(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 112912872 B
(45)授权公告日 2024.08.06

(21)申请号 201980068323.5
(22)申请日 2019.10.15
(65)同一申请的已公布的文献号
申请公布号 CN 112912872 A
(43)申请公布日 2021.06.04
(30)优先权数据
201841039503 2018.10.18 IN
16/439,532 2019.06.12 US
(85)PCT国际申请进入国家阶段日
2021.04.16
(86)PCT国际申请的申请数据
PCT/US2019/056363 2019.10.15
(87)PCT国际申请的公布数据
WO2020/081586 EN 2020.04.23

(73)专利权人 甲骨文国际公司
地址 美国加利福尼亚
(72)发明人 P·卡特卡德 N·瑞奇曼
S·瑞阿波
(74)专利代理机构 中国贸促会专利商标事务所有限公司 11038
专利代理师 吴信刚

(51)Int.Cl.
G06F 16/28(2006.01)
(56)对比文件
US 2017116309 A1,2017.04.27
审查员 倪赛华

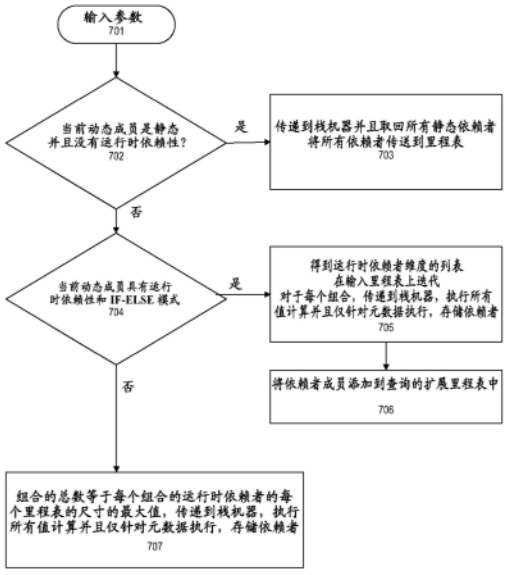
权利要求书3页 说明书15页 附图8页

(54)发明名称

用于多维数据库环境中的依赖性分析的系统和方法

(57)摘要

根据一实施例,对每个动态(或临时)成员进行依赖性分析,以收集BSO(块存储选项)和ASO(聚合存储选项)数据立方体中的运行时依赖者以及静态依赖者的列表。这使成员能够加入混合流,以便以自下而上的方式执行。



CN 112912872 B

1.一种用于多维数据库中的依赖性分析的系统,包括:

包括一个或多个微处理器的计算机;

在所述计算机上执行的多维数据库服务器,其中所述多维数据库服务器支持至少一个多维数据立方体,其中所述多维数据库包括多个维度,所述多个维度中的每一个包括多个成员;以及

动态成员,其中所述动态成员依赖于所述多个成员的集合;

其中,所述系统进行依赖性分析以确定所述动态成员所依赖的所述多个成员的集合;

其中,所述依赖性分析使包括一个或多个微处理器的所述计算机执行包括以下的步骤:

对所述动态成员进行令牌化,得到所述动态成员的一个或多个令牌;

分析所述动态成员的所述一个或多个令牌中的每一个;以及

基于对所述动态成员的所述一个或多个令牌中的每一个的所述分析,确定所述动态成员所依赖的所述多个成员的集合仅包括静态依赖者,或者确定所述动态成员所依赖的所述多个成员的集合包括一个或多个运行时依赖者。

2.根据权利要求1所述的系统,

其中,在支持所述至少一个多维数据库立方体的所述多维数据库服务器的启动操作时进行所述依赖性分析。

3.根据权利要求1或2所述的系统,

其中,基于确定所述动态成员所依赖的所述多个成员的集合仅包括静态依赖者,所述动态成员所依赖的所有静态依赖者被取回并被传送到输入里程表。

4.根据权利要求1或2所述的系统,

其中,基于确定所述动态成员所依赖的所述多个成员的集合包括一个或多个运行时依赖者,产生所述一个或多个运行时依赖者的数组;并且

其中,基于所产生的所述一个或多个运行时依赖者的数组,产生所述多维数据库立方体的运行时依赖者成员列表。

5.根据权利要求4所述的系统,

其中,在所述多维数据库立方体上运行对所述动态成员的查询。

6.根据权利要求5所述的系统,

其中,对所述动态成员的所述查询利用所述多维数据库立方体的所述运行时依赖者成员列表来取回与该依赖者成员列表上的每一个依赖者成员相关联的值。

7.一种用于多维数据库中的依赖性分析的方法,包括:

在包括一个或多个微处理器的计算机处提供:

在所述计算机上执行的多维数据库服务器,其中所述多维数据库服务器支持至少一个多维数据立方体,其中所述多维数据库包括多个维度,所述多个维度中的每一个包括多个成员;以及

动态成员,其中所述动态成员依赖于所述多个成员的集合;以及

进行依赖性分析以确定所述动态成员所依赖的所述多个成员的集合;

其中,所述进行依赖性分析包括:

对所述动态成员进行令牌化,得到所述动态成员的一个或多个令牌;

分析所述动态成员的所述一个或多个令牌中的每一个;以及

基于对所述动态成员的所述一个或多个令牌中的每一个的所述分析,确定所述动态成员仅依赖于静态依赖者,或者确定所述动态成员依赖于一个或多个运行时依赖者。

8.根据权利要求7所述的方法,还包括:

在支持所述至少一个多维数据库立方体的所述多维数据库服务器的启动操作时进行所述依赖性分析。

9.根据权利要求7或8所述的方法,其中,所述进行依赖性分析还包括:

基于所述动态成员仅依赖于静态依赖者的所述确定,取回所述动态成员所依赖的所有静态依赖者并传送到输入里程表。

10.根据权利要求7或8所述的方法,其中,所述进行依赖性分析还包括:

基于所述动态成员依赖于一个或多个运行时依赖者的所述确定,产生所述一个或多个运行时依赖者的数组;以及

基于所产生的所述一个或多个运行时依赖者的数组,产生所述多维数据库立方体的运行时依赖者成员列表。

11.根据权利要求10所述的方法,还包括:

在所述多维数据库立方体上运行对所述动态成员的查询。

12.根据权利要求11所述的方法,

其中,对所述动态成员的所述查询利用所述多维数据库立方体的所述运行时依赖者成员列表来取回与该依赖者成员列表上的每一个依赖者成员相关联的值。

13.一种非暂态计算机可读存储介质,其上具有用于多维数据库中的依赖性分析的指令,所述指令在被读取并执行时使计算机执行包括以下的步骤:

在包括一个或多个微处理器的所述计算机处提供:

在所述计算机上执行的多维数据库服务器,其中所述多维数据库服务器支持至少一个多维数据立方体,其中所述多维数据库包括多个维度,所述多个维度中的每一个包括多个成员;以及

动态成员,其中所述动态成员依赖于所述多个成员的集合;以及

进行依赖性分析以确定所述动态成员所依赖的所述多个成员的集合;

其中,所述进行依赖性分析包括:

对所述动态成员进行令牌化,得到所述动态成员的一个或多个令牌;

分析所述动态成员的所述一个或多个令牌中的每一个;以及

基于对所述动态成员的所述一个或多个令牌中的每一个的所述分析,确定所述动态成员仅依赖于静态依赖者,或者确定所述动态成员依赖于一个或多个运行时依赖者。

14.根据权利要求13所述的非暂态计算机可读存储介质,还包括:

在支持所述至少一个多维数据库立方体的所述多维数据库服务器的启动操作时进行所述依赖性分析。

15.根据权利要求13或14所述的非暂态计算机可读存储介质,

其中,所述进行依赖性分析还包括:

基于所述动态成员仅依赖于静态依赖者的所述确定,取回所述动态成员所依赖的所有静态依赖者并传送到输入里程表。

16.根据权利要求13或14所述的非暂态计算机可读存储介质，

其中，所述进行依赖性分析还包括：

基于所述动态成员依赖于一个或多个运行时依赖者的所述确定，产生所述一个或多个运行时依赖者的数组；以及

基于所产生的所述一个或多个运行时依赖者的数组，产生所述多维数据库立方体的运行时依赖者成员列表。

17.根据权利要求16所述的非暂态计算机可读存储介质，还包括

在所述多维数据库立方体上运行对所述动态成员的查询；

其中，对所述动态成员的所述查询利用所述多维数据库立方体的所述运行时依赖者成员列表来取回与该依赖者成员列表上的每一个依赖者成员相关联的值。

用于多维数据库环境中的依赖性分析的系统和方法

[0001] 版权声明

[0002] 本专利文件的一部分公开内容包含受版权保护的材料。版权所有者不反对任何人对如专利商标局专利文件或记录中所出现的专利文件或专利公开进行传真复制,但在其他方面均保留所有版权。

[0003] 优先权要求

[0004] 本申请要求于2019年6月12日提交的标题为“STSTEM AND METHOD FOR DEPENDENCY ANALYSIS IN A MULTIDIMENSIONAL DATABASE ENVIRONMENT”的申请号为16/439,532的美国专利申请、以及于2018年10月18日提交的标题为“STSTEM AND METHOD FOR DEPENDENCY ANALYSIS IN A MULTIDIMENSIONAL DATABASE ENVIRONMENT”的申请号为201841039503的印度专利申请的优先权的权益,这些申请通过引用并入本文。

技术领域

[0005] 本发明的实施例总体上涉及数据库和数据仓库,并且具体地涉及用于多维数据库环境中的依赖性分析的系统和方法。

背景技术

[0006] 多维数据库计算环境使公司能够在其需要时将关键业务信息交付给合适的人,包括利用和集成来自多个现有数据源的数据,以及以最符合终端用户的需要的格式将过滤后的信息分发给终端用户社区的能力。用户可以沿着熟悉的业务维度实时交互和浏览数据,从而实现思维速度(speed-of-thought)分析。这些是可以使用本发明的实施例的环境类型的一些示例。

发明内容

[0007] 根据一实施例,本文描述了用于多维数据库中的依赖性分析的系统和方法,根据一实施例。动态成员可以依赖于来自同一大纲(outline)的其它成员,其被称为公式的依赖者,自然地,为了评估用于每一个交集的原始公式,必须首先计算这些依赖者。因此,为了评估混合数据聚合模型中的动态成员的公式,系统可以首先准备该公式所依赖的所有成员的列表。此外,动态成员的依赖者列表可以被分类为“运行时依赖者”和“静态依赖者”。动态成员的运行时依赖者是对于每一个交集都不同的依赖者,而静态依赖者是与交集无关的恒定的依赖者。这种识别动态成员的运行时依赖者以及静态依赖者的分析处理被称为混合数据聚合模型中的依赖性分析。

[0008] 根据一实施例,一种用于多维数据库中的依赖性分析的示例性方法可以在包括一个或多个微处理器的计算机处提供:在该计算机上执行的多维数据库服务器,其中该多维数据库服务器支持至少一个多维数据立方体,其中多维数据库包括多个维度,所述多个维度中的每一个维度包括多个成员;以及动态成员,其中该动态成员依赖于所述多个成员的集合。该方法可以进行依赖性分析以确定动态成员所依赖的所述多个成员的集合。

[0009] 根据一实施例,所述进行依赖性分析可以包括对动态成员进行令牌化,从而得到该动态成员的一个或多个令牌。然后,依赖性分析可以分析动态成员的所述一个或多个令牌中的每一个令牌,并且基于对动态成员的所述一个或多个令牌中的每一个令牌的分析,确定动态成员仅依赖于静态依赖者。在这种情况下,该方法可以基于动态成员仅依赖于静态依赖者的确定,来取回动态成员所依赖的所有静态依赖者,并将其转移至输入里程表。

[0010] 根据一实施例,所述进行依赖性分析可以包括对动态成员进行令牌化,从而得到该动态成员的一个或多个令牌。然后,依赖分析可以分析动态成员的所述一个或多个令牌中的每一个令牌,并且基于对动态成员的所述一个或多个令牌中的每一个令牌的分析,确定动态成员依赖于一个或多个运行时依赖者。然后,该方法可以基于动态成员依赖于一个或多个运行时依赖者的确定,来产生所述一个或多个运行时依赖者的数组。该方法还可以基于所产生的所述一个或多个运行时依赖者的数组,来产生多维数据库立方体的运行时依赖者成员列表。

附图说明

[0011] 图1示出了根据一实施例的多维数据库环境的示例。

[0012] 图2示出了根据一实施例的针对多维数据库使用动态流。

[0013] 图3进一步示出了根据一实施例的针对多维数据库使用动态流。

[0014] 图4示出了根据一实施例的示例性函数栈。

[0015] 图5示出了根据一实施例的示例性数据集。

[0016] 图6示出了根据一实施例的示例性数据集。

[0017] 图7是根据一实施例的用于依赖性分析的示例性方法的流程图。

[0018] 图8是根据一实施例的用于依赖性分析的示例性方法的流程图。

具体实施方式

[0019] 参考所包含的说明书、权利要求书和附图,前述以及其它特征将变得显而易见。为了提供对各种实施例的理解,对具体细节进行了阐述。然而,显而易见的是,可以在没有这些具体细节的情况下实践各种实施例。所包含的说明书和附图不是限制性的。

[0020] 多维数据库环境(其一个示例包括Oracle Essbase)可用于集成大量数据(在某些情况下来自多个数据源的大量数据),并以解决终端用户的特定需求的方式将过滤后的信息分发给这些用户。

[0021] 图1示出了根据一实施例的多维数据库环境100的示例。

[0022] 如图1所示,根据一实施例,作为数据库层操作的多维数据库环境可以包括一个或多个多维数据库服务器系统102,每一个多维数据库服务器系统可以包括物理计算机资源或部件104(例如,微处理器/CPU、物理存储器、网络组件)、操作系统106以及一个或多个多维数据库服务器110(例如,Essbase服务器)。

[0023] 根据一实施例,中间层120可以包括一个或多个服务,例如提供者服务122(例如,Hyperion提供者服务)、管理服务124(例如,Essbase管理服务)或工作室/集成服务126(例如,Essbase工作室/Essbase集成服务)。中间层可以经由ODBC/JDBC 127、128或其它类型的接口提供对元数据目录129和/或一个或多个数据源130(例如,关系数据库)的访问,以供多

维数据库环境使用。

[0024] 根据一实施例,一个或多个数据源还可以由一个或多个多维数据库服务器经由ODBC/JDBC 132或其它类型的接口来访问,以用于提供多维数据库。

[0025] 根据一实施例,客户端层140可以包括一个或多个多维数据库客户端142(例如,Essbase服务器客户端),其能够访问多维数据库(例如,Smart View(智能视图)、Spreadsheet Add-in(电子表格加载项)、Smart Search(智能搜索)、Administration Services(管理服务)、MaxL、XMLA、CAPI或VB API应用,Oracle Business Intelligence Enterprise Edition Plus(Oracle商业智能企业增强版)或其它类型的多维数据库客户端)。客户端层还可以包括与中间层中的服务一起使用的控制台,例如管理服务控制台144或工作室/集成服务控制台146。

[0026] 根据一实施例,可以通过TCP/IP、HTTP或其它类型的网络通信协议中的一个或多个来提供客户端层、中间层和数据库层之间的通信。

[0027] 根据一实施例,多维数据库服务器可以集成来自一个或多个数据源的数据,以提供多维数据库、数据结构或立方体150,然后可以对其进行访问以向终端用户提供过滤后的信息。

[0028] 通常,多维数据库中的每一个数据值都存储在立方体的一个单元中;并且可以通过指定特定的数据值的沿立方体的维度的坐标来引用该特定的数据值。来自一个维度的成员与来自一个或多个其它维度中的每一个维度的成员的交集表示数据值。

[0029] 例如,如图1所示,其示出了可以在面向销售量的业务应用中使用的立方体162,当查询指示“Sales(销售量)”时,系统可以将该查询解释为数据库内的一片或一层数据值164,该数据库包含所有“Sales”数据值,其中“Sales”与“Actual(实际)”和“Budget(预算)”相交。为了参考多维数据库中的特定数据值166,查询可以指定每一个维度上的成员,例如,通过指定“Sales、Actual、January(一月)”。以不同的方式对数据库进行切片,提供了数据的不同视角;例如,针对“February(二月)”的一片数据值168检查时间/年份维度被固定为“February”的所有那些数据值。

[0030] 数据库大纲

[0031] 根据一实施例,多维数据库的开发始于数据库大纲的创建,该数据库大纲定义了数据库中的成员之间的结构关系;组织数据库中的数据;以及定义合并和数学关系。在数据库大纲的层次树或数据结构内,每一个维度包括一个或多个成员,这些成员继而可以包括其它成员。维度的指定指示系统如何合并其各个成员的值。合并是树的一个分支内的一组成员。

[0032] 维度和成员

[0033] 根据一实施例,维度表示数据库大纲中的最高合并级别。可以选择标准维度来表示与部门职能相关的业务计划的组成部分(例如,时间、帐户、产品线、市场、部门)。与标准维度相关联的属性维度使用户能够基于成员属性或特性对标准维度的成员进行分组和分析。成员(例如,产品A、产品B、产品C)是一个维度的各个组成部分。

[0034] 维度和成员关系

[0035] 根据一实施例,多维数据库使用家庭(父、子、同胞;后代和祖先)和层次(代和级;根和叶)术语,来描述数据库大纲内的成员的角色和关系。

[0036] 根据一实施例,父是在其下具有分支的成员。例如,“Margin(差额)”可以是“Sales”和“Cost of Goods Sold(销售商品成本)”(COGS)的父。子是在其上有父的成员。在上面的示例中,“Sales”和“Cost of Goods Sold”是父“Margin”的子。同胞是同一代内的同一直系父的子。

[0037] 根据一实施例,后代是父下方的分支中的成员。例如,“Profit(利润)”、“Inventory(库存)”和“Ratios(比率)”可以是Measures(度量)的后代;在这种情况下,“Profit”、“Inventory”和“Ratios”的子也是Measures的后代。祖先是成员上方分支中的成员。在上面的示例中,“Margin”、“Profit”和Measures可以是“Sales”的祖先。

[0038] 根据一实施例,根是一个分支中的顶部成员。例如,Measures可以是“Profit”、“Inventory”和“Ratios”的根;并且对于“Profit”、“Inventory”和“Ratios”的子也是根。叶(第0级)成员没有子。例如,打开“Inventory”、添加和结束“Inventory”可以是叶成员。

[0039] 根据一实施例,代是指维度内的合并级。树的根分支被认为是“第1代”,并且代数从根成员向叶成员增加。级是指维度内的分支;并且以与代所用的编号顺序相反的顺序进行编号,级号从叶成员向根成员减小。

[0040] 根据一实施例,用户可以将名称分配给代或级,并且将该名称用作该代或级中的所有成员的简写。

[0041] 稀疏和密集维度

[0042] 多维数据库内的数据集常具有两种特性:数据不平滑且不均匀地分布;并且对于大多数成员组合不存在数据。

[0043] 根据一实施例,为了解决这个问题,系统可以识别两种类型的标准维度:稀疏维度和密集维度。稀疏维度是被填充的可用数据位置所占百分比相对较低的维度;而密集维度是这样的维度,其中每一个维度组合中都有一个或多个单元被占用的概率相对较高。许多多维数据库固有地是稀疏的,因为它们对于大多数成员组合缺少数据值。

[0044] 数据块和索引系统

[0045] 根据一实施例,多维数据库使用数据块和索引来存储和访问数据。系统可以为稀疏标准维度成员的每个唯一组合创建多维数组或数据块,其中每一个数据块表示其稀疏维度成员的组合的密集维度成员。为每一个数据块创建一个索引,其中该索引表示稀疏标准维度成员的组合,并且包括针对存在至少一个数据值的稀疏标准维度成员的每个唯一组合的入口或指针。

[0046] 根据一实施例,当多维数据库服务器搜索数据值时,它可以使用索引提供的指针来定位适当的数据块;并且,在该数据块内定位包含该数据值的单元。

[0047] 管理服务

[0048] 根据一实施例,管理服务(例如,Essbase管理服务)提供使用户能够设计、开发、维护和管理服务器、应用和数据库的单点访问。

[0049] 工作室

[0050] 根据一实施例,工作室(例如,Essbase Studio)提供向导驱动用户界面,以进行与数据建模、立方体设计和分析应用构建有关的任务。

[0051] 电子表格加载项

[0052] 根据一实施例,电子表格加载项将多维数据库与电子表格集成,其为诸如

Connect、Pivot、Drill-down和Calculate的增强命令提供支持。

[0053] 集成服务

[0054] 根据一实施例,集成服务(例如,Essbase集成服务)提供元数据驱动环境,以用于多维数据库中存储的数据与关系数据库中存储的数据之间的集成。

[0055] 提供者服务

[0056] 根据一实施例,提供者服务(例如,Hyperion提供者服务)用作用于Java API、Smart View和XMLA客户端的数据源提供者。

[0057] 智能视图

[0058] 根据一实施例,智能视图提供用于例如Hyperion Financial Management、Hyperion Planning和Hyperion Enterprise Performance Management Workspace数据的公共接口。

[0059] 开发者产品

[0060] 根据一实施例,开发者产品能够快速创建、管理和部署定制的企业分析应用。

[0061] 生命周期管理

[0062] 根据一实施例,生命周期管理(例如,Hyperion Enterprise Performance Management System Lifecycle Management)提供了用于使企业绩效管理产品能够跨产品环境迁移应用、储存库或单个人工制品的手段。

[0063] OLAP

[0064] 根据一实施例,在线分析处理(OLAP)提供了使用户能够分析企业数据的环境。例如,财务部门可以将OLAP用于诸如预算、基于活动的成本核算、财务绩效分析和财务建模的应用,以提供“准时制(just-in-time)”信息。

[0065] 根据一实施例,OLAP系统可以多个维度组织数据,从而允许数据集的搜索者/用户进行遍历各个维度的定向搜索,以最终得到感兴趣的结果。OLAP系统可以将数据视为驻留于维度的交集处。换句话说,可以将OLAP下的数据组织和存储为多维数据库,该多维数据库是所有维度的叉积的实例化。这允许用户/搜索者以自组织的方式沿感兴趣的维度遍历详细信息的层次结构,以获得特定的目标数据。缓慢变化的数据可以表示为当前数据集中的元数据。

[0066] 混合多维数据库

[0067] 根据一实施例,系统支持在多维数据库(例如,Essbase)计算环境中使用动态流(在本文的一些示例中称为查询处理动态流(QPDF))。动态流处理使得能够混合使用例如聚合存储选项(ASO)、块存储选项(BSO)或其它类型的存储容器,并提供通用流来以自下而上的模式处理接收到的输入查询。该方法可用于减小立方体的大小,其提供动态成员的高效计算。

[0068] 例如,根据一实施例,对于访问动态稀疏成员的查询,系统可以使用聚合存储引擎来满足请求。对于聚合存储引擎无法处理的查询,系统可以采用块存储引擎来满足请求,包括例如将数据带入聚合存储临时表空间。

[0069] 例如,根据一实施例,当由计算机系统执行时,动态流处理可以在多维数据库上操作以:(1)扩展输入查询来找到所有基础/计算的数据;(2)分析扩展的查询来找到依赖性和计算顺序;(3)根据前面的步骤定义计算单元;(4)用定义的计算单元建立处理流,并将它们

连接;(5)执行处理流,并确定对输入查询的响应。

[0070] 图2示出了根据一实施例的针对多维数据库使用动态流。

[0071] 在典型的多维环境中,为了准备系统以响应输入查询,数据库服务器预先计算某些维度的值,并将这些预先计算的值存储在立方体中以供以后查找。

[0072] 根据一实施例,当代之以使用动态流时,支持动态查询处理的能力使数据库服务器能够避免预先计算和存储这些值,这提高了性能并减少了潜在的空单元的存储。

[0073] 如图2所示,根据一实施例,该系统可以包括一个或多个查询处理器200,例如多维表达式(MDX)查询处理器202和/或电子表格提取器(SSE)204查询处理器,所述查询处理器200使得能够从客户端接收206输入查询208,从而检索、访问或以其他方式检查来自数据源的数据集,其由多维数据库提供并可以经由多维数据库访问。

[0074] 根据一实施例,预处理器组件210可包括数据检索层212或数据取回组件(在某些环境中,其可并入存储在存储器中的基于内核的里程表检索器或里程表或数据结构,所述里程表检索器或里程表或数据结构管理指向数据块的指针,包含控制信息或者以其他方式充当指向已存储成员的指针的一个或多个数组)、聚合器组件214和计算器组件216,这些层和组件中的每一个都可以设置为可由计算机系统执行的软件或程序代码。

[0075] 通常,根据一实施例所描述的,预处理器从一个或多个查询处理器接收218输入查询,以针对多维数据库进行处理。聚合器适于进行数据的分层聚合。计算器适于对数据进行计算,并与聚合器协作(如下文进一步所述),以对立方体内的填充和/或搜索的至少一项使用数据检索层(适当地包括里程表),并处理对输入查询的响应。

[0076] 根据一实施例,该系统可以包括一个或多个存储容器220,例如,聚合存储选项(ASO)222、块存储选项(BSO)224或其它类型的存储容器226中的一个或多个,每一个所述存储容器可以充当从数据源或多维数据库读取/写入230到数据源或多维数据库的数据与在预处理器处进行聚合和计算可能需要的任何数据之间的接口。

[0077] 图3进一步示出了根据一实施例的针对多维数据库使用动态流。

[0078] 如图3所示,根据一实施例,响应于数据库服务器接收到输入查询,作为动态流244的一部分,聚合器可以与计算器结合地操作240、242以处理查询,其可以类似地被设置为可由计算机系统执行的软件或程序代码。

[0079] 例如,如图3所示,动态流处理使得在本示例中能够混合使用一个或多个ASO、BSO或其它类型的存储容器,并提供通用流来以自下而上的模式使用这些存储容器处理查询。

[0080] 根据一实施例,当系统开始处理输入查询时,它首先根据对输入查询的检查来确定需要检索哪些特定数据或其它信息,即元数据。然后,系统可以为该输入查询定义246初始计算单元250,该初始计算单元250封装将从存储容器检索数据集的聚合/计算过程。

[0081] 根据一实施例,数据缓冲器260(在本文中,在一些示例中被称为一个或多个输出桶)作为数据结构操作,每一个计算单元都可以在该数据结构中读取/写入数据252,并且该数据结构允许临时存储从存储容器接收254的数据,以供计算单元消费。

[0082] 根据一实施例,当动态流与BSO型存储容器一起使用时,动态流处理对输入查询进行预分析,并将请求点扩展到其基础数据。

[0083] 然而,这样扩展的基础数据的量可能相当大。

[0084] 为了解决这个问题,并减少扩展数据的量,根据一实施例,可以在从内核取回数据

期间进行第一动态聚合,而不是对相关内核结构(例如,如上所述的内核侧里程表)进行完全扩展。

[0085] 根据一实施例,然后,动态流进行操作以扩展输入查询,找到所有基础/计算数据;并分析扩展的查询以找到依赖性和计算顺序。

[0086] 依赖性分析

[0087] 根据一实施例,多维数据库(例如,Essbase)可以包括“m”个维度,每一个维度具有“n”个成员,其中一个维度的每一个成员可以与加载的输入值或动态成员一起存储,当在查询中询问该动态成员时,在实际检索期间该动态成员的值在运行时被计算。维度成员本质上是分层次的。维度的基数可以是维度成员的总数,并且在所有维度成员之中制定的每一个组合表示多维数据库立方体中的交集。每一个立方体可以具有与之关联的存储值或计算值,并且一个组合的每个坐标表示交集值的含义(例如,诸如业务含义的提取信息)。

[0088] 根据一实施例,多维数据库的动态成员可以包括具有有效数学公式的成员或临时成员(经由与多维数据库立方体相关联或可与之通信的语言(例如,对于Essbase的MDX)根据请求创建的)。这样的公式可以是简单的算术公式或复杂公式,该复杂公式涉及对来自该成员所依赖的同一大纲的其它成员的计算而得到的值的复杂条件计算。因此,这种动态成员的公式表示其值,并且对每一个交集进行评估。

[0089] 根据一实施例,动态成员可以依赖于来自同一大纲的其它成员,所述其他成员被称为公式的依赖者,自然地,为了评估用于每一个交集的原始公式,必须首先计算这些依赖者。因此,为了评估混合数据聚合模型(例如,“Hybrid Essbase”(自下而上))中的动态成员的公式,系统可以首先准备该公式所依赖的所有成员的列表。此外,动态成员的依赖者列表可以被分类为“运行时依赖者”和“静态依赖者”。动态成员的运行时依赖者是对于每个交集不同的依赖者,而静态依赖者是与交集无关的恒定的那些依赖者。这种识别动态成员的运行时依赖者以及静态依赖者的分析处理被称为混合数据聚合模型中的“依赖性分析”。

[0090] 根据一实施例,用于块存储数据库的混合聚合是聚合模型,其中以类似于聚合存储数据库的效率来执行块存储数据计算。混合聚合通过去除稀疏聚合、减小尺寸和存储器占用以及加快批处理例程的速度,提供了快速执行的好处。部署考虑被简化,因为用户不再需要考虑是对大量使用的第0级计算采用块存储,还是对许多上级聚合使用聚合存储,还是设计其中沿维度线拆分立方体以促进计算性能的分区模型。在块存储数据库中,必须存储大的稀疏维度:使它们动态化将导致查询时的块I/O过多,从而影响性能。非常大的存储的稀疏维度可以导致冗长的批处理聚合时间,以及随稀疏维度的数量和尺寸而增长的大的数据库尺寸。即使有这些缺点,块存储也因其强大的功能而被广泛使用。聚合存储被设计为实现具有更多和更大维度的大型数据库。与块存储不同,它不需要预先聚合大的稀疏维度以实现良好的查询性能。关键在于聚合存储引擎,该引擎促进了跨大维度的快速动态聚合。混合在可能时利用ASO计算引擎,并在需要时切换到BSO计算引擎。

[0091] 根据一实施例,依赖性分析可以形成多维数据库环境中的混合流的一部分,这使得用户能够通过在开始其实际评估之前识别其所有所需的依赖者来以自下而上的方式评估动态成员。本文所述的依赖性分析通过消除冗余和过时的递归的自上而下的行程来提高性能。

[0092] 根据一实施例,对每一个动态(或临时)成员进行依赖性分析,以收集BSO(块存储

选项)和ASO(聚合存储选项)立方体中的运行时依赖者以及静态依赖者的列表。这使成员能够加入混合流,以便以自下而上的方式执行。

[0093] 对于以下示例,根据一实施例,考虑具有如下所示的样本公式的动态成员,其示出了示例性的动态成员。行之间的斜体文本讨论了上方行中的功能

[0094] 1)IF(@ISLEV("Market",0)and(@ISLEV("SITE",1)

[0095] 如果用户查询中的当前成员在"Market(市场)"维度的第0级,以及如果用户查询中的当前成员在"SITE(地点)"维度的第1级。

[0096] 2)IF(@ISMBR("NewYork"))

[0097] 以及,如果当前成员是"New York(纽约)"

[0098] 3)@PARENTVAL("Market","Sales");

[0099] 则返回在维度"Market"和"Sales"处的父值

[0100] 4)ELSE

[0101] 5)6;

[0102] 否则返回"6"

[0103] 6)ENDIF;

[0104] 7)ELSE

[0105] 8)@PARENTVAL("Product","Sales");

[0106] 如果以上的条件不满足,则返回在维度"Market"和"Sales"处的父值

[0107] 9)ENDIF;

[0108] 根据一实施例,在上述动态成员之后,对依赖性分析要采取两个主要步骤。在第一步骤中,在服务器启动期间,基于动态成员进行依赖性分析。在第二步骤中,使用第一步骤的结果来取回实际依赖者。

[0109] 第一步骤-在服务器启动期间进行依赖性分析

[0110] 根据一实施例,第一步骤是检测运行时依赖者维度的列表和动态公式中存在的依赖性模式。该逻辑在服务器启动期间执行,因此在实际检索期间它不会增加MDX查询的周转时间。

[0111] 根据一实施例,在加载立方体(例如,服务器启动)期间,本文的系统和方法可以读取立方体的整个大纲、每一个成员,并查看每个成员的公式以收集每个成员的依赖者。所有这些依赖性信息都是在服务器启动期间收集的。因为所有依赖性信息是静态的,因此这种采集步骤不依赖于于用户查询。然后,在调用用于动态成员的查询时,可以存储并调用该依赖性映射。

[0112] 根据一实施例,依赖性分析的第一步骤的输入是将动态公式分解为令牌化字符串的列表。

[0113] 根据一实施例,这种步骤的输出至少有两个部分。第一输出是维度数组,动态公式在该维度数组上具有"运行时"依赖性。在查看上述动态成员的示例时,运行时依赖性维度为Product和Market。这些维度将在运行时在对该公式成员的实际检索期间基于存储的依赖性映射生成运行时依赖者成员的确切列表(例如,形成上面示例中的依赖性维度Product和Market的依赖性映射的那些维度,无论是静态维度还是运行时维度)。

[0114] 根据一实施例,第二输出是对公式中存在的任何模式的检测,其表明,该公式在其

“IF”(条件逻辑/条件语句)条件中使用的“上下文”依赖者维度的列表与该公式的每一个“IF”的主体部分中使用的不同还是相同。当公式中的IF-ELSE嵌套增加时,该模式检测将变得更加复杂。

[0115] 根据一实施例,在以下示例中,术语“立方体/数据库”可以意指包括多个维度的实际服务器或立方体或数据库,其中每一个维度的成员是分层的,并且可以在大纲或树状视图中表示。此外,术语MDX可以指在多维数据库中使用的多维表达式(MDX)。另外,术语“自上而下或运行时依赖函数”可以指依赖于实际查询并且其结果值是在实际运行的查询的上下文中评估的函数。可以有许多自上而下的函数,例如:@PARENT、@CURRMBR等。

[0116] 根据一实施例,以上述示例为例,来自第一阶段的高级输出将包括:动态成员所依赖的维度的列表,确定是否存在任何IF ELSE模式,以及仅由运行时依赖函数在“IF”条件中使用的维度的列表。在高级输出处,上面的示例所依赖的维度的列表包括Market和Product。该示例中有IF ELSE模式。并且“IF”条件中使用的维度列表包括Market、SITE。

[0117] 根据一实施例,第一阶段依赖性分析可以被提供为针对每一个查询创建的栈(即,函数栈和自变量栈)中的方法。这些栈可以基于查询中的每一个调用链。

[0118] 根据一实施例,如上所述,用于依赖性分析的输入可以是查询的公式,该公式被分解为令牌化列表。然后,系统和方法可以声明内部使用的所有必要变量,并将它们分类到相应的栈中。

[0119] 根据一实施例,以下伪代码表示依赖性分析中采取的主要步骤:

[0120] For each输入令牌列表中的字符串令牌

[0121] Do

[0122] 清空内部的两个栈:函数栈和自变量栈

[0123] 定位输入列表中的第一函数名条目

[0124] 如果找到第一函数,则将其推到函数栈上

[0125] While(不是当前调用链的末尾(通过匹配打开和关闭的自变量(诸如括号)的数量来检测))

[0126] {

[0127] If(令牌是函数名)

[0128] {

[0129] 取得函数的记录并存储指针;

[0130] 将函数名记录推到函数栈上

[0131] }

[0132] Else if(令牌是函数使用的自变量)

[0133] {

[0134] 将该自变量推到自变量栈上

[0135] }

[0136] Else if(令牌是自变量分隔符)

[0137] Continue

[0138] }

[0139] If(找到要分析的调用链)

[0140] {

[0141] 调用函数以分析个体调用链

[0142] }

[0143] 清空内部的两个栈

[0144] Done

[0145] 接下来,可以以示例的形式描述上述伪代码。对于以下示例,可以参考图4使用以下大纲公式,图4示出了根据一实施例的示例性函数栈。

[0146] @WeightedSumX(@Range("Entered Delta",USD:ZAR),_FCCS_Rates_,"Rate.Average",@CONCATENATE("Rate_",@NAME(@CURRMBR(Currency)))+@PRIOR("Reporting");

[0147] 根据一实施例,如图所示,从上述公式产生三个单独的函数栈。第一函数堆栈410包括函数A(@WeightedSumX)401和函数B(@RANGE)402。这表示第一函数调用链的末尾,该调用链中的其余值"Entered Delta"和USD:ZAR是自变量,其将被放置在自变量栈中。

[0148] 根据一实施例,第二函数栈420包括函数A(@WeightedSumX)401、函数C(@CONCATENATE)403、函数D(@NAME)404和函数E(@CURRMBR)405。这表示第二函数调用链的末尾,该调用链中的其余值"Rate_"和Currency是自变量,其将被放置在自变量栈中。

[0149] 根据一实施例,第三函数栈430包括函数F(@PRIOR)406。这表示第三函数调用链的末尾,该调用链中的其余值"Reporting"是自变量,其将被放置在自变量栈中。

[0150] 根据一实施例,可以分析以上示例中的每一个调用链,并且可以标注运行时维度。在上面的示例中,运行时维度为Currency和Reporting。

[0151] 根据一实施例,以上示例不包含任何IF-ELSE模式。

[0152] 第二阶段:使用第一阶段的输出取回实际依赖者

[0153] 根据一实施例,一旦已进行了第一阶段(注意第一阶段可以在立方体的生命周期中执行一次—在服务器启动期间该立方体被首次加载时),第二阶段就可以被执行。下面描述的用于取回实际的运行时依赖者的第二阶段是具体的,并且依赖于检索请求(例如,MDX查询)。因此,实际取回依赖者的逻辑被执行以同于实现检索请求(即,在实际的查询运行时期间)。在依赖性分析的第一阶段中收集的信息用于查找和收集查询运行时依赖者。

[0154] 根据一实施例,在第二阶段中,系统和方法将查询里程表和(从第一阶段找到的)运行时依赖者列表作为其输入。

[0155] 根据一实施例,接下来,系统和方法从第一阶段中检测到的仅运行时依赖者维度的里程表来制定成员的组合(或交集)。

[0156] 根据一实施例,对于每一个交集,系统和方法执行动态成员的内部预编译程序,以取回对每一个交集而言是特定且不同的依赖者成员。然后,系统和方法可以将针对每一个交集找到的所有依赖累积为输出,并将其填入查询的扩展里程表中。该步骤经过栈机器以仅针对元数据来评估程序。多维数据库已经知晓针对值检索执行了公式程序。然而,系统和方法允许执行相同的公式,并且通过跳过值检索而仅取回依赖者成员。

[0157] 根据一实施例,考虑下面的动态成员。对于本示例,动态成员将被命名为"Test(测试)"。

```
1）IF(@ISLEV(“Market”, 0) and (@ISLEV(“SITE”, 1)
2）   IF(@ISMBR(“NewYork”))
3）      @PARENTVAL(“Market”, “Sales”);
4）   ELSE
5）         6;
6）   ENDIF;
7）ELSE
8）   @PARENTVAL(“Product”, “Sales”);
9）ENDIF;
```

[0158]

[0159] 图5示出了根据一实施例的示例性数据集。

[0160] 在图5内,示出了多个维度,例如市场500、地点510、产品520和销售量530。

[0161] 根据一实施例以及以下示例的目的,假设用户已提交以下请求以检索MDX查询。

[0162] SELECT{[Cola],[Old Fashioned],[Dark Cream],[Grape]}ON ROWS,{[East].Children,[East]}ON COLUMNS FROM[TPDNTest.TPDNTest]WHERE([Jan],[Test],[Scenario]);

[0163] 根据一实施例,[TPDNTest.TPDNTest]可以包括出于测试目的而创建的内部立方体。运行动态成员“Test”,可以得到图6所示的数据集。

[0164] 根据一实施例,在考虑动态成员“Test”时,第一阶段(依赖性分析)返回维度Market和Product作为运行时依赖性维度。“Test”也具有IF-ELSE模式。第二阶段将该信息以及输入里程表一起作为输入,以取回实际运行时依赖者。因此,作为第二阶段的输出,新发现的依赖者是Measures维度中的Sales。此外,Product维度中有四个成员,即Colas、Root Beer、Cream Soda和Fruit Soda。

[0165] 根据一实施例,输出可以包括:第一输出是维度数组,该公式在其上具有“运行时”依赖性,即,在上面的示例中的两个维度:Product、Market。这些维度可以在运行时在实际检索该公式成员期间生成运行时依赖者成员的确切列表。

[0166] 根据一实施例,下一输出是:对公式中存在的模式的检测,这表明,该公式在其“IF”条件中使用的依赖者维度的列表与该公式的每个“IF”的主体部分中使用的是否不同。当在公式中IF-ELSE嵌套增加时,这种模式检测变得更加复杂,并且这种模式检测是依赖性分析的在运行时正确取回每一个交集的依赖者成员的至关重要的部分。

[0167] 根据一实施例,下面提供了第二阶段的步骤的概要。

[0168] 根据一实施例,第二阶段可以在来自输入里程表的运行时依赖者维度的所有成员(Market和Product)中形成所有数学组合,并对每一个组合执行程序以取回依赖者。

[0169] 根据一实施例,来自Market和Product维度的输入里程表成员是:

[0170] Product:{[100-10],[200-10],[300-10],[400-10]}OR{[Cola],[Old Fashioned],[Dark Cream],[Grape]}其中第二集合是对应产品名称的别名

[0171] Market:{[New York],[Massachusetts],[Florida],[Connecticut],[New

Hampshire],[East]}

[0172] 根据一实施例,全部的组合如下所示:

[0173] {([100-10],[New York]),([100-10],[Massachusetts]),([100-10],[Florida]),([100-10],[Connecticut]),([100-10],[New Hampshire]),([100-10],[East]),

[0174] ([200-10],[New York]),([200-10],[Massachusetts]),([200-10],[Florida]),([200-10],[Connecticut]),([200-10],[New Hampshire]),([200-10],[East]),

[0175] ([300-10],[New York]),([300-10],[Massachusetts]),([300-10],[Florida]),([300-10],[Connecticut]),([300-10],[New Hampshire]),([300-10],[East]),

[0176] ([400-10],[New York]),([400-10],[Massachusetts]),([400-10],[Florida]),([400-10],[Connecticut]),([400-10],[New Hampshire]),([400-10],[East])}

[0177] 根据一实施例,然后,第二阶段可以在该所有组合的列表上进行迭代,并且对于每一个组合,可以仅仅为了取回依赖者成员来评估成员“Test”的公式。第二阶段可以将这些成员作为依赖者分别添加到Measures、Market和Product的扩展里程表中。

[0178] {[Sales]},{[East]},{[100],[200],[300],[400]}

[0179] 根据一实施例,对于输入里程表,通过针对元数据执行一次公式来收集依赖者。该逻辑还重复了制定组合的同一处理以及执行用于查找更多依赖者的同一程序。此时,它仅在针对运行时依赖者维度的新添加成员之中制定组合。

[0180] 另外,根据一实施例,为了评估针对特定组合的公式,可以将公式的预编译可执行程序与输入组合一起馈送到栈机器框架中,该栈机器框架知道如何评估该程序。该栈机器是已有的作品,其知道要针对其值来评估公式而不是仅取回依赖者元数据。因此,该栈机器被增强有特征,该特征允许栈机器将公式的程序与组合一起作为其输入,并仅仅是为了取回作为其依赖者的元数据才执行该程序,而不涉及公式的值计算部分。其中执行程序仅仅是为了取回依赖者的栈机器的这种模式被称为“元数据”模式。

[0181] 根据一实施例,该处理的第二阶段可以在可容纳发现的依赖者成员的临时容器内执行。该结构可以用于形成成员的动态可扩大列表,并且可容纳在多维数据库内创建的任何对象。

[0182] 图7是根据一实施例的用于依赖性分析的示例性方法的流程图。

[0183] 根据一实施例,在步骤701,该方法可以接收输入参数。这包括动态成员以及里程表,该里程表包含用户的检索请求。

[0184] 根据一实施例,在步骤702,该方法检查动态成员是否是静态的并且不具有运行时依赖性。如果是这样,则在步骤703,该方法以默认组合将该公式程序传递给栈机器。该方法仅对默认组合执行一次程序以取回所有静态依赖者。然后,该方法将所有这些依赖者从临时容器转移到里程表中。

[0185] 根据一实施例,在步骤704,该方法检查动态成员是否具有运行时依赖性以及动态成员的公式中是否存在IF-ELSE模式。如果是这样,则在步骤705,该方法得到运行时依赖者

维度的列表。该方法在运行时依赖者维度的输入里程表上进行迭代。对于来自运行时依赖者维度的里程表中的成员的每一个组合(总数＝每一个里程表的尺寸的乘积),该方法:将当前公式程序和当前组合传递到栈机器以执行程序;当执行仅用于元数据的程序时,该方法从程序中排除所有值计算部分,并且仅执行元数据指令;并且该方法将从本次对当前cmi的程序运行中找到的依赖者存储到临时容器中。

[0186] 根据一实施例,此时,针对所有可能的组合执行程序,并且找到的所有依赖者都存在于临时容器中。现在,在该临时容器上迭代。

[0187] 根据一实施例,在步骤706,该方法针对临时容器中的每一个成员,将依赖者成员添加到查询的扩展里程表中。

[0188] 根据一实施例,在步骤707,该方法不需要所有组合。代替地,该方法选择较少数量的组合,其中每一个纵坐标在所有组合中仅出现一次。在这种情况下,组合的总数等于运行时依赖者的每一个里程表的尺寸的最大值。对于来自运行时依赖者维度的里程表的成员的每一个组合(总数＝每一个里程表的尺寸的最大值),该方法:将当前公式程序和当前组合传递到栈机器以执行程序;当仅针对元数据执行程序时,该方法从程序中排除所有值计算部分,并且仅执行元数据指令;然后,该方法将从本次对当前cmi的程序运行中找到的依赖存储到临时容器中。

[0189] 根据一实施例,此时,已经针对所有可能的组合执行了程序,并且找到的所有依赖都存在于临时容器中。现在,该方法在该临时容器上迭代。

[0190] 根据一实施例,针对临时容器中的每一个成员,该方法将依赖者成员添加到查询的扩展里程表中。

[0191] 根据一实施例,所有动态成员都被扩展,即,每一个动态成员的所有所需依赖者都是已知的,并且被添加到查询的扩展里程表中。然后,在多维数据库环境中的混合流将这个扩展后的里程表向前推进,该混合流以自下而上的方式检索完整的里程表,并将预期结果发送回用户。

[0192] 图8示出了用于多维数据库中的依赖性分析的示例性方法。

[0193] 在步骤810,该方法可以在包括一个或多个微处理器的计算机处提供:在该计算机上执行的多维数据库服务器,其中该多维数据库服务器支持至少一个多维数据立方体,其中该多维数据库包括多个维度,所述多个维度中的每一个维度包括多个成员;以及动态成员,其中该动态成员依赖于所述多个成员的集合。

[0194] 在步骤820,该方法可以进行依赖性分析以确定动态成员所依赖的所述多个成员的集合。

[0195] 尽管上面已经描述了本发明的各种实施例,但是应当理解,它们以示例而非限制的方式呈现。为了解释本发明的原理及其实际应用,选择和描述了这些实施例。这些实施例示出了这样的系统和方法,其中利用本发明通过提供新的和/或改进的特征和/或提供诸如减少资源使用、增加容量、提高效率和减少延迟的益处来提高该系统和方法的性能。

[0196] 在一些实施例中,本发明的特征全部或部分地在计算机中实现,该计算机包括处理器、诸如存储器的存储介质和用于与其它计算机通信的网卡。在一些实施例中,本发明的特征在分布式计算环境中实现,在该分布式计算环境中一个或多个计算机集群通过诸如局域网(LAN)、交换网网络(例如,InfiniBand)或广域网(WAN)之类的网络进行连接。分布式计

算环境可以具有在单个位置处的所有计算机,或者具有在通过WAN连接的不同的远程地理位置处的计算机集群。

[0197] 在一些实施例中,基于以自服务、可计量的方式使用Web技术将共享的、弹性的资源传递给用户,本发明的特征作为云计算系统的一部分或作为云计算系统的服务在云中全部或部分地来实现。云具有五种特性(由美国国家标准技术研究院定义):按需自服务;广泛的网络接入;资源池;快速弹性;以及可测量的服务。云部署模型包括:公共、私有和混合。云服务模型包括软件即服务(SaaS)、平台即服务(PaaS)、数据库即服务(DBaaS)和基础架构即服务(IaaS)。如本文中所使用的,云是硬件、软件、网络和Web技术的组合,它们以自服务、可计量的方式向用户传送共享的弹性资源。除非另有说明,否则本文所用的云包括公共云、私有云和混合云的实施例,并且所有云部署模型包括但不限于云SaaS、云DbaaS、云PaaS和云IaaS。

[0198] 在一些实施例中,使用或借助于硬件、软件、固件或其组合来实现本发明的特征。在一些实施例中,使用被配置或被编程为执行本发明的一个或多个功能的处理器来实现本发明的特征。在一些实施例中,处理器是设计用来执行本文所述的功能的单芯片或多芯片处理器、数字信号处理器(DSP)、片上系统(SOC)、专用集成电路(ASIC)、现场可编程门阵列(FPGA)或其它可编程逻辑器件、状态机、离散门或晶体管逻辑、离散硬件组件或其任何组合。在一些实施方式中,可以通过专门针对给定功能的电路来实现本发明的特征。在其它实施方式中,这些特征可以在处理器中实现,该处理器被配置为使用存储在例如计算机可读存储介质上的指令来执行特定功能。

[0199] 在一些实施例中,本发明的特征被并入软件和/或固件中,以控制处理和/或联网系统的硬件,并使得处理器和/或网络能够与利用本发明的特征的其它系统进行交互。这样的软件或固件可以包括但不限于应用代码、设备驱动程序、操作系统、虚拟机、管理程序、应用编程接口、编程语言和执行环境/容器。对软件领域的技术人员而言显而易见的是,熟练的程序员可以基于本公开的教导容易地准备适当的软件编码。

[0200] 在一些实施例中,本发明包括一种计算机程序产品,该计算机程序产品是其上/其中存储有指令的存储介质或计算机可读介质,这些指令可用于对诸如计算机的系统进行编程或以其他方式配置,以执行本发明的任何处理或功能。存储介质或计算机可读介质可以包括但不限于任何类型的盘,包括软盘、光盘、DVD、CD-ROM、微硬盘以及磁光盘,ROM、RAM、EPROM、EEPROM、DRAM、VRAM、闪存设备、磁卡或光卡、纳米系统(包括分子存储器IC)或适合用于存储指令和/或数据的任何类型的介质或设备。在特定实施例中,存储介质或计算机可读介质是非暂态存储介质或非暂态计算机可读介质。

[0201] 前述描述并非旨在穷举或将本发明限制于所公开的精确形式。另外,在已经使用特定系列的事物和步骤描述了本发明的实施例的情况下,对于本领域技术人员而言应该显而易见的是,本发明的范围不限于所描述的一系列事物和步骤。此外,在已经使用硬件和软件的特定组合描述了本发明的实施例的情况下,应当认识到,硬件和软件的其它组合也在本发明的范围内。此外,尽管各个实施例描述了本发明的特征的特定组合,但是应当理解,这些特征的不同组合均在本发明的范围内,这对于本领域技术人员而言将是显而易见的,从而使得可以将一个实施例的特征并入另一个实施例。而且,对于相关领域的技术人员而言显而易见的是,在不脱离本发明的精神和范围的情况下,可以在形式、细节、实现和应用

上进行各种增加、减少、删除、变化以及其它修改和改变。这意味着本发明的更广泛的精神和范围由所附权利要求及其等同物限定。

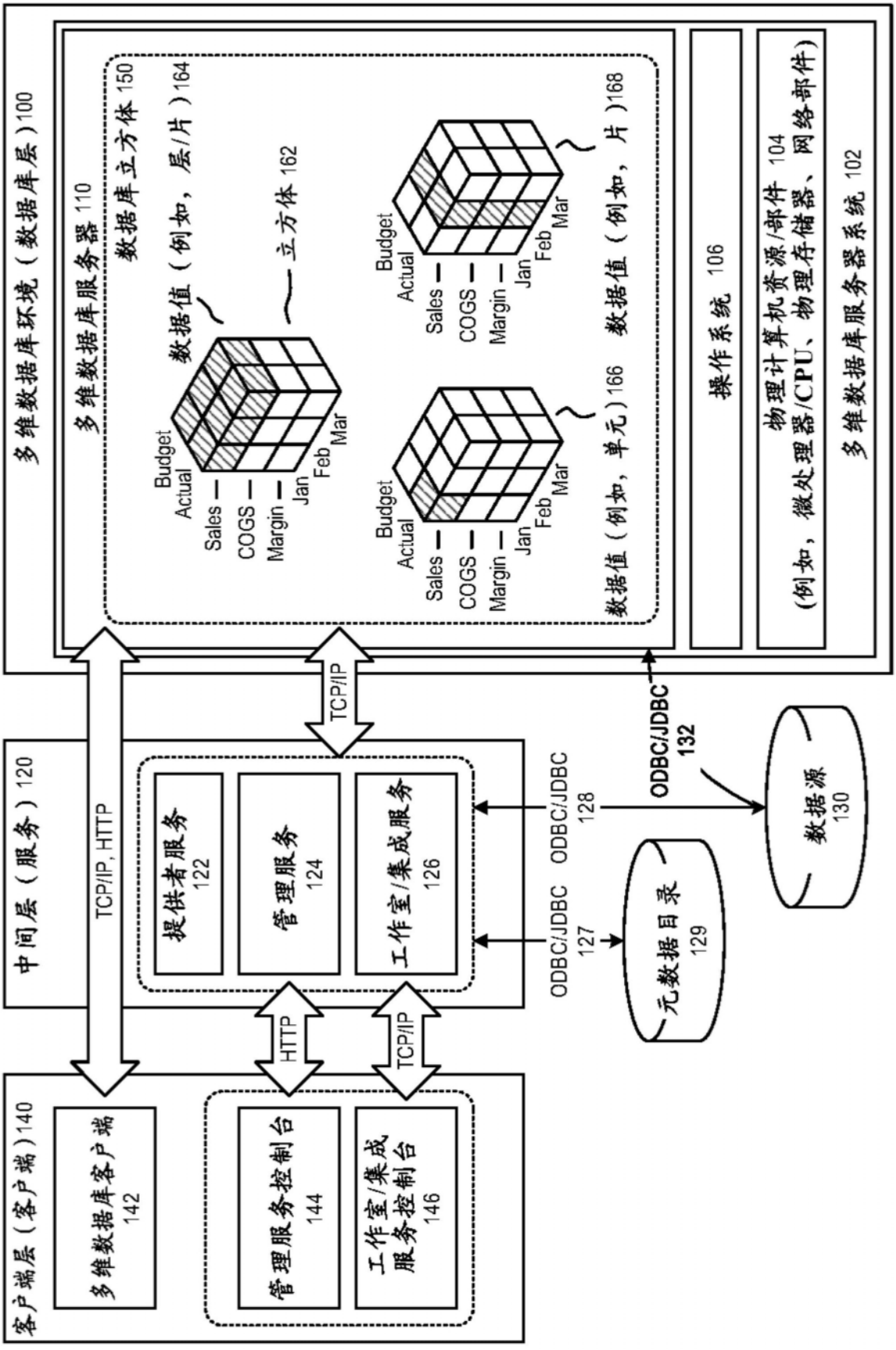

多维数据库环境（数据库层）100
多维数据库服务器 110
数据库立方体 150
数据值（例如，层/片）164
立方体 162
Budget
Actual
Sales
COGS
Margin
Jan
Feb
Mar
数据值（例如，单元）166
数据值（例如，片）168
操作系统 106
物理计算机资源/部件 104
（例如，微处理器/CPU、物理存储器、网络部件）
多维数据库服务器系统 102
中间层（服务）120
提供者服务 122
管理服务 124
工作室/集成服务 126
客户端层（客户端）140
多维数据库客户端 142
管理服务控制台 144
工作室/集成服务控制台 146
TCP/IP, HTTP
TCP/IP
HTTP
TCP/IP
ODBC/JDBC 127
ODBC/JDBC 128
ODBC/JDBC 132
元数据目录 129
数据源 130

图1

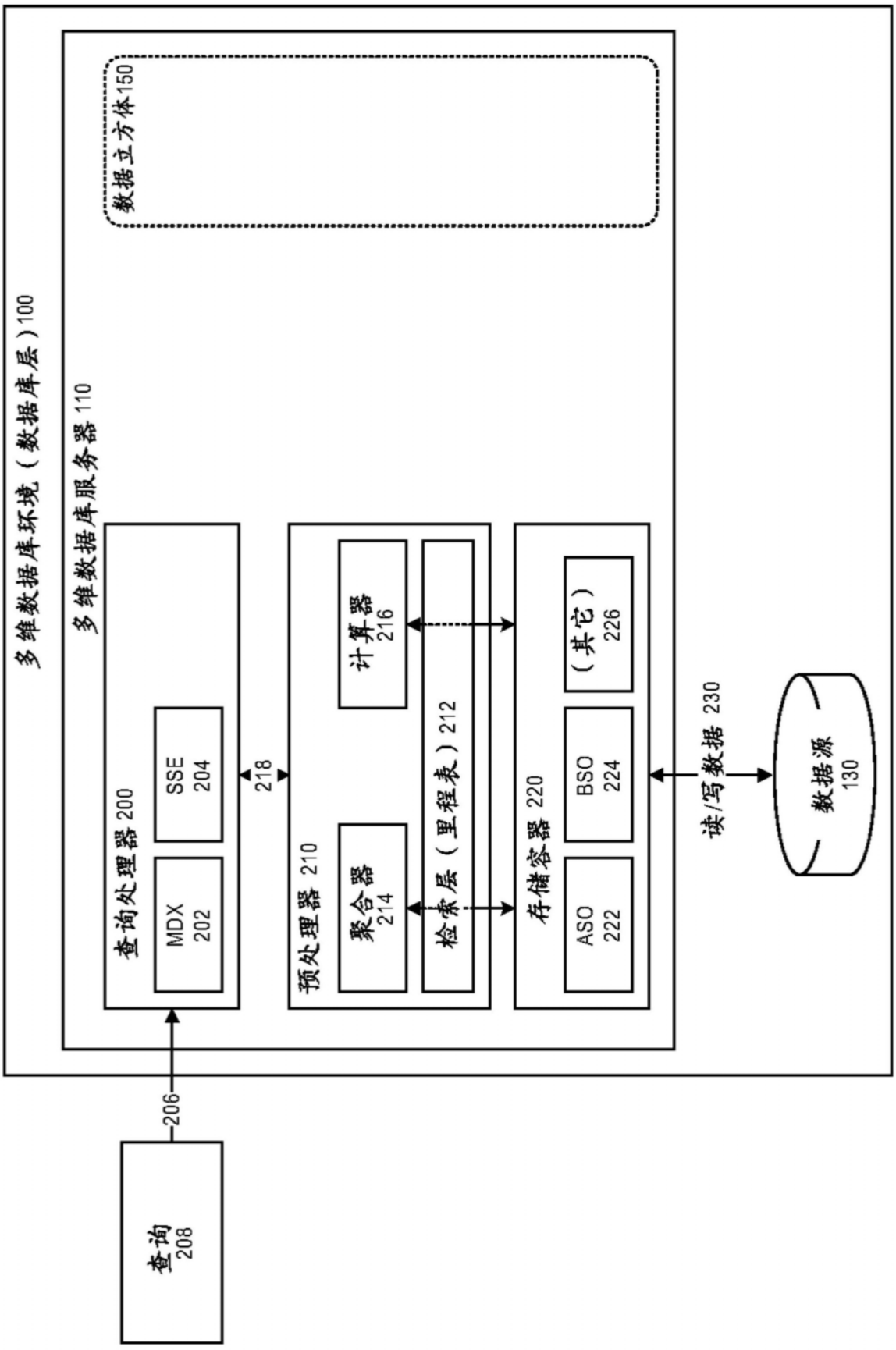

多维数据库环境（数据库层）100
多维数据库服务器 110
数据立方体150
查询处理器 200
MDX 202
SSE 204
218
预处理器 210
聚合器 214
检索层（里程表）212
计算器 216
存储容器 220
ASO 222
BSO 224
（其它）226
读/写数据 230
数据源 130
206
查询 208

图2

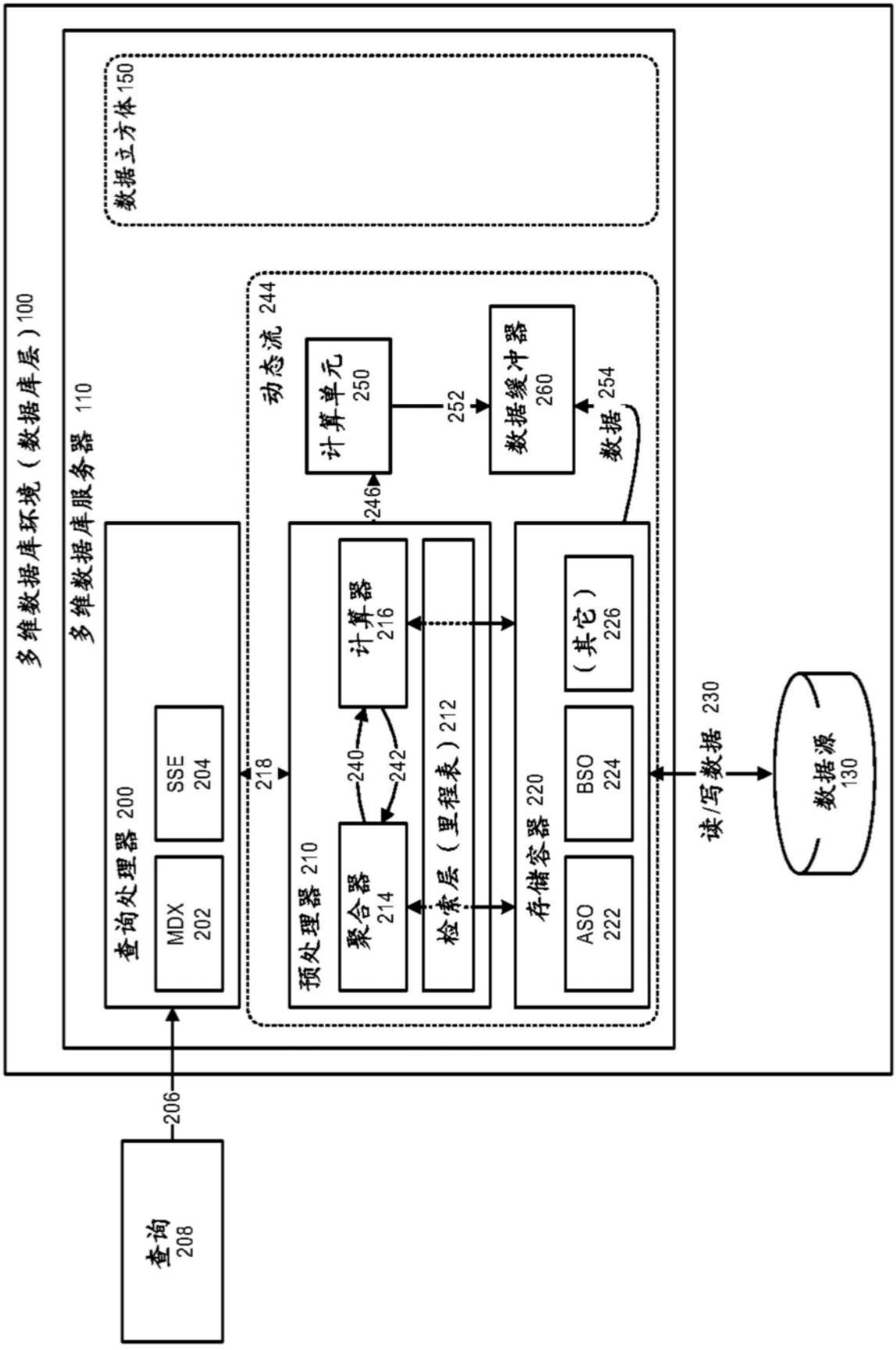

多维数据库环境（数据库层）100
多维数据库服务器 110
数据立方体 150
动态流 244
计算单元 250
252
数据缓冲器 260
254
数据
246
查询处理器 200
MDX 202
SSE 204
预处理器 210
聚合器 214
计算器 216
检索层（里程表）212
240
242
218
存储容器 220
ASO 222
BSO 224
（其它）226
读/写数据 230
数据源 130
查询 208
206

图3

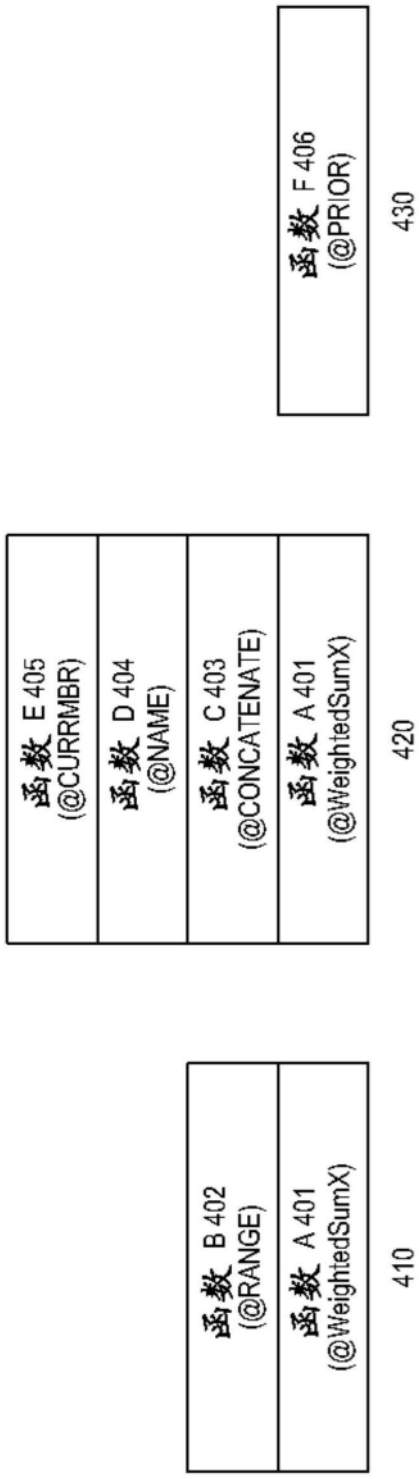
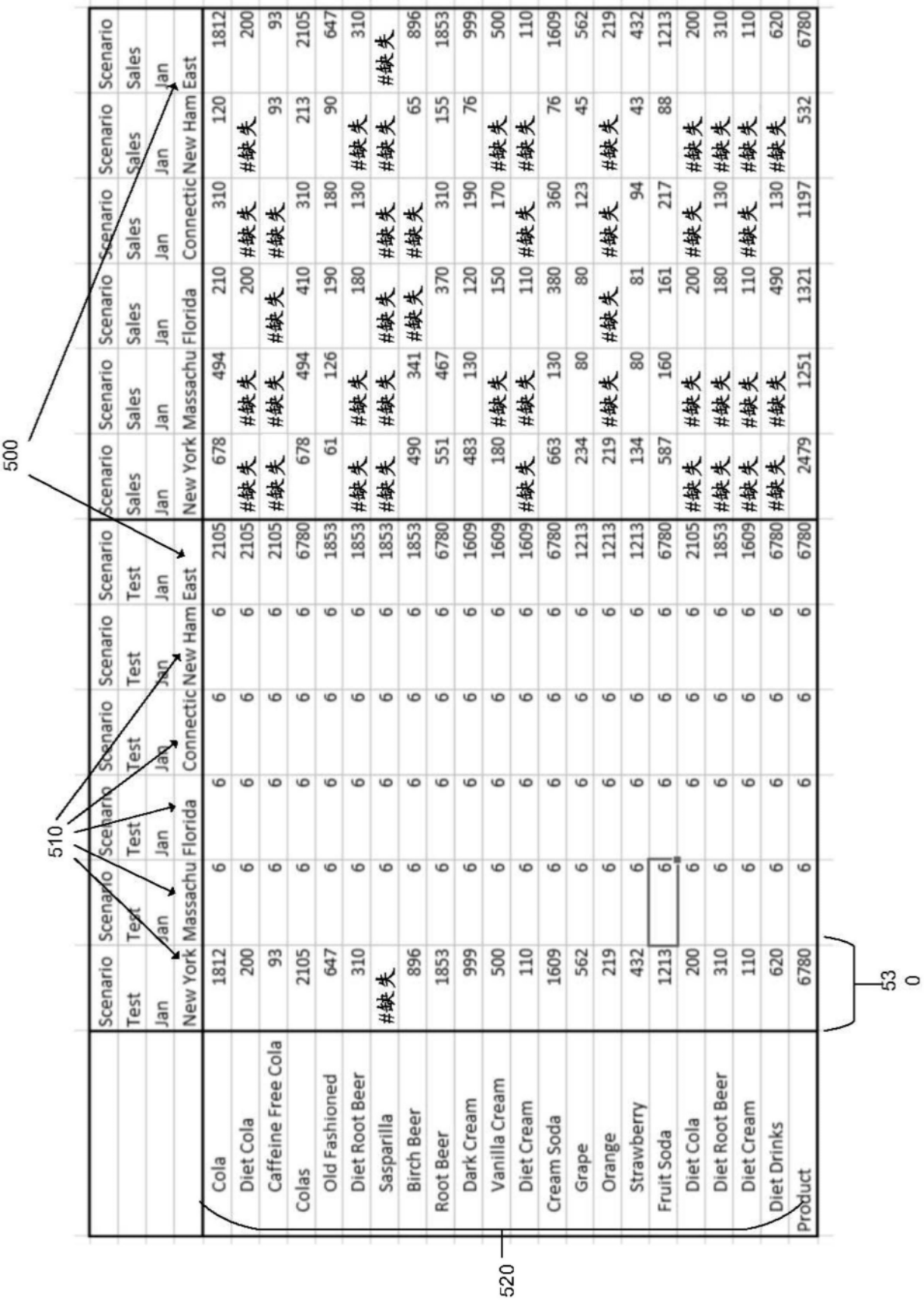

函数 B 402
(@RANGE)
函数 A 401
(@WeightedSumX)
410
函数 E 405
(@CURRMBR)
函数 D 404
(@NAME)
函数 C 403
(@CONCATENATE)
函数 A 401
(@WeightedSumX)
420
函数 F 406
(@PRIOR)
430

图4



	Scenario	Scenario	Scenario	Scenario	Scenario	Scenario	Scenario	Scenario	Scenario	Scenario	Scenario	Scenario	Scenario
	Test	Test	Test	Test	Test	Test	Sales	Sales	Sales	Sales	Sales	Sales	Sales
	Jan	Jan	Jan	Jan	Jan	Jan	Jan	Jan	Jan	Jan	Jan	Jan	Jan
	New York	Massachu	Florida	Connectic	New Ham	East	New York	Massachu	Florida	Connectic	New Ham	East	
Cola	1812	6	6	6	6	2105	678	494	210	310	120	1812	
Diet Cola	200	6	6	6	6	2105	#缺失	#缺失	200	#缺失	#缺失	200	
Caffeine Free Cola	93	6	6	6	6	2105	#缺失	#缺失	#缺失	#缺失	93	93	
Colas	2105	6	6	6	6	6780	678	494	410	310	213	2105	
Old Fashioned	647	6	6	6	6	1853	61	126	190	180	90	647	
Diet Root Beer	310	6	6	6	6	1853	#缺失	#缺失	180	130	#缺失	310	
Sasparilla	#缺失	6	6	6	6	1853	#缺失	#缺失	#缺失	#缺失	#缺失	#缺失	
Birch Beer	896	6	6	6	6	1853	490	341	#缺失	#缺失	65	896	
Root Beer	1853	6	6	6	6	6780	551	467	370	310	155	1853	
Dark Cream	999	6	6	6	6	1609	483	130	120	190	76	999	
Vanilla Cream	500	6	6	6	6	1609	180	#缺失	150	170	#缺失	500	
Diet Cream	110	6	6	6	6	1609	#缺失	#缺失	110	#缺失	#缺失	110	
Cream Soda	1609	6	6	6	6	6780	663	130	380	360	76	1609	
Grape	562	6	6	6	6	1213	234	80	80	123	45	562	
Orange	219	6	6	6	6	1213	219	#缺失	#缺失	#缺失	#缺失	219	
Strawberry	432	6	6	6	6	1213	134	80	81	94	43	432	
Fruit Soda	1213	6	6	6	6	6780	587	160	161	217	88	1213	
Diet Cola	200	6	6	6	6	2105	#缺失	#缺失	200	#缺失	#缺失	200	
Diet Root Beer	310	6	6	6	6	1853	#缺失	#缺失	180	130	#缺失	310	
Diet Cream	110	6	6	6	6	1609	#缺失	#缺失	110	#缺失	#缺失	110	
Diet Drinks	620	6	6	6	6	6780	#缺失	#缺失	490	130	#缺失	620	
Product	6780	6	6	6	6	6780	2479	1251	1321	1197	532	6780	

图5

	Jan	Test	Scenario			
	New York	Massachu	Florida	Connectic	New Ham	East
Cola	1812	6	6	6	6	2105
Old Fashioned	647	6	6	6	6	1853
Dark Cream	999	6	6	6	6	1609
Grape	562	6	6	6	6	1213

图6

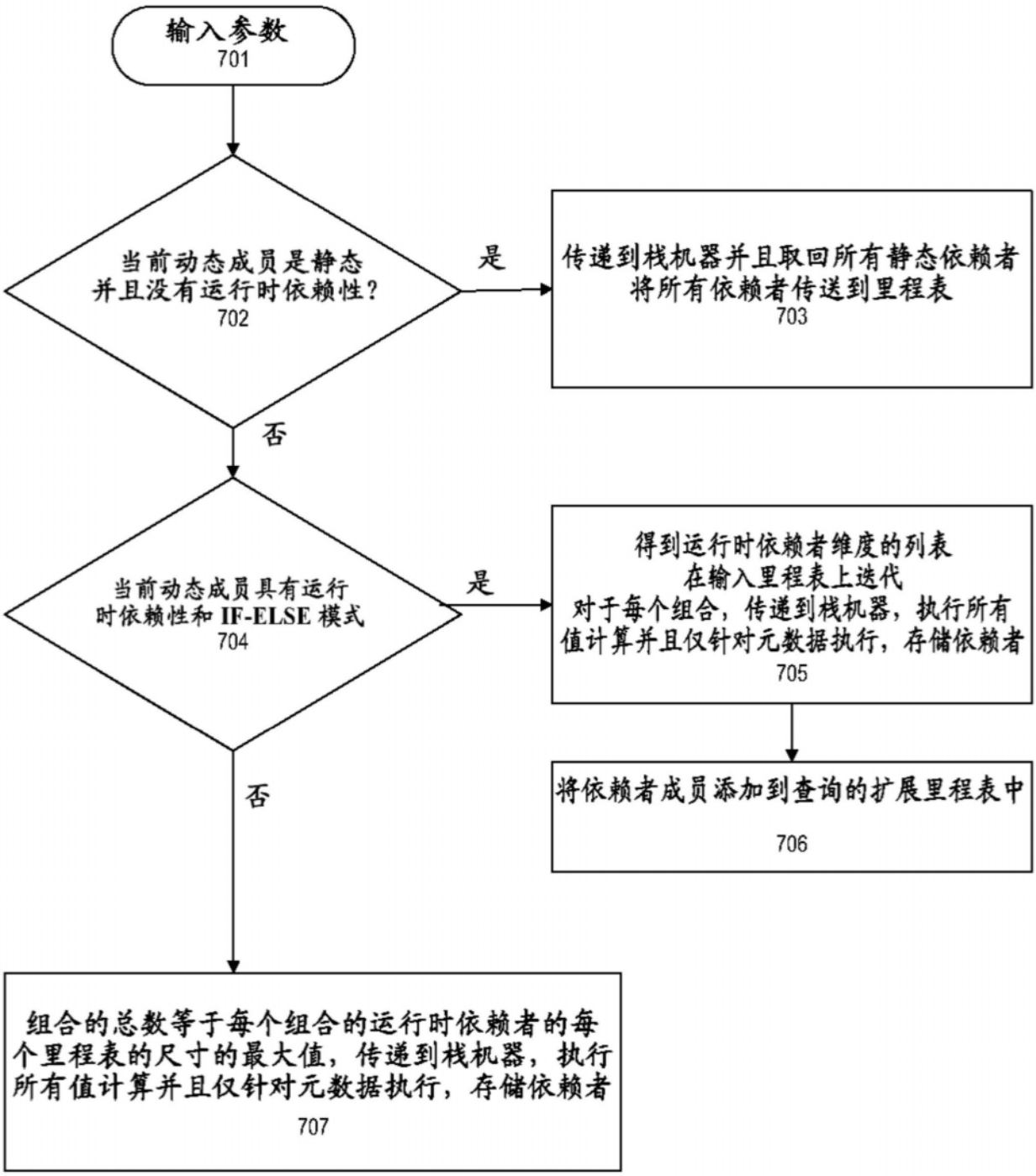

输入参数
701
当前动态成员是静态
并且没有运行时依赖性?
702
是
传递到栈机器并且取回所有静态依赖者
将所有依赖者传送到里程表
703
否
当前动态成员具有运行
时依赖性和 IF-ELSE 模式
704
是
得到运行时依赖者维度的列表
在输入里程表上迭代
对于每个组合，传递到栈机器，执行所有
值计算并且仅针对元数据执行，存储依赖者
705
将依赖者成员添加到查询的扩展里程表中
706
否
组合的总数等于每个组合的运行时依赖者的每
个里程表的尺寸的最大值，传递到栈机器，执行
所有值计算并且仅针对元数据执行，存储依赖者
707

图7

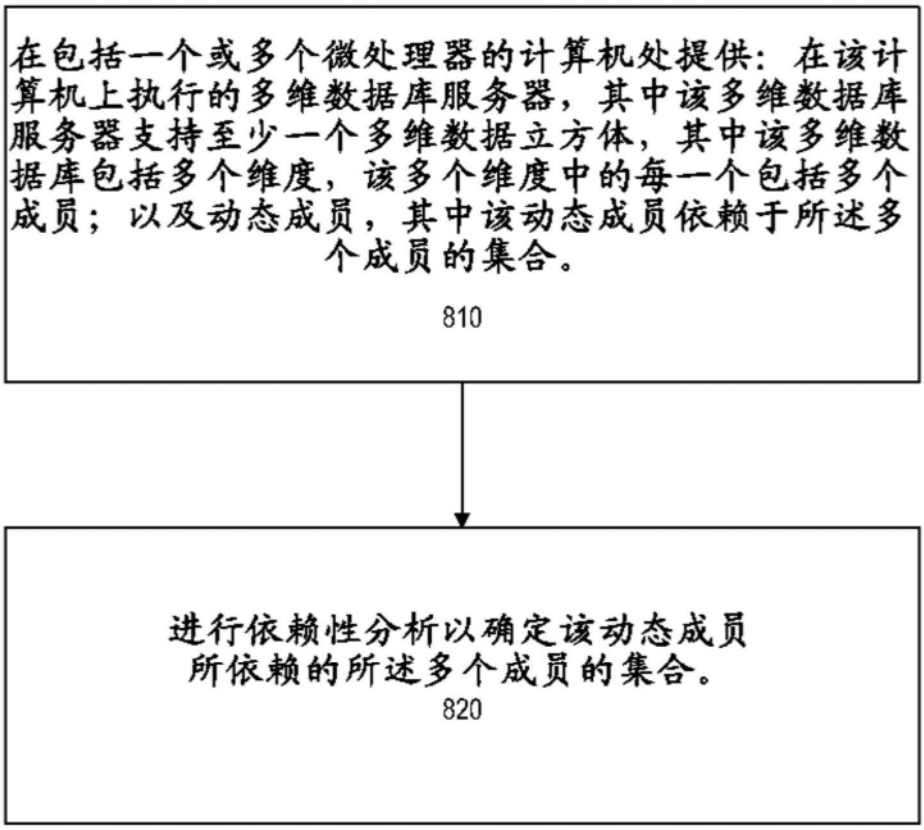

在包括一个或多个微处理器的计算机处提供：在该计算机上执行的多维数据库服务器，其中该多维数据库服务器支持至少一个多维数据立方体，其中该多维数据库包括多个维度，该多个维度中的每一个包括多个成员；以及动态成员，其中该动态成员依赖于所述多个成员的集合。
810
进行依赖性分析以确定该动态成员所依赖的所述多个成员的集合。
820

图8