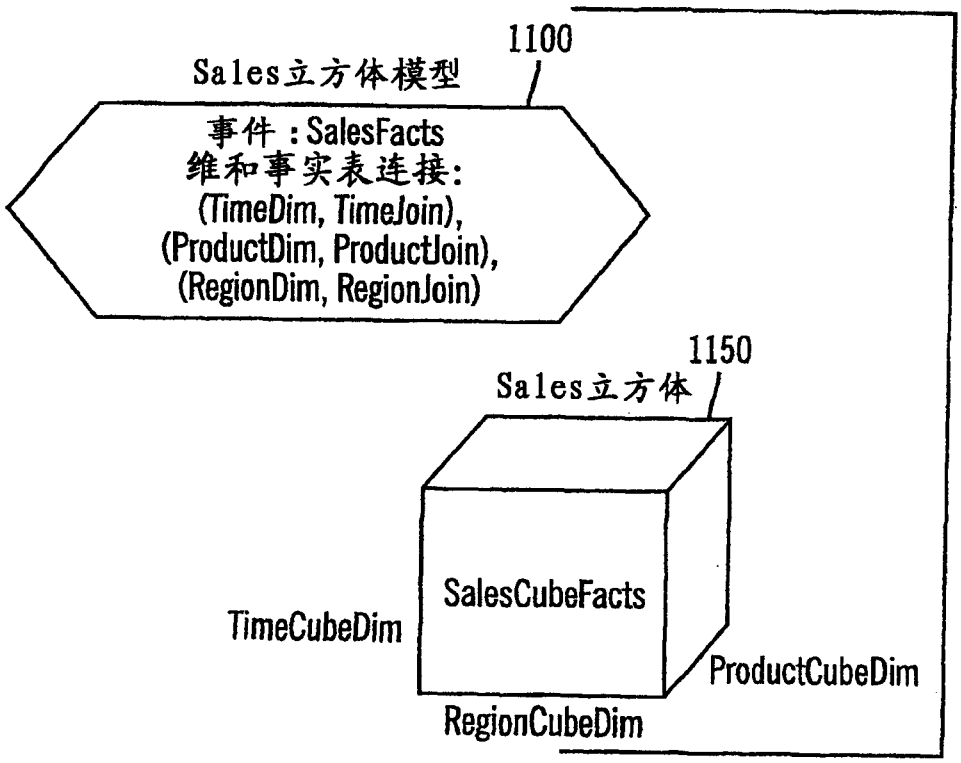

1100
Sales立方体模型
事件：SalesFacts
维和事实表连接：
(TimeDim, TimeJoin),
(ProductDim, ProductJoin),
(RegionDim, RegionJoin)
1150
Sales立方体
SalesCubeFacts
TimeCubeDim
ProductCubeDim
RegionCubeDim

图11

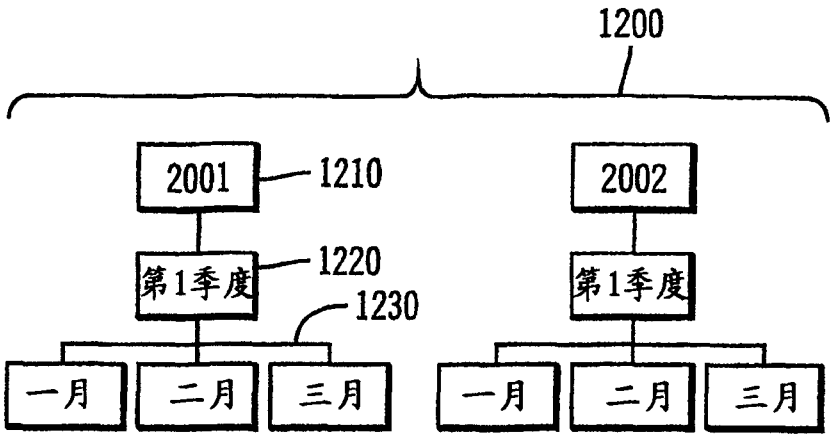

1200
2001
1210
2002
第1季度
1220
1230
第1季度
一月
二月
三月
一月
二月
三月

图12

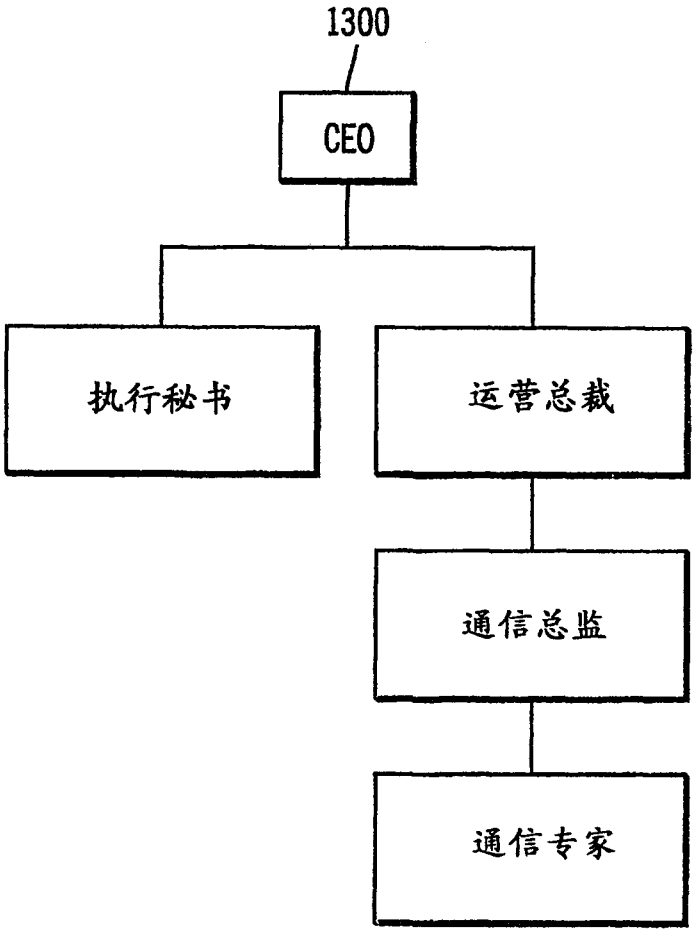

1300
CEO
执行秘书
运营总裁
通信总监
通信专家

图13

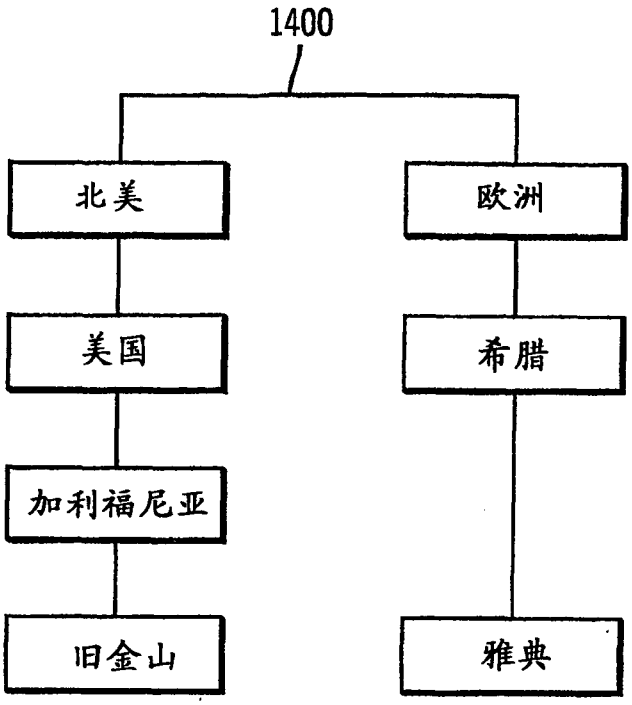


图14

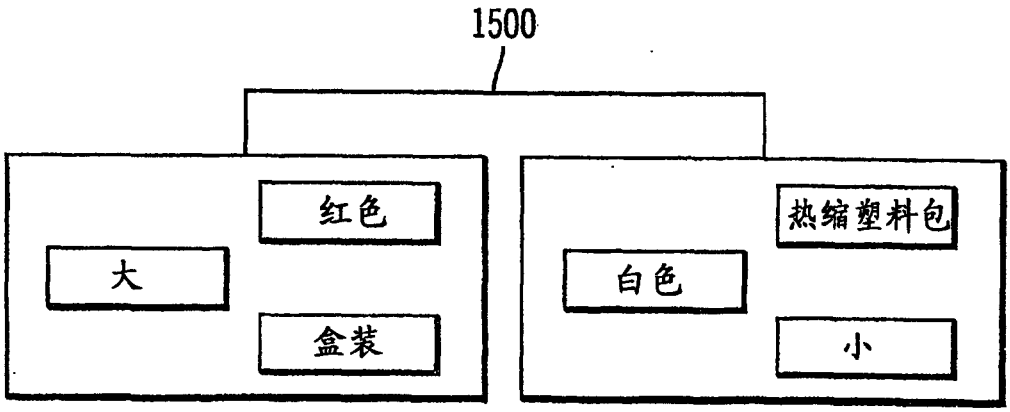


图15

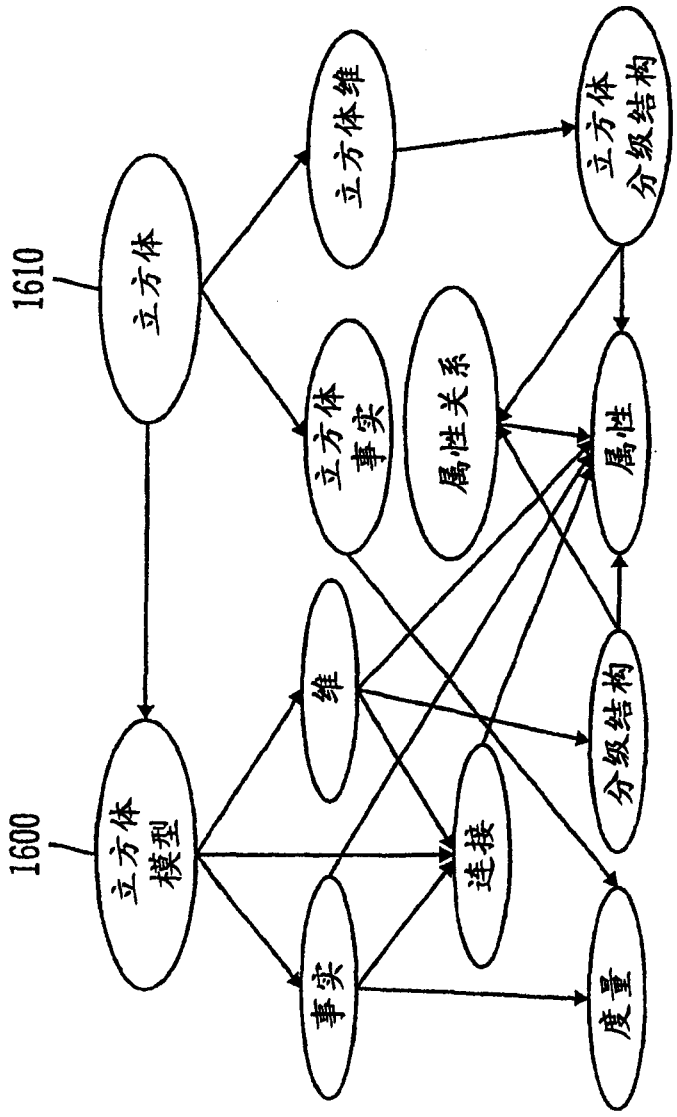


图16

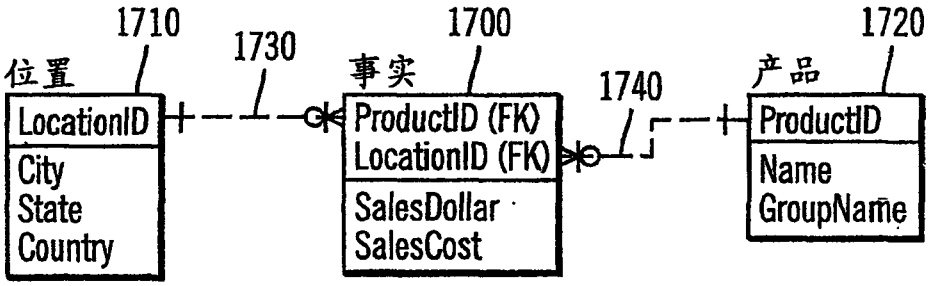

1710
1730
1700
1740
1720
位置
事实
产品
LocationID
City
State
Country
ProductID (FK)
LocationID (FK)
SalesDollar
SalesCost
ProductID
Name
GroupName

图17

立方体模型：

1800

属性	值
名称	LocationProduct
事实	SalesFacts
维集	(Location, LocationFacts), (Product, ProductFacts)

立方体：

1802

属性	值
名称	LocationProduct
立方体模型	LocationProduct
立方体事实	SalesCFacts
立方体维列表	LocationCD, ProductCD

事实：

1804

属性	值
名称	SalesFacts
度量集	Revenue, Profit
属性集	ProductID_Facts, LocationID_Facts
连接集	<no joins used>

立方体事实

1806

属性	值
名称	SalesCFacts
事实	SalesFacts
度量集	Revenue, Profit

度量

1808

属性	值
名称	Revenue
SQL表达式模板列表	"{$$1}"
列和度量的列表	*Column*: Facts.SalesDollar
聚合列表	(SUM, <empty>)
数据类型	DOUBLE

1810

属性	值
名称	Profit
SQL表达式模板列表	"{$$1} - {$$2}"
列和度量的列表	*Measure*: Revenue, *Column*: Facts.SalesCost
聚合列表	(SUM, <empty>)
数据类型	DOUBLE

图18A

维：

1812

属性	值
名称	Location
属性集	LocationID, Country, State, City
连接集	<未使用连接>
分级结构集	LocDetail, LocOverview
类型	常规

1814

属性	值
名称	Product
属性集	ProductID, GroupName, ProdName
连接集	<未使用连接>
分级结构集	Product
类型	常规

立方体维

1816

属性	值
名称	LocationCD
维	Location
立方体分级结构	LocOverviewCH

1818

属性	值
名称	ProductCD
维	Product
立方体分级结构	ProductCH

分级结构

1820

属性	值
名称	Loc Detail
属性列表	Country, State, City
属性关系集	<未使用属性关系>
类型	平衡
部署	标准

1822

属性	值
名称	LocOverview
属性列表	Country, State
属性关系集	<未使用属性关系>
类型	平衡
部署	标准

图18B

1824

属性	值
名称	Product
属性列表	GroupName, ProdName
属性关系集	<未使用属性关系>
类型	平衡
部署	标准

立方体分级结构

1826

属性	值
名称	LocOverviewCH
分级结构	LocOverview
属性列表	Country, State
属性关系集	<未使用属性关系>

1828

属性	值
名称	ProductCH
维	ProductCD
分级结构	Product
属性列表	GroupName, ProdName
属性关系集	<未使用属性关系>

连接:

1830

属性	值
名称	LocationFacts
连接三元组	(LocationID, LocationID_Facts, =)
类型	内部

1832

属性	值
名称	ProductFacts
连接三元组	(ProductID, ProductID_Facts, =)
类型	内部

图18C

属性

1834

属性	值
名称	City
列和属性的列表	*Column*: Location.City
SQL表达式模板	"{$$1}"
数据类型	VARCHAR(40)
角色	级别

1836

属性	值
名称	State
列和属性的列表	*Column*: Location.State
SQL表达式模板	"{$$1}"
数据类型	VARCHAR(40)
角色	级别

1838

属性	值
名称	ProdName
列和属性的列表	*Column*: Product.Name
SQL表达式模板	"{$$1}"
数据类型	VARCHAR(40)
角色	级别

1840

属性	值
名称	GroupName
列和属性的列表	*Column*: Product.GroupName
SQL表达式模板	"{$$1}"
数据类型	VARCHAR(40)
角色	级别

1842

属性	值
名称	ProductID
列和属性的列表	*Column*: Product.ProductID
SQL表达式模板	"{$$1}"
数据类型	INTEGER
角色	维键

1844

属性	值
名称	ProductID_Facts
列和属性的列表	*Column*: Facts.ProductID
SQL表达式模板	"{$$1}"
数据类型	INTEGER
角色	维键

图18D

1846

属性	值
名称	LocationID
列和属性的列表	*Column*: Location.LocationID
SQL表达式模板	"{$$1}"
数据类型	INTEGER
角色	维键

1848

属性	值
名称	LocationID_Facts
列和属性的列表	*Column*: Facts.LocationID
SQL表达式模板	"{$$1}"
数据类型	INTEGER
角色	维键

图18E

1900

表 A

	1910 单价	1912 收益	1914 利润
1902 — 裤子	40.00	680.00	68.00
1904 — 衬衫	60.00	780.00	117.00
1906 — 领带	25.00	175.00	52.50

图 19

2000
表B
2004
一月
二月
三月
2006
季度1
圣何塞
10
8
15
8+
洛杉矶
18
22
10
10+
2002—加利福尼亚
28*
30*
25*
25+

图20

2100
表 C
一月
二月
三月
季度1
裤子
SJ = 10
LA = 18
CA =
SJ = 8
LA = 22
CA =
SJ = 15
LA = 10
CA =
SJ =
LA =
CA =
衬衫
SJ = 22
LA = 36
CA =
SJ = 27
LA = 28
CA =
SJ = 24
LA = 31
CA =
SJ =
LA =
CA =
领带
SJ = 8
LA = 12
CA =
SJ = 11
LA = 7
CA =
SJ = 6
LA = 13
CA =
SJ =
LA =
CA =
衣服
SJ =
LA =
CA =
SJ =
LA =
CA =
SJ =
LA =
CA =
SJ =
LA =
CA =

图 21A

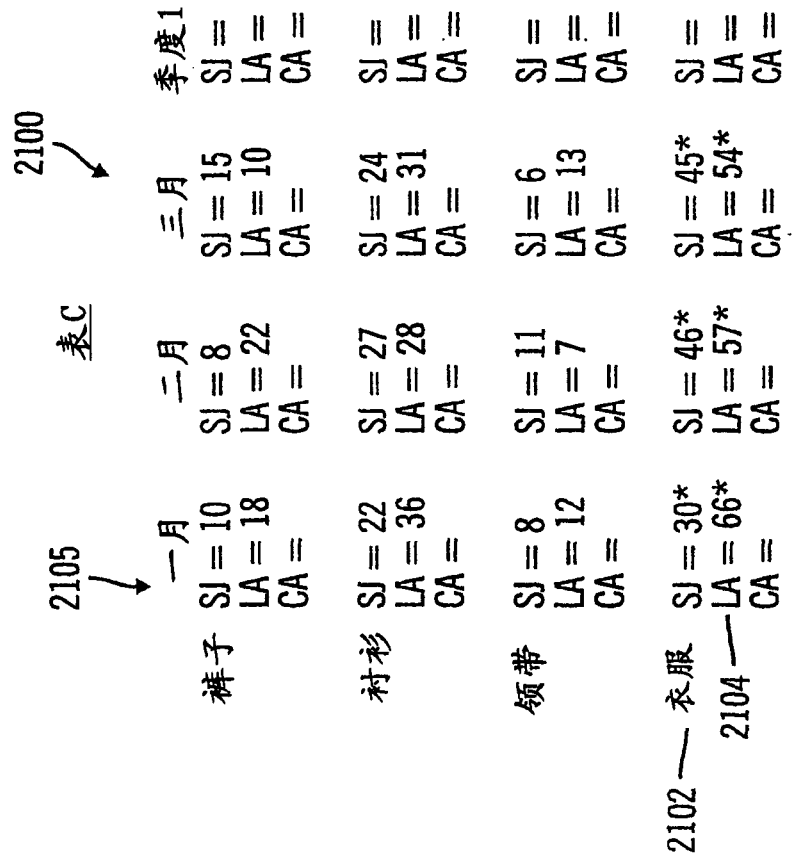

2100
2105
表C
一月
二月
三月
季度1
裤子
SJ = 10
LA = 18
CA =
SJ = 8
LA = 22
CA =
SJ = 15
LA = 10
CA =
SJ =
LA =
CA =
衬衫
SJ = 22
LA = 36
CA =
SJ = 27
LA = 28
CA =
SJ = 24
LA = 31
CA =
SJ =
LA =
CA =
领带
SJ = 8
LA = 12
CA =
SJ = 11
LA = 7
CA =
SJ = 6
LA = 13
CA =
SJ =
LA =
CA =
2102 — 衣服
2104 —
SJ = 30*
LA = 66*
CA =
SJ = 46*
LA = 57*
CA =
SJ = 45*
LA = 54*
CA =
SJ =
LA =
CA =

图21B

2100

表C

2106	2105 一月	二月	三月	2108 季度1 2109
裤子	SJ = 10	SJ = 8	SJ = 15	SJ = 11+
	LA = 18	LA = 22	LA = 10	LA = 16.6+
	CA =	CA =	CA =	CA =
衬衫	SJ = 22	SJ = 27	SJ = 24	SJ = 24.3+
	LA = 36	LA = 28	LA = 31	LA = 31.6+
	CA =	CA =	CA =	CA =
领带	SJ = 8	SJ = 11	SJ = 6	SJ = 8.3+
	LA = 12	LA = 7	LA = 13	LA = 10.6+
	CA =	CA =	CA =	CA =
2102 衣服	SJ = 30*	SJ = 46*	SJ = 45*	SJ = 40.3+
2104	LA = 66*	LA = 57*	LA = 54*	LA = 59+
	CA =	CA =	CA =	CA =

图21C

2100
表C
2105
2106
2108
2109
一月
二月
三月
季度1
裤子
SJ = 10
LA = 18
CA = 18-
SJ = 8
LA = 22
CA = 22-
SJ = 15
LA = 10
CA = 15-
SJ = 11+
LA = 16.6+
CA = 16.6-
2110 — 衬衫
SJ = 22
LA = 36
CA = 36-
SJ = 27
LA = 28
CA = 28-
SJ = 24
LA = 31
CA = 31-
SJ = 24.3+
LA = 31.6+
CA = 31.6-
2112 — 领带
SJ = 8
LA = 12
CA = 12-
SJ = 11
LA = 7
CA = 11-
SJ = 6
LA = 13
CA = 13-
SJ = 8.3+
LA = 10.6+
CA = 10.6-
2102 — 衣服
SJ = 30*
LA = 66*
CA = 66-
SJ = 46*
LA = 57*
CA = 57-
SJ = 45*
LA = 54*
CA = 54-
SJ = 40.3+
LA = 59+
CA = 59-
2104

图 21D

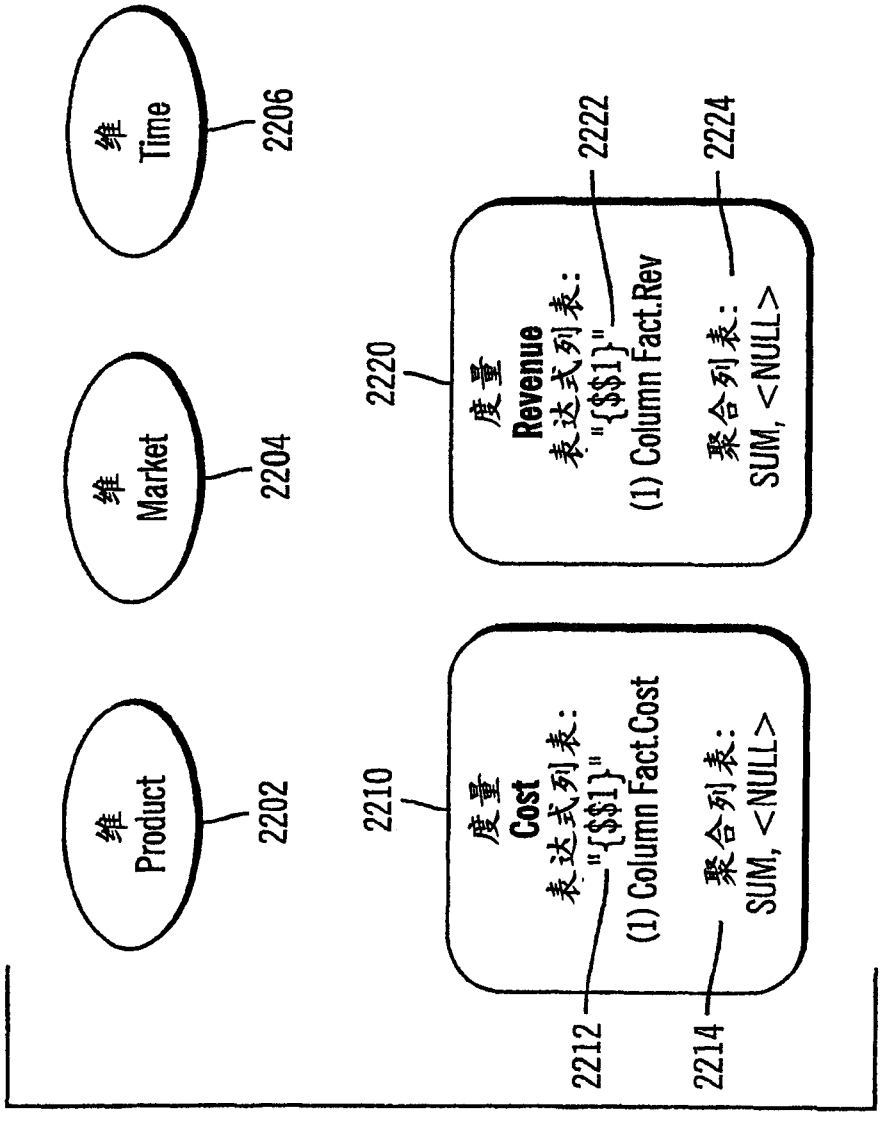

维
Product
2202
维
Market
2204
维
Time
2206
2210
度量
Cost
表达式列表:
"{$$1}"
(1) Column Fact.Cost
聚合列表:
SUM, <NULL>
2220
度量
Revenue
表达式列表:
"{$$1}"
(1) Column Fact.Rev
聚合列表:
SUM, <NULL>
2212
2214
2222
2224

图 22

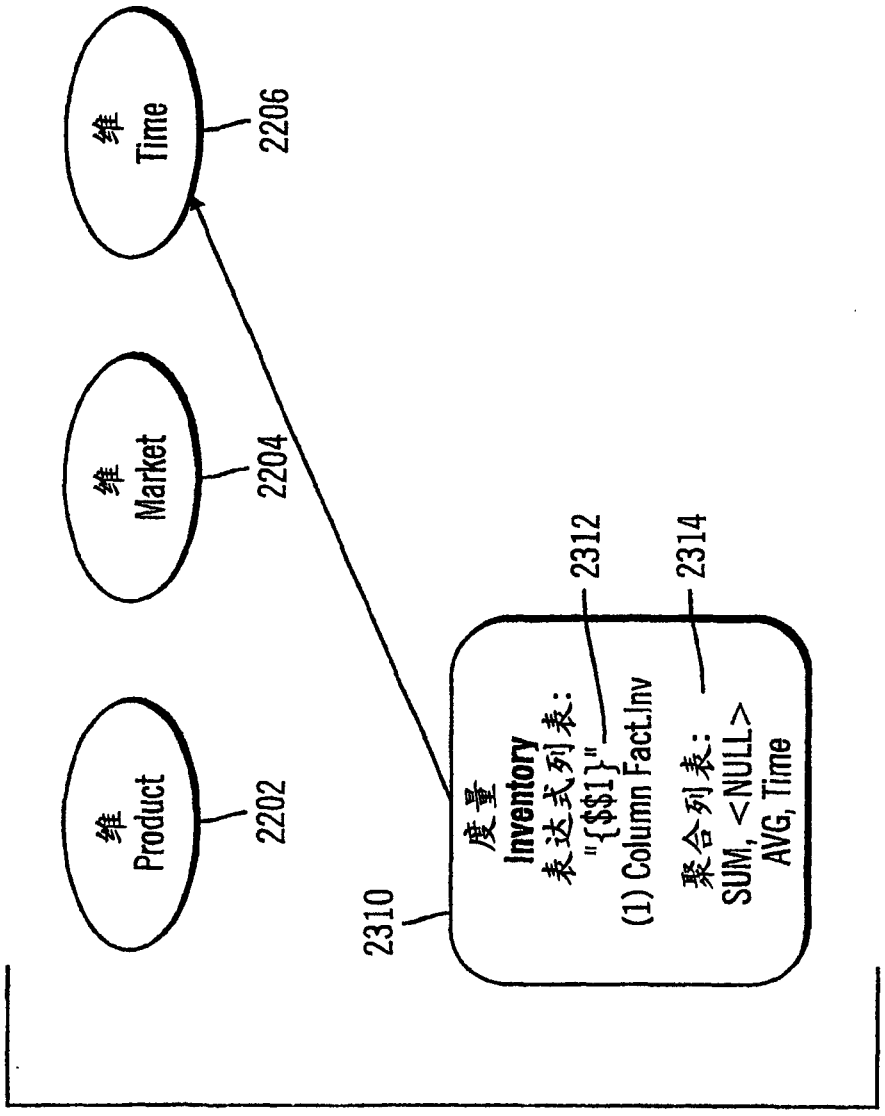

维
Time
2206
维
Market
2204
维
Product
2202
2310
度量
Inventory
表达式列表:
"{$$1}"
(1) Column Fact.Inv
2312
聚合列表:
SUM, <NULL>
AVG, Time
2314

图 23

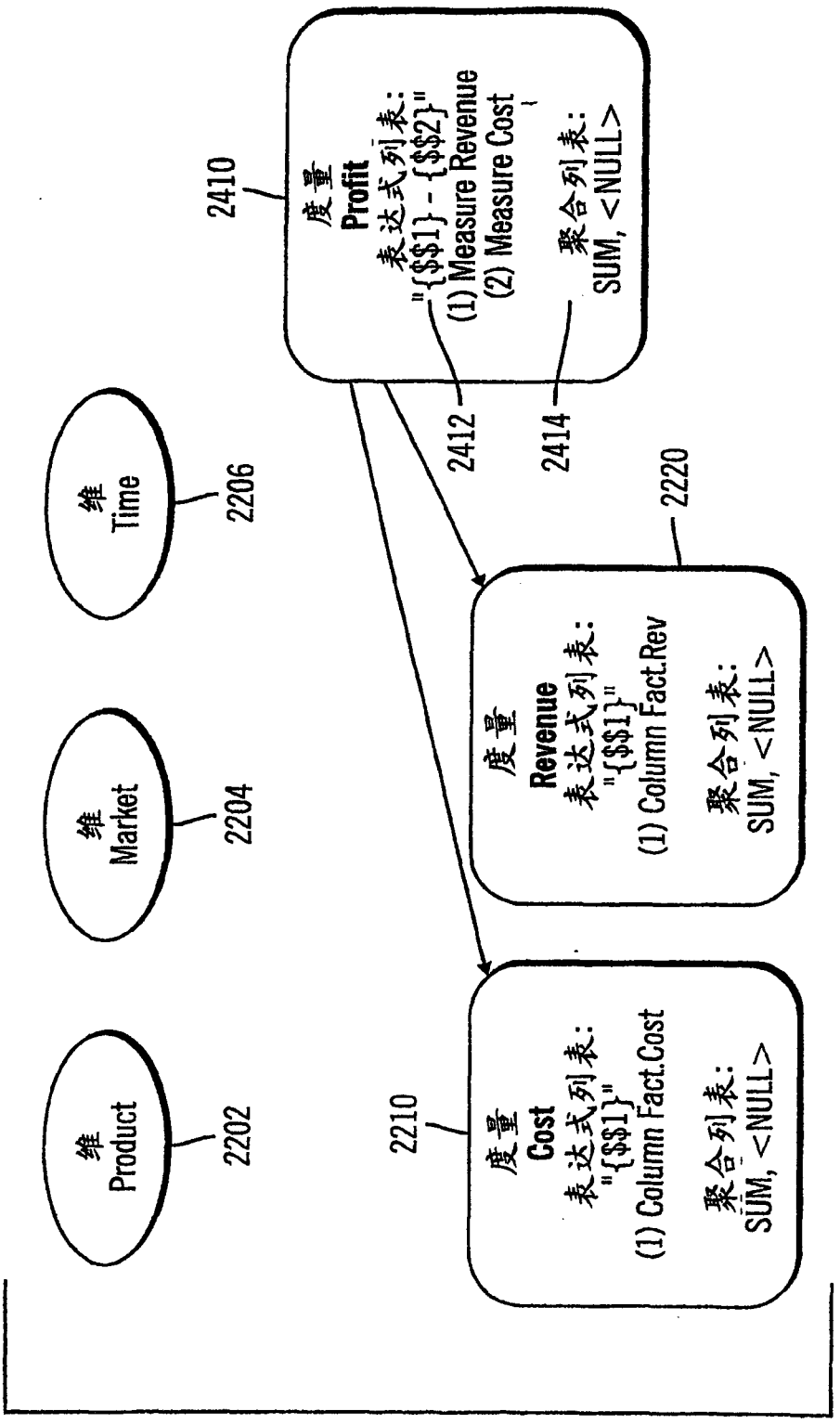

维
Product
2202
维
Market
2204
维
Time
2206
2210
度量
Cost
表达式列表:
"{$$1}"
(1) Column Fact.Cost
聚合列表:
SUM, <NULL>
2220
度量
Revenue
表达式列表:
"{$$1}"
(1) Column Fact.Rev
聚合列表:
SUM, <NULL>
2410
度量
Profit
表达式列表:
"{$$1} - {$$2}"
(1) Measure Revenue
(2) Measure Cost
聚合列表:
SUM, <NULL>
2412
2414

图 24

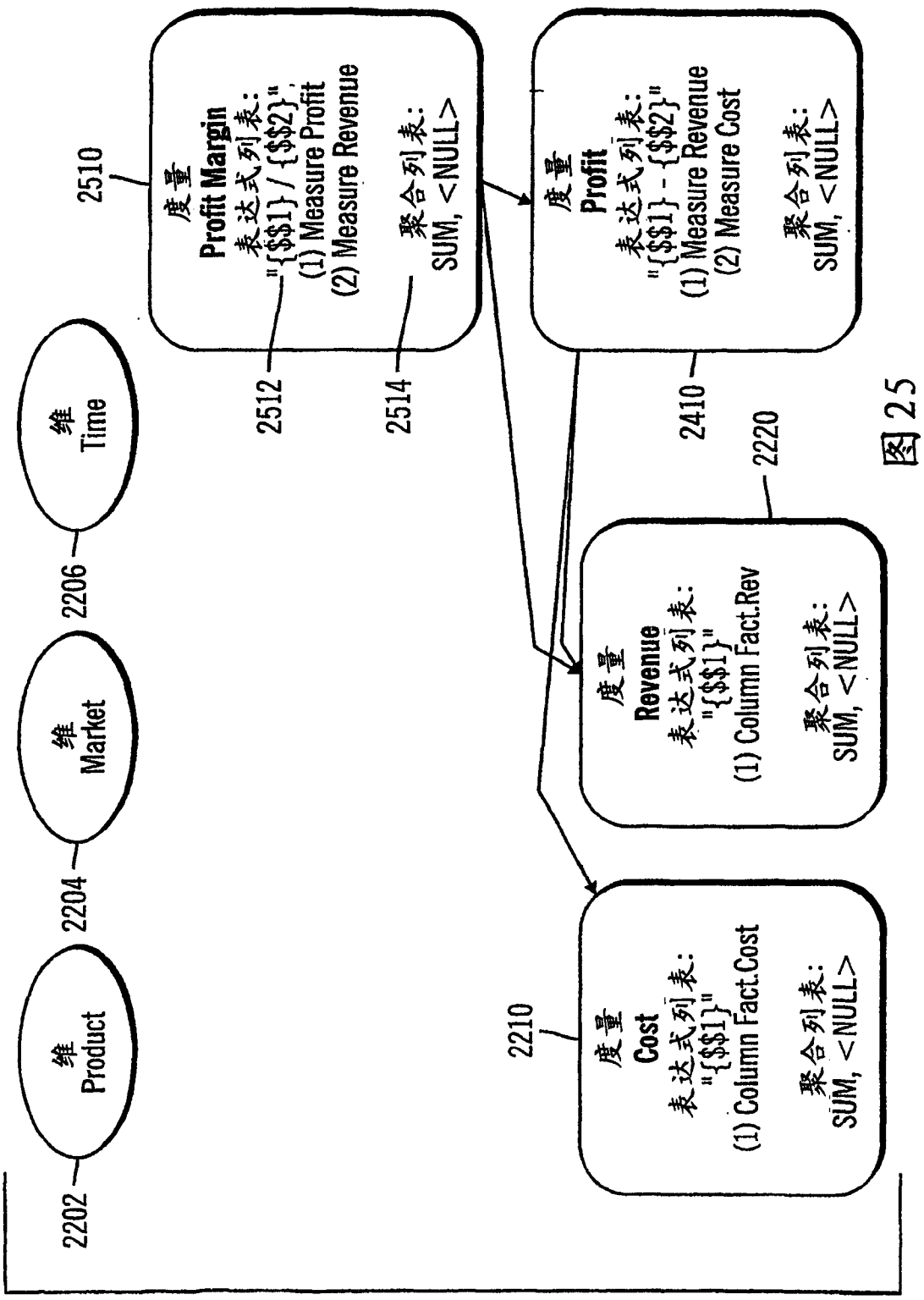

维
Product
2202
维
Market
2204
维
Time
2206
度量
Profit Margin
表达式列表:
"{$$1} / {$$2}"
(1) Measure Profit
(2) Measure Revenue
聚合列表:
SUM, <NULL>
2510
2512
2514
度量
Profit
表达式列表:
"{$$1} - {$$2}"
(1) Measure Revenue
(2) Measure Cost
聚合列表:
SUM, <NULL>
2410
度量
Revenue
表达式列表:
"{$$1}"
(1) Column Fact.Rev
聚合列表:
SUM, <NULL>
2220
度量
Cost
表达式列表:
"{$$1}"
(1) Column Fact.Cost
聚合列表:
SUM, <NULL>
2210

图 25

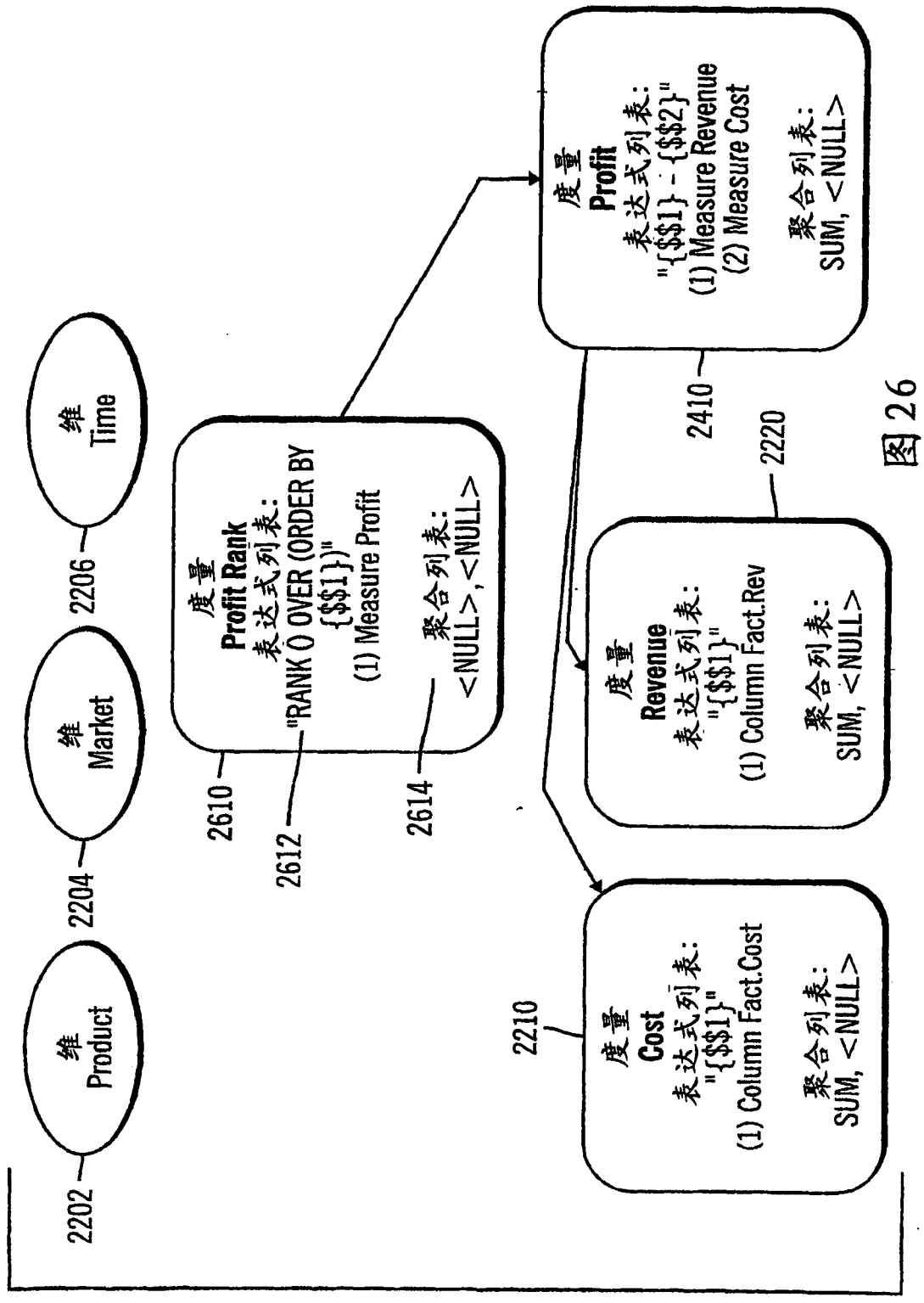

维
Time
2206
维
Market
2204
维
Product
2202
度量
Profit Rank
表达式列表:
"RANK() OVER (ORDER BY
{$$1})"
(1) Measure Profit
聚合列表:
<NULL>,<NULL>
2610
2612
2614
度量
Profit
表达式列表:
"{$$1} - {$$2}"
(1) Measure Revenue
(2) Measure Cost
聚合列表:
SUM, <NULL>
2410
度量
Revenue
表达式列表:
"{$$1}"
(1) Column Fact.Rev
聚合列表:
SUM, <NULL>
2220
度量
Cost
表达式列表:
"{$$1}"
(1) Column Fact.Cost
聚合列表:
SUM, <NULL>
2210

图 26

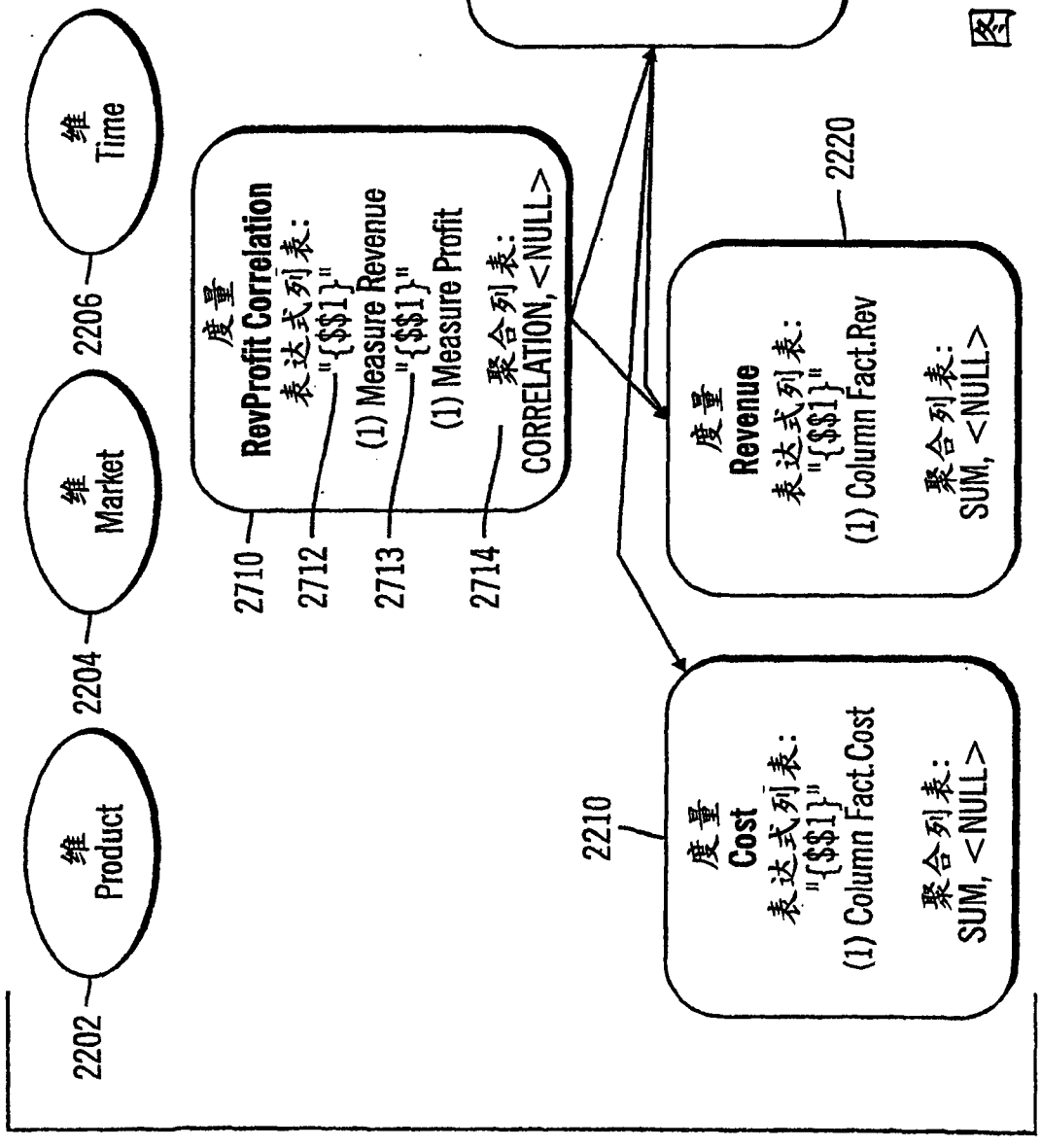

维
Product
2202
维
Market
2204
维
Time
2206
度量
RevProfit Correlation
表达式列表:
"{$$1}"
(1) Measure Revenue
"{$$1}"
(1) Measure Profit
聚合列表:
CORRELATION, <NULL>
2710
2712
2713
2714
度量
Profit
表达式列表:
"{$$1} - {$$2}"
(1) Measure Revenue
(2) Measure Cost
聚合列表:
SUM, <NULL>
2410
度量
Revenue
表达式列表:
"{$$1}"
(1) Column Fact.Rev
聚合列表:
SUM, <NULL>
2220
度量
Cost
表达式列表:
"{$$1}"
(1) Column Fact.Cost
聚合列表:
SUM, <NULL>
2210

图 27

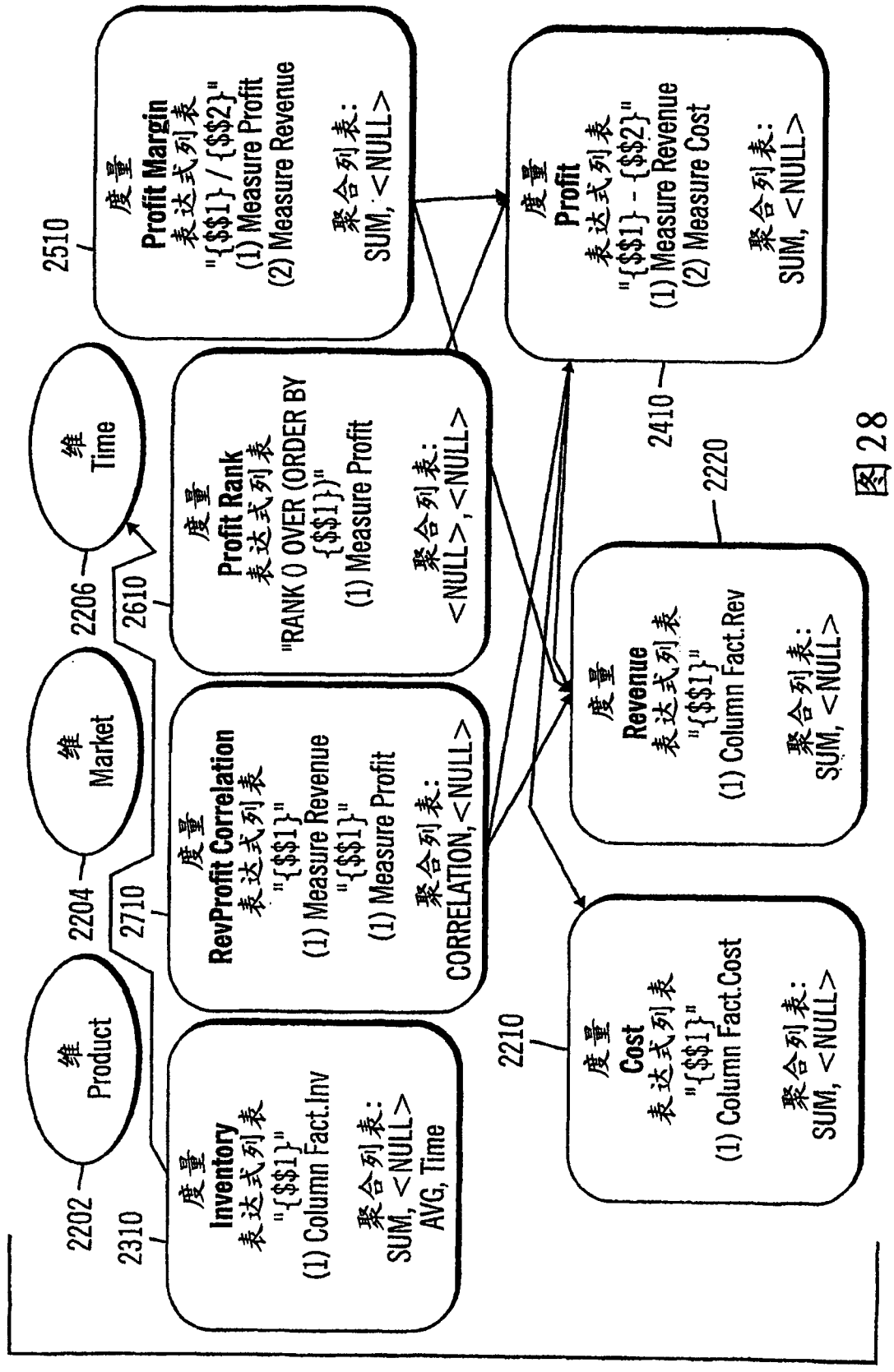

维 Product
2202
维 Market
2204
维 Time
2206
2310
度量
Inventory
表达式列表
"{$$1}"
(1) Column Fact.Inv
聚合列表:
SUM, <NULL>
AVG, Time
2710
度量
RevProfit Correlation
表达式列表
"{$$1}"
(1) Measure Revenue
"{$$1}"
(1) Measure Profit
聚合列表:
CORRELATION, <NULL>
2610
度量
Profit Rank
表达式列表
"RANK () OVER (ORDER BY {$$1})"
(1) Measure Profit
聚合列表:
<NULL>, <NULL>
2510
度量
Profit Margin
表达式列表
"{$$1} / {$$2}"
(1) Measure Profit
(2) Measure Revenue
聚合列表:
SUM, <NULL>
2210
度量
Cost
表达式列表
"{$$1}"
(1) Column Fact.Cost
聚合列表:
SUM, <NULL>
2220
度量
Revenue
表达式列表
"{$$1}"
(1) Column Fact.Rev
聚合列表:
SUM, <NULL>
2410
度量
Profit
表达式列表
"{$$1} - {$$2}"
(1) Measure Revenue
(2) Measure Cost
聚合列表:
SUM, <NULL>

图 28

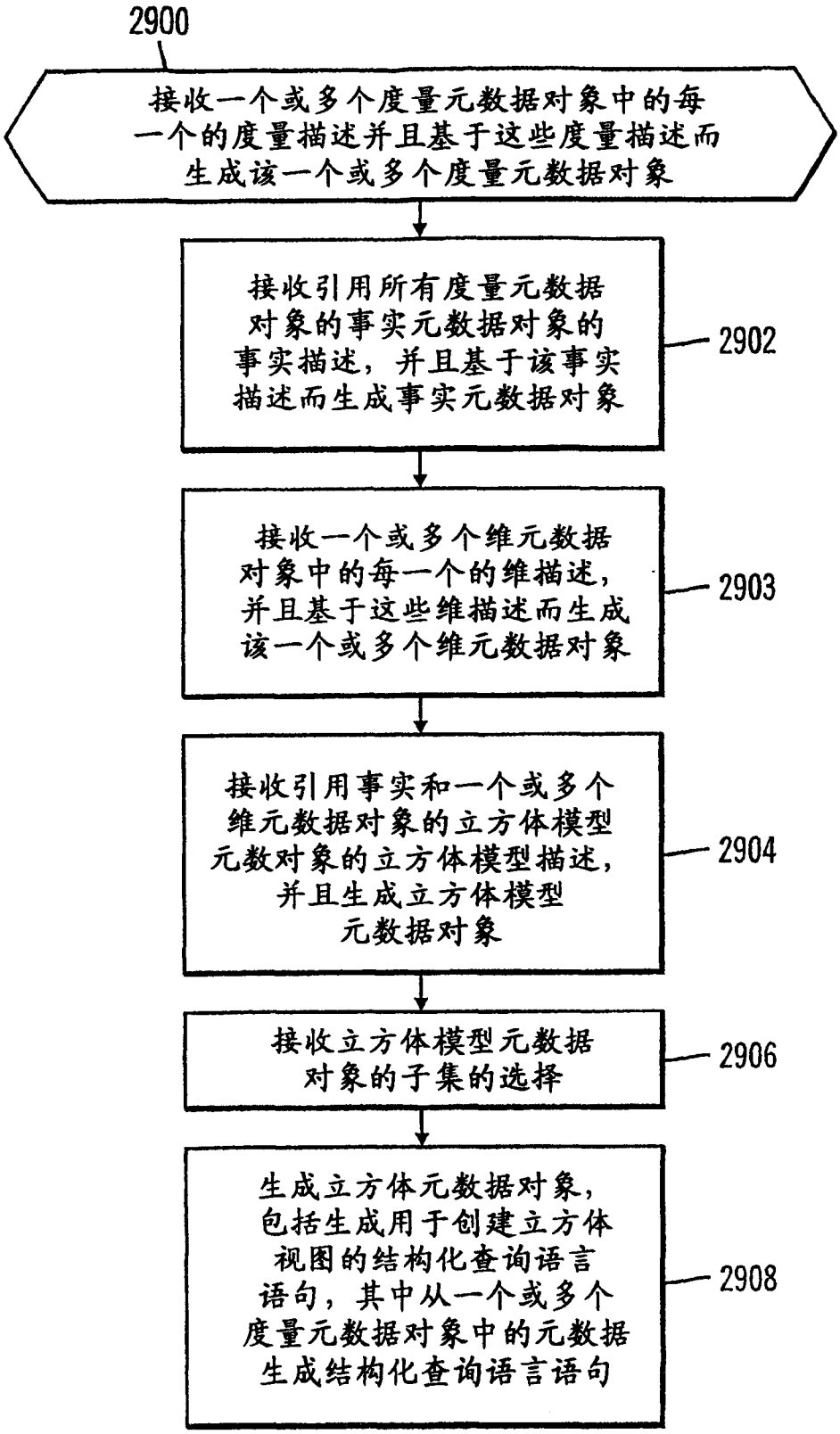

2900
接收一个或多个度量元数据对象中的每一个的度量描述并且基于这些度量描述而生成该一个或多个度量元数据对象
接收引用所有度量元数据对象的事实元数据对象的事实描述，并且基于该事实描述而生成事实元数据对象
2902
接收一个或多个维元数据对象中的每一个的维描述，并且基于这些维描述而生成该一个或多个维元数据对象
2903
接收引用事实和一个或多个维元数据对象的立方体模型元数对象的立方体模型描述，并且生成立方体模型元数据对象
2904
接收立方体模型元数据对象的子集的选择
2906
生成立方体元数据对象，包括生成用于创建立方体视图的结构化查询语言语句，其中从一个或多个度量元数据对象中的元数据生成结构化查询语言语句
2908

图29A

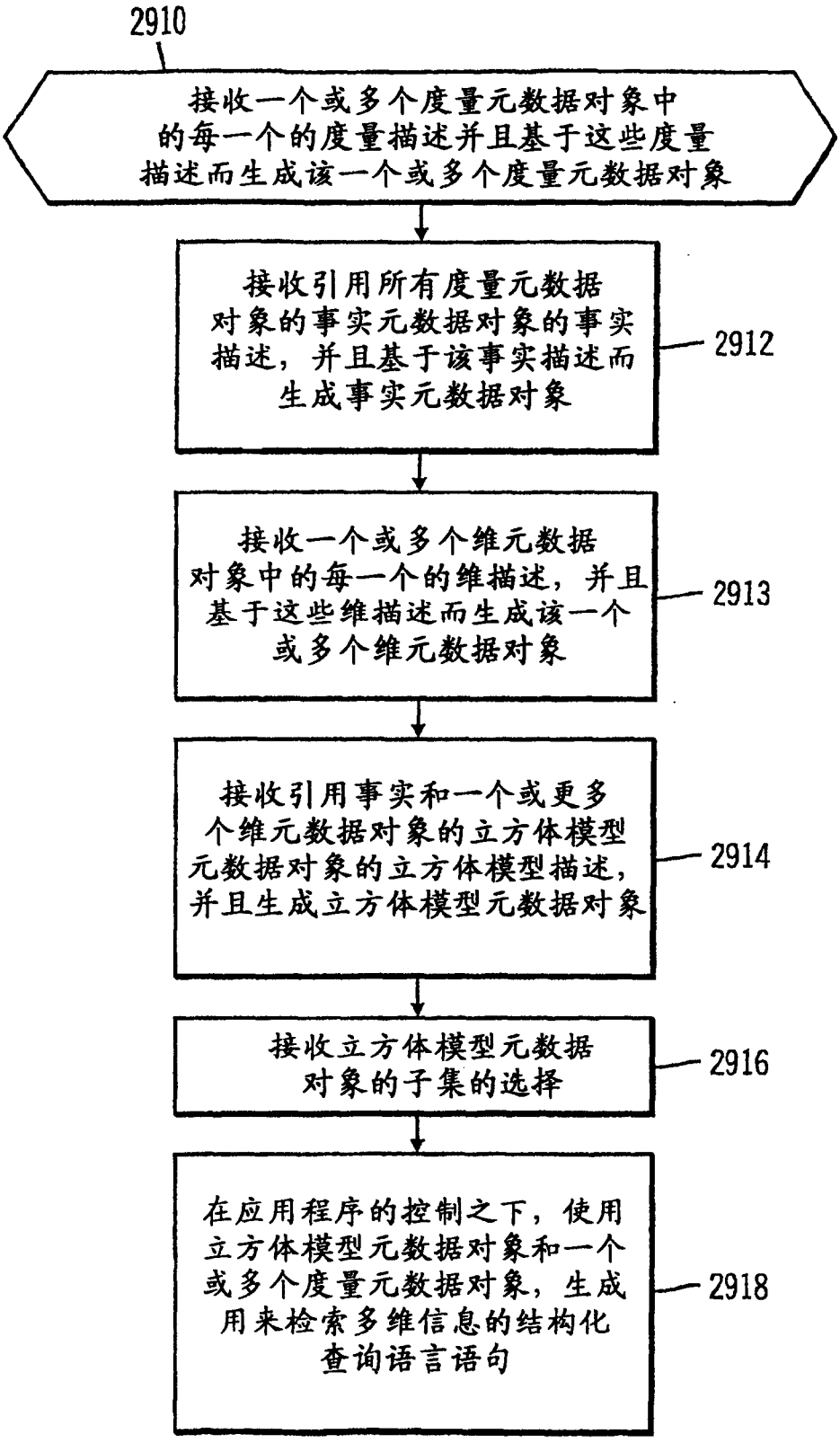

2910
接收一个或多个度量元数据对象中的每一个的度量描述并且基于这些度量描述而生成该一个或多个度量元数据对象
接收引用所有度量元数据对象的事实元数据对象的事实描述，并且基于该事实描述而生成事实元数据对象
2912
接收一个或多个维元数据对象中的每一个的维描述，并且基于这些维描述而生成该一个或多个维元数据对象
2913
接收引用事实和一个或更多个维元数据对象的立方体模型元数据对象的立方体模型描述，并且生成立方体模型元数据对象
2914
接收立方体模型元数据对象的子集的选择
2916
在应用程序的控制之下，使用立方体模型元数据对象和一个或多个度量元数据对象，生成用来检索多维信息的结构化查询语言语句
2918

图29B

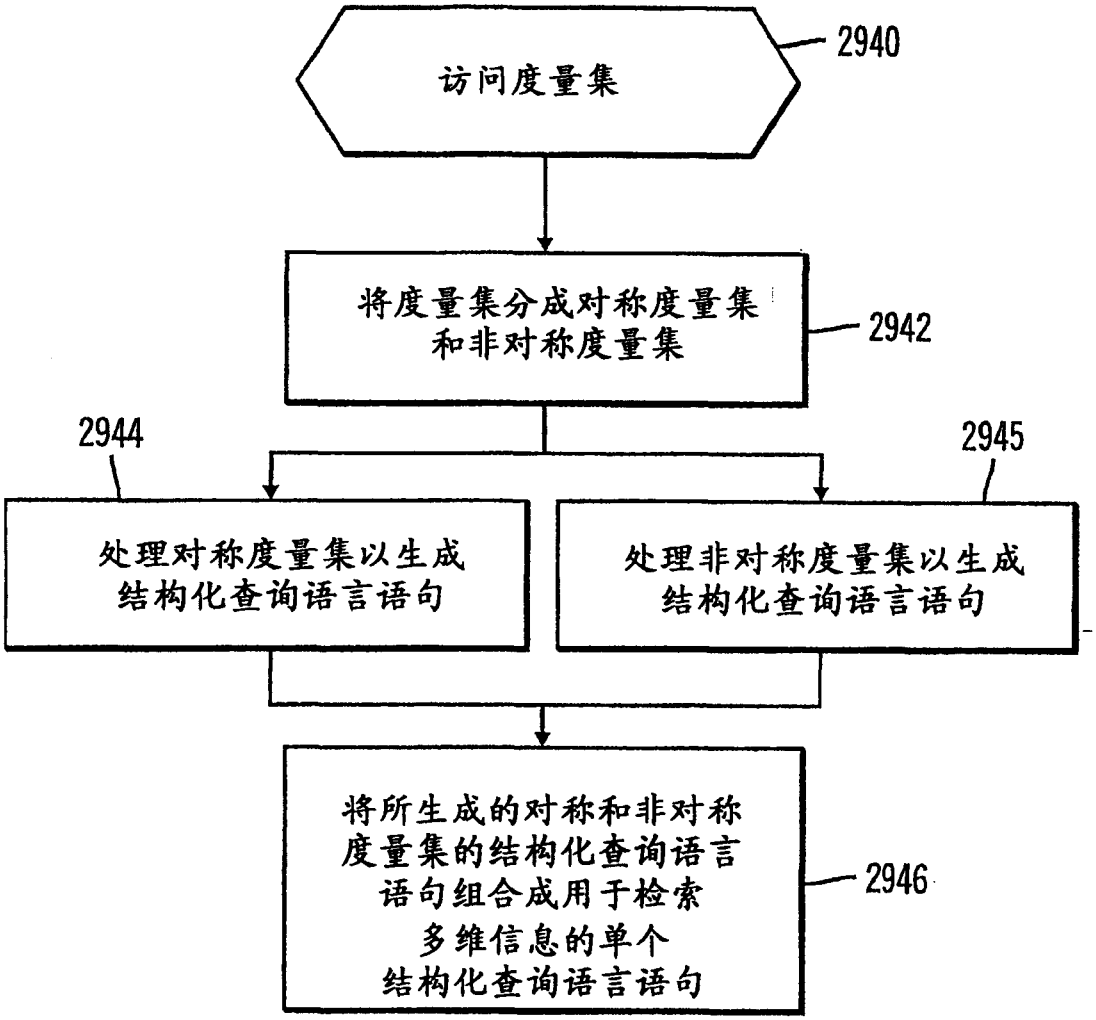

访问度量集
2940
将度量集分成对称度量集
和非对称度量集
2942
2944
2945
处理对称度量集以生成
结构化查询语言语句
处理非对称度量集以生成
结构化查询语言语句
将所生成的对称和非对称
度量集的结构化查询语言
语句组合成用于检索
多维信息的单个
结构化查询语言语句
2946

图29C

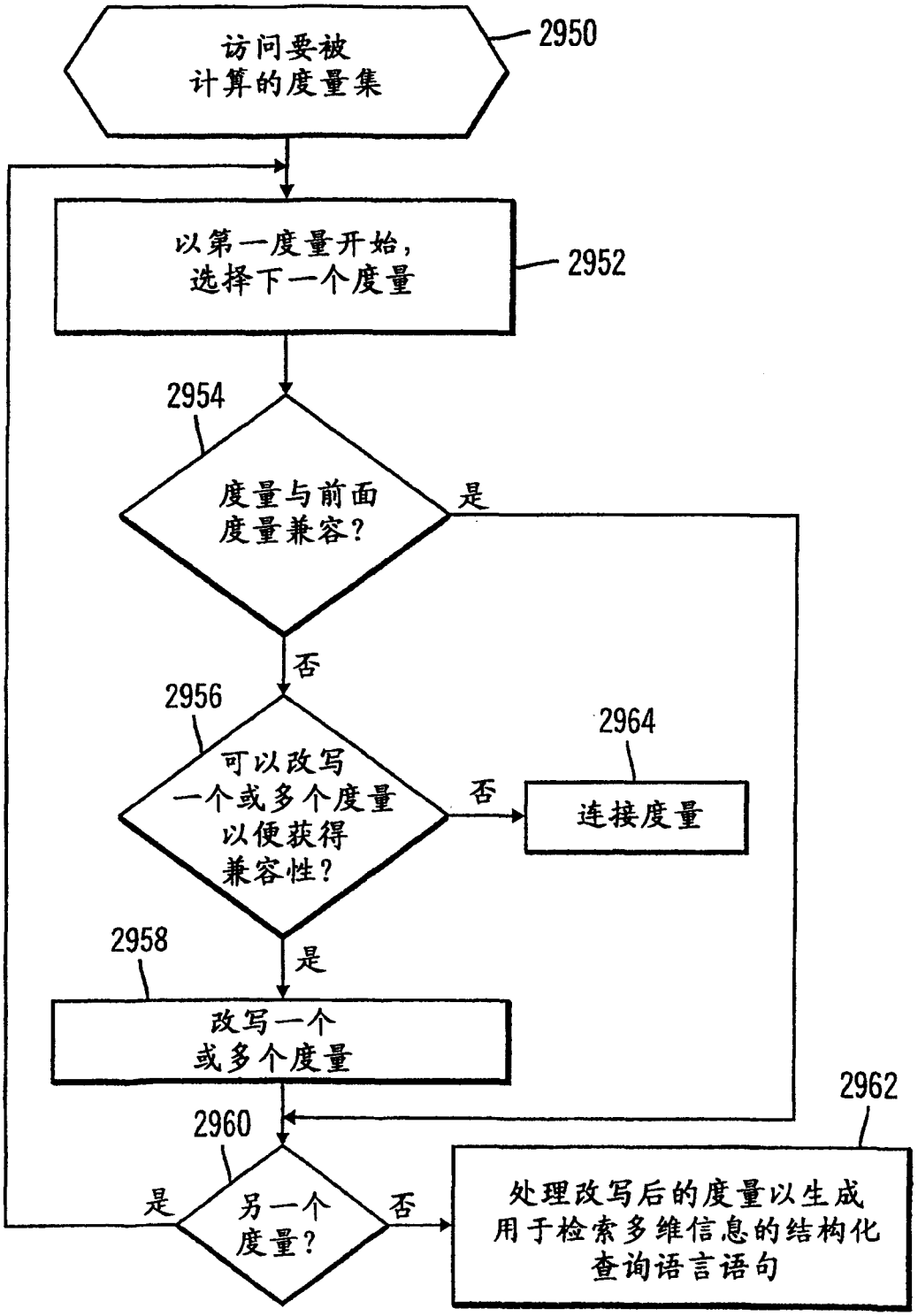

访问要被
计算的度量集
2950
以第一度量开始，
选择下一个度量
2952
2954
度量与前面
度量兼容？
是
否
2956
可以改写
一个或多个度量
以便获得
兼容性？
否
2964
连接度量
是
2958
改写一个
或多个度量
2962
2960
是
另一个
度量？
否
处理改写后的度量以生成
用于检索多维信息的结构化
查询语言语句

图29D

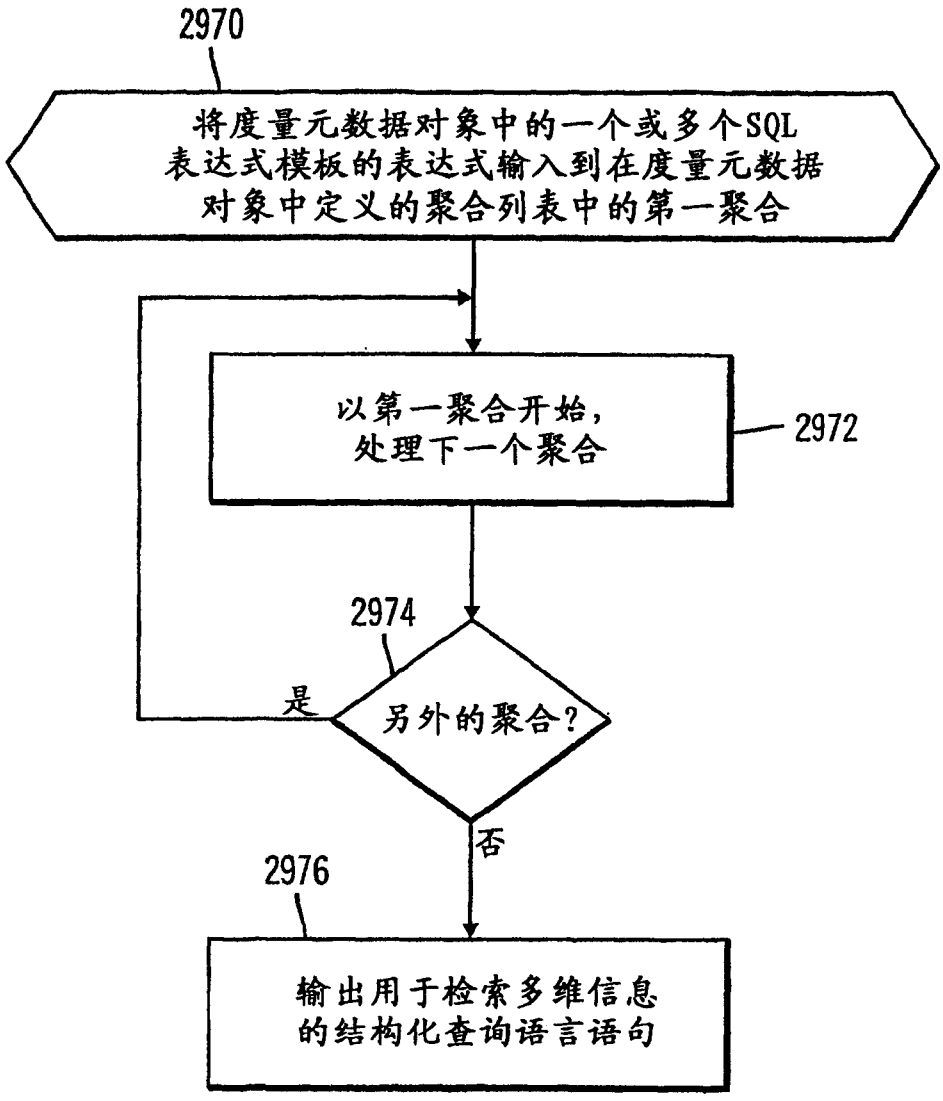

2970
将度量元数据对象中的一个或多个SQL表达式模板的表达式输入到在度量元数据对象中定义的聚合列表中的第一聚合
以第一聚合开始，处理下一个聚合
2972
2974
是
另外的聚合?
否
2976
输出用于检索多维信息的结构化查询语言语句

图29E

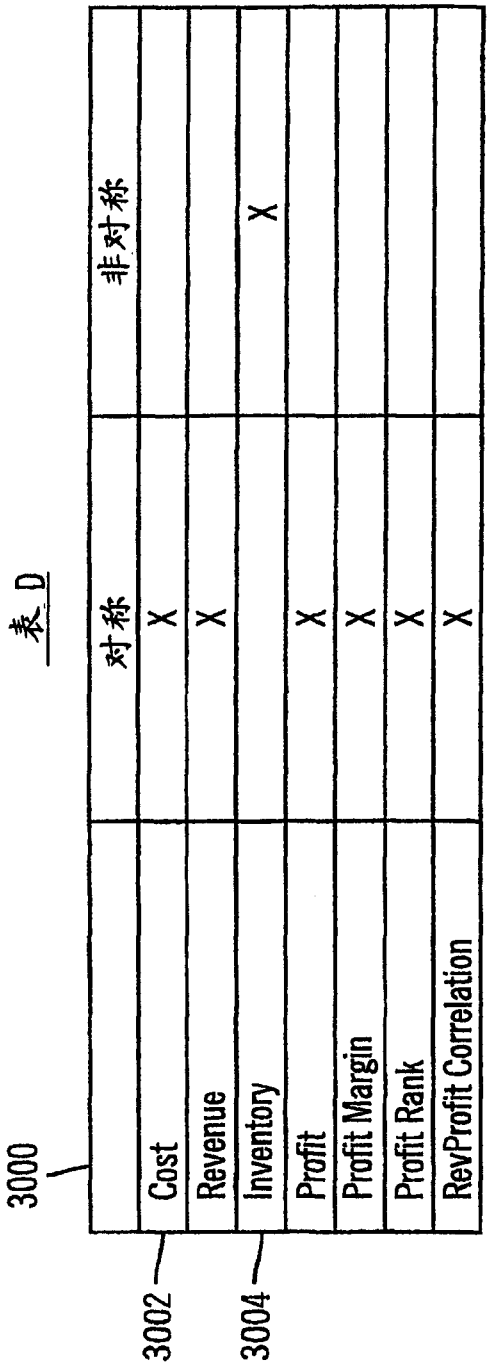


表 D
3000
3002
3004
对称
非对称
Cost
Revenue
Inventory
Profit
Profit Margin
Profit Rank
RevProfit Correlation
X

图 30

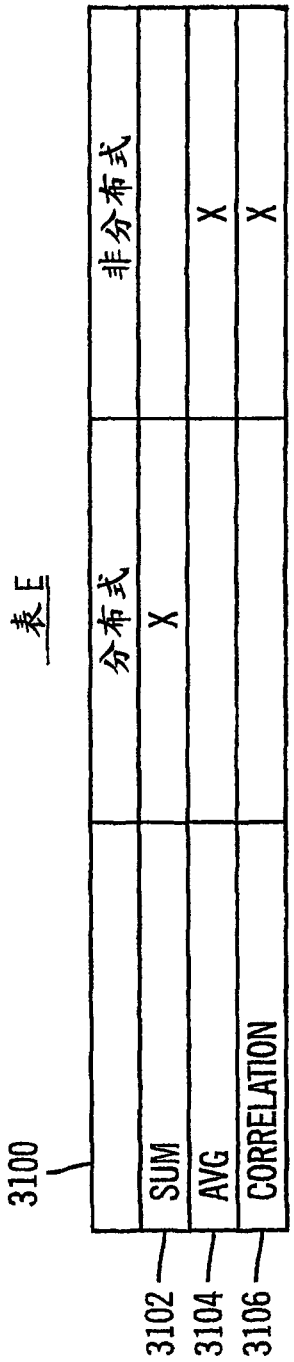


图 31

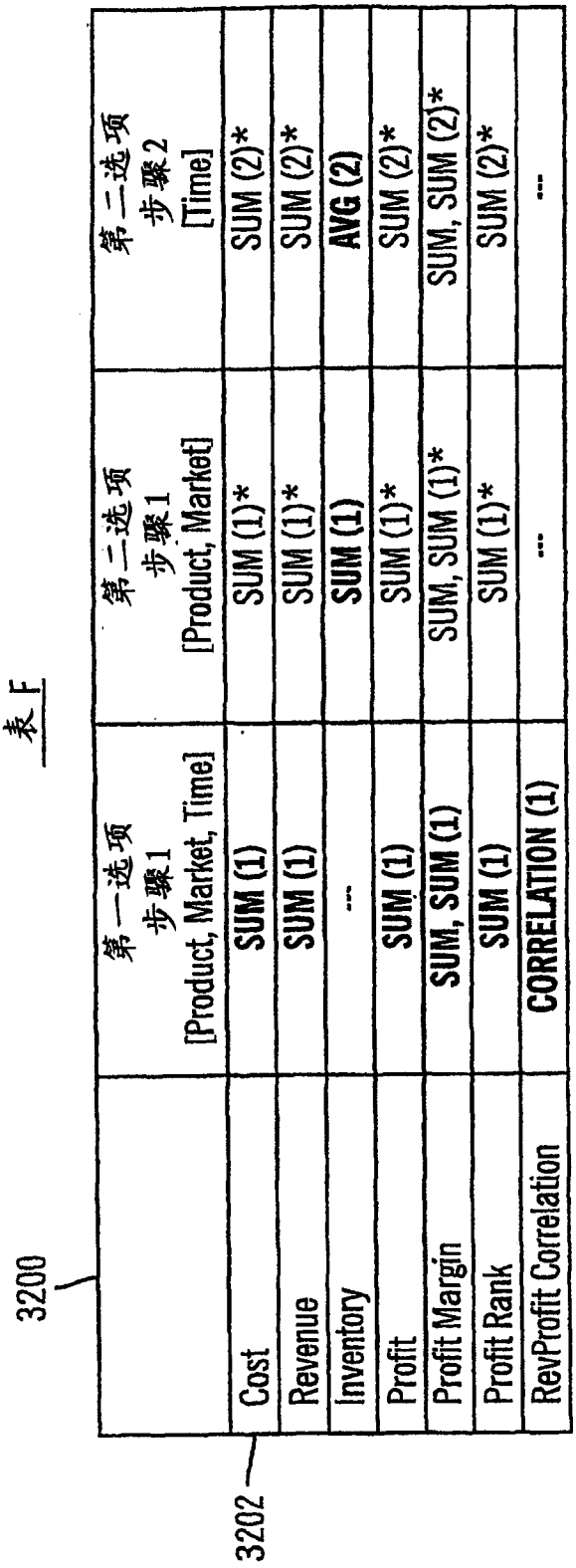

3200
表 F
3202
第一选项
步骤 1
[Product, Market, Time]
第二选项
步骤 1
[Product, Market]
第二选项
步骤 2
[Time]
Cost
SUM (1)
SUM (1)*
SUM (2)*
Revenue
SUM (1)
SUM (1)*
SUM (2)*
Inventory
...
SUM (1)
AVG (2)
Profit
SUM (1)
SUM (1)*
SUM (2)*
Profit Margin
SUM, SUM (1)
SUM, SUM (1)*
SUM, SUM (2)*
Profit Rank
SUM (1)
SUM (1)*
SUM (2)*
RevProfit Correlation
CORRELATION (1)
...
...

图 32

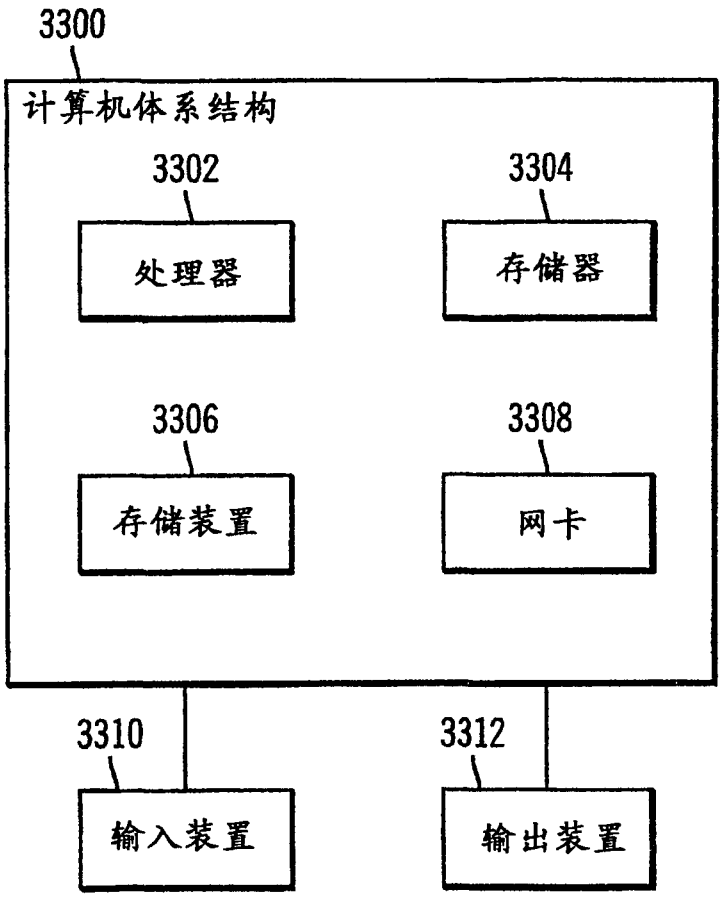

3300
计算机体系结构
3302
处理器
3304
存储器
3306
存储装置
3308
网卡
3310
输入装置
3312
输出装置

图33