

支持 What-if 分析的 OLAP 系统研究

王 珊^{1),2)} 肖艳芹^{1),2),3)} 张延松^{1),2),4)} 陈 红^{1),2)}

¹⁾(中国人民大学数据工程与知识工程教育部重点实验室 北京 100872)

²⁾(中国人民大学信息学院 北京 100872)

³⁾(河北大学计算中心 河北 保定 071002)

⁴⁾(哈尔滨金融高等专科学校计算机系 哈尔滨 150030)

摘 要 数据仓库和 OLAP 是决策支持系统的重要组成部分. What-if 分析是基于历史数据,对假设场景进行分析的重要手段,它可以为决策者提供重要的预测信息,是一种非常重要的决策支持分析过程. 文中分析了 what-if 分析的研究现状,从不同的角度对 what-if 分析进行了分类. 将其分为基于规则的 what-if 分析、基于 delta 表的差分数据存储与合并机制的 what-if 分析和基于 CUBE 增量维护方法的 what-if 分析三类,分析了各种实现方法的优点以及适用范围. 给出了基于内存数据库的 delta 表模式的 what-if 分析算法和一种新的 CUBE 维护算法. 最后对基于 what-if 分析的 OLAP 系统的未来研究方向进行了展望.

关键词 数据仓库; OLAP; what-if 分析; CUBE; delta 表

中图法分类号 TP311

Research on OLAP System Supporting What-if Analysis

WANG Shan^{1),2)} XIAO Yan-Qin^{1),2),3)} ZHANG Yan-Song^{1),2),4)} CHEN Hong^{1),2)}

¹⁾(Key Laboratory of Data Engineering and Knowledge Engineering of Ministry of Education, Renmin University of China, Beijing 100872)

²⁾(School of Information, Renmin University of China, Beijing 100872)

³⁾(Computer Center, Hebei University, Baoding, Hebei 071002)

⁴⁾(Department of Computer Science, Harbin Finance College, Harbin, Heilongjiang 150030)

Abstract Data Warehouse and On-Line Analytical Processing (OLAP) are the key components of Decision Support System (DSS). What-if analysis focuses on analysis on hypothetical scenarios based on historical data. It is an important type of DSS analytical processing procedure, and can provide important predicting information for DSS users. This paper firstly surveys major related work of what-if analysis and then briefly discusses the classifications from different angles. The main contribution of this paper is to present the global classification of what-if analysis. Three classical prototypes of what-if analysis based on rules, what-if analysis based on delta table and what-if analysis based on incremental CUBE maintaining are analyzed. Also the advantage and disadvantage of different type of what-if analysis and applying boundary are discussed. The authors' recent research work of delta table based on Main-Memory Database and a new CUBE maintain algorithm are discussed to discovery new algorithms on advanced Main-Memory Database architecture. Finally the future work of what-if analysis is summarized.

Keywords data warehouse; OLAP; what-if analysis; CUBE; delta table

收稿日期:2008-05-28. 本课题得到国家自然科学基金(60473069,60496325)、国际合作(HP Lab)项目(Large Scale Data Management)、北京市教委产学研合作项目(基于内存的联机分析处理系统)资助. 王 珊,女,1944 年生,教授,博士生导师,主要研究领域为高性能可扩展数据库、数据仓库与知识工程、数据库信息检索. 肖艳芹(通信作者),女,1974 年生,博士研究生,主要研究方向为内存数据库、内存 OLAP. 张延松,男,1973 年生,副教授,主要研究方向为内存数据库、内存 OLAP. 陈 红,女,1965 年生,教授,博士生导师,主要研究领域为数据仓库与数据挖掘、传感器数据管理.

1 引言

在联机分析(OLAP)和决策支持系统(DSS)中,基于星型结构或雪花型结构的多维数据分析是企业决策和宏观经济分析的重要方法。

基于假设场景的分析是企业决策时经常采用的方法,主要有两类应用场景:(1)在决策方案选择时对候选方案进行预测,即在进行决策方案选择时,需要结合历史的数据与经验对决策方案的未来执行情况与效果进行预测与评估,决策方案的未来执行情况可以采用假设的历史数据进行拟合;(2)对历史的多个候选决策方案进行评估,即在真实的历史数据上应用未采用的候选决策方案,并根据假设的方案进行量化分析。由于历史数据是真实可靠的,基于历史数据的假设分析结果可以更有效地评估候选决策方案,从而为企业决策积累宝贵的经验。

例 1. 假如在上一年 3 月增加了 30% 的国产原材料库存,减少了 20% 的进口原料库存,并随之调整相应产品型号的生产比例,将会对企业总利润产生什么样的影响?

例 2. 如果将企业总利润提高 10%,那么企业的产品结构需要如何进行调整?

例 1 是在基本历史数据的基础上进行假设更新,分析决策方案的采用对企业总利润的影响,从而可以量化地确定决策方案的最终效果,属于自底向上的分析。例 2 是对聚集的历史数据进行假设更新,进而分析该更新对底层基本数据的影响,属于自顶向下的分析。

What-if 分析^[1]是基于假设数据进行的分析,在决策支持系统领域(DSS),what-if 分析是一个重要的辅助功能^[2-3]并且较早就得到了广泛的应用。在 OLAP 分析工具中,大多数分析工具支持一定程度的 what-if 分析功能,但其功能还远远不足,支持 what-if 的 OLAP 分析还未成为成熟的技术,主要原因是 what-if 分析在数据存储结构、实现技术、系统功能和性能等方面有许多需要进一步探索研究的问题。

What-if 分析的应用需求和实现技术具有很大的差异性,目前比较典型的解决方案与工具主要分为两类:

(1)基于平面数据的 what-if 分析工具。其中比较有代表性的是基于 Excel sheet data 的 what-if 分析功能,Excel 自身和一些扩展工具支持基于 sheet

表数据的 what-if 分析,主要面向平面表的数据假设分析和问题假设求解,功能相对简单,直观易用。但难以对企业海量数据进行多维、多用户、多版本的 what-if 分析。

(2)基于数据仓库的 what-if 分析工具。受应用需求多样性与复杂性的影响,OLAP 分析没有通用的解决方案,只能提供有限的 what-if 分析功能。复杂的 what-if 分析需要结合企业业务数据的特点和商务处理规则的特点进行定制,其解决方案往往带有较强的个案特征,难以形成通用的数据分析处理标准。

What-if 分析既可以在数据库、数据仓库中的关系表上进行,也可以在 OLAP DB 中的多维数据上进行。图 1 说明了 what-if 的分析对象。本文中的 what-if 分析主要是指针对 OLAP DB 的 what-if 分析,即图 1 中虚线框内部分。

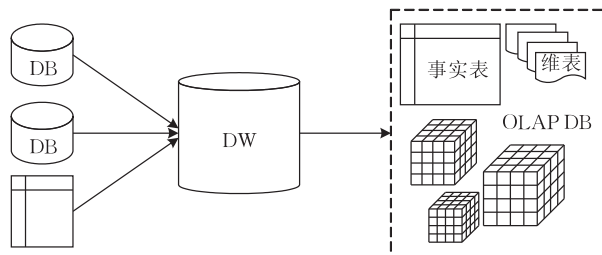


图 1 What-if 的分析对象

本文第 2 节介绍 what-if 分析的相关概念;第 3 节介绍 what-if 分析的分类以及具体的实现技术;第 4 节介绍现有的 what-if 分析工具及系统;第 5 节介绍我们关于 what-if 分析的研究工作;第 6 节对未来研究方向进行展望。

2 What-if 分析的相关概念

我们首先归纳并定义 what-if OLAP 中的相关概念。

定义 1. 基准表(base table)是 what-if 假设分析的基础数据对象,既可以是事实表也可以是某个物化视图(cuboid)。一个 n 个维 m 个度量值的 CUBE 中,事实表表示为 $\text{fact_table}(D_1, D_2, \dots, D_n, M_1, M_2, \dots, M_m)$, 则物化视图可以表示为 $\text{MView}(D_i, \dots, D_j, M_k, \dots, M_l)$, 其中, $i, j \in [1, n], k, l \in [1, m]$, M_k, \dots, M_l 代表聚集操作后的结果。

定义 2. Delta 表(delta table)。Delta 表存储与基准表相对应的基于假设场景的 what-if 假设数据。Delta 表与基准表类似,在基准表中增加若干字

段来存储 what-if 更新数据、版本等信息。

定义 3. What-if 更新. What-if 分析中基于假设的更新操作称为 what-if 更新. What-if 更新操作包括 insert, delete, update 操作. 一个 what-if 更新可包含多个 insert, delete, update 操作.

定义 4. What-if 更新版本. What-if 更新版本是假设更新的逻辑单位,它代表决策用户一个完整的假设分析所对应的假设更新操作/数据集合(查询重写模式中为操作集合,delta 表模式中为数据集合),逻辑上与企业宏观决策相对应,物理上产生一系列基于假设的更新操作/数据,从数据角度来看,一个 what-if 更新版本对应一个 what-if 分析的数据视图.

如图 2 所示,what-if 更新版本 V_2 是在版本 V_1 基础上的假设,我们称这样的 what-if 更新版本为级联更新版本(cascade version), V_3 更新版本直接依赖于基准表,与其它假设版本无关,我们称这样的 what-if 更新版本为独立更新版本(independent update version). 我们用符号“ \odot ”表示 what-if 更新的数据视图与当前 what-if 假设数据合并为新的 what-if 数据视图的操作,则级联更新可以表示为

$\text{what-if_view}_n = \text{what-if_view}_{n-1} \odot \text{delta_table}_n$,
what-if_view₀为基准表.

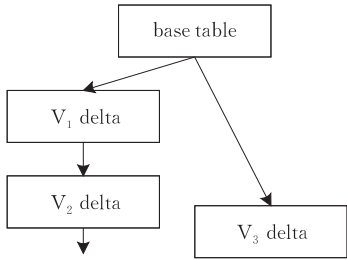


图 2 What-if 更新版本

独立更新视图表示为

$\text{what-if_view}_n = \text{base_table} \odot \text{delta_table}_n$.

我们用 $A \ltimes B$ 表示 A 数据依赖于 B ,以图 2 为例,what-if 数据视图与 delta 表之间的依赖关系可以表示为

$\text{delta_table}_1 \ltimes \text{base_table}$,
 $\text{delta_table}_2 \ltimes \text{what-if_view}_1$,
 $\text{delta_table}_3 \ltimes \text{base_table}$.

对于级联更新版本 V_1 和 V_2 ,我们称 delta_table_1 直接数据依赖于 base_table ,而 delta_table_2 间接数据依赖于 base_table , V_2 版本的 what-if 数据视图可以表示为

$\text{what-if_view}_2 =$
 $(\text{base_table} \odot \text{delta_table}_1) \odot \text{delta_table}_2$.

数据依赖 \ltimes 具有传递性(证明略),上述公式可以改写为

$\text{what-if_view}_2 =$
 $\text{base_table} \odot (\text{delta_table}_1 \odot \text{delta_table}_2)$.

根据此性质,在访问多级 what-if 更新版本的数据视图时,可以先合并各版本的 delta 表,最后再与事实表合并,这样可以大大减少 what-if 数据视图合并时的事实表扫描次数.

定义 5. What-if OLAP 分析. What-if OLAP 分析是 what-if 分析在 OLAP 系统中的扩展,假设更新的对象包括 OLAP DB 中的事实表、维表、物化视图,面向整个多维空间中的数据对象,what-if 分析是基于假设更新后的 CUBE 数据空间的多维查询,即通过标准的 OLAP 多维查询透明地访问基于假设更新后的 OLAP DB. 本文其后部分如无特殊说明,what-if 分析即表示 what-if OLAP 分析.

定义 6. What-if 数据视图. What-if 数据视图是在不同的 what-if 更新版本中对用户可见的、基于假设更新的数据集合. 我们用 what-if_view_i 来表示 what-if 数据视图,其中, $i \geq 0$. 当 $i = 0$ 时, what-if 数据视图为基准表. delta_table_j 表示版本 j 的 what-if 假设数据, $j \geq 1$. What-if 假设分析可以由不同管理层次的多个用户协同完成,各个用户自定义的 what-if 假设可能彼此独立,也可能具有相关性,如图 2 所示. 如某个维度上的 what-if 假设可以是另一维度上 what-if 假设的基础(如时间维度上的 what-if 假设作为区域维度上 what-if 假设的基础);某个层次上的 what-if 假设可以是其高层管理用户 what-if 假设分析的基础. 因此,基准表中的记录可能在多次 what-if 假设分析中被更新多次.

3 What-if 分析的分类

近三十年,有很多对 what-if 分析的研究^[4-8],这些文献基于不同的研究角度与应用需求,提出面向不同应用场景的解决方案或技术路线,但对 what-if 分析并没有完整的分类或体系结构的论述,对于 what-if 分析的应用背景、应用场景、应用模式等也没有系统的框架,大多数的相关研究是基于个案的解决方案.

本节通过归纳分类 what-if 分析的不同类别,提

出 what-if 分析的应用框架. 从不同的角度分析, what-if 分析可以分为不同的类别.

3.1 根据 what-if 更新的数据层次分类

3.1.1 基于 schema 假设更新的 what-if 分析

OLAP DB 的 schema 相对稳定, 基于 schema 的假设更新分为两种情况: 一是 OLAP DB 的 schema 根据应用的需求发生了改变, 为保持与以往 OLAP 分析的一致性而在新的 schema 上进行基于原始 schema 的 OLAP 分析; 另一种情况是 OLAP DB 的 schema 没有发生变化, 根据应用的需求进行假设分析, 如在 what-if 分析中将原 schema 中的一个维字段拆分为多个维字段或将多个维字段合并为一个维字段进行分析. 基于 schema 的假设分析需求相对较少, 需要在数据仓库层次上进行模式映射.

文献[9-10]提出了多版本数据仓库的管理方法, 它通过对整个数据仓库进行显式版本化来维护数据仓库的变化. 多版本数据同时并存, 使得用户可以跨版本进行查询, 并在不同版本之间进行比较. 而且, 它允许在真实数据版本的基础上创建和管理虚拟的业务场景, 以满足 what-if 分析的需求.

3.1.2 基于事实表假设更新的 what-if 分析

基于事实表假设更新的 what-if 分析是指当底层事实表发生改变时, 其上的 CUBE 将随之发生怎样的变化.

在 OLAP 系统中, 由于 CUBE 的计算是一项非常耗时的工作, 因此有许多研究工作是关于如何高效计算 CUBE 的^[11-14], 为了提高查询效率, 通常会物化某些 cuboid. 但是当基准事实表或维表进行更新时, 物化的 cuboid 不再有效, 此时需要维护该 CUBE. 通常采用的重新计算策略有两种: (1) 抛弃现有的物化视图, 从基准事实表开始完全重新计算一个新的 CUBE; (2) 在现有的物化视图的基础上, 增量重新计算出新的 CUBE. 第 1 种策略的优点是可以应用于所有视图的计算, 缺点是不能充分利用现有资源, 完全重新计算可能会耗费太多的系统资源. 第 2 种策略的优点是可以充分发挥现在有资源的作用, 缩短计算时间, 节省系统资源, 其缺点是应用受限, 因为有些聚集函数不可以进行分布式计算 (例如 median 函数), 因此不能通过原 CUBE 和增量 CUBE 求得. 对此类函数的更新计算可能比完全重新计算还要慢, 尤其是当数据的更新量比较大时.

文献[12]提出采用 delta 方法来维护 CUBE. 它将对基本事实表的修改存储在独立的 delta 表中, 需要插入的数据存储在 delta_insert 表中, 需要删除

的数据存储在 delta_delete 表中. 该算法分为两个阶段进行: propagation 阶段和 refresh 阶段. 首先, 根据 delta_insert 和 delta_delete 表中的数据计算出一个 delta CUBE(propagation 阶段), 计算其中一个 delta cuboid 的方法如下 (以 SUM 函数为例):

```
CREATE VIEW delta_abc(a, b, c, m, count) AS
SELECT a, b, c, sum(_m) as m, sum(count) as count
FROM ((SELECT a, b, c, m, as _m, 1 as count
FROM delta_insert)
UNION ALL
(SELECT a, b, c, -m as _m, -1 as count
FROM delta_delete))
GROUP BY a, b, c
```

其中, a, b, c 表示维属性, m 表示度量值属性.

在计算 delta cuboid 时, 需要计算每个分组中记录的个数, 为后面的 refresh 阶段做准备, 所以, 在 delta cuboid 视图中, 除了维属性和度量值属性之外, 还有一个 count(计数)属性. 另外, 因为 delta_delete 表中存储的是需要删除的数据, 所以将度量属性的值进行了取相反数的操作. 对于不同的聚集函数, 需要对度量属性的值采用不同的处理, 下表中给出一些针对不同聚集函数的度量属性的处理方式:

	delta_insert	delta_delete
COUNT(*)	1	-1
COUNT(expr)	Case when expr is null then 0 else 1	Case when expr is null then 0 else -1
SUM(expr)	expr	-expr
MIN(expr)	expr	expr
MAX(expr)	expr	expr

然后根据 delta CUBE 中各个 cuboid 的内容更新原 CUBE 中的相应的各个 cuboid, 得到新的 CUBE(refresh 阶段). 更新的基本思想是 (以 SUM 函数为例): 对于 delta cuboid ΔCubeV 中的每个元组 δt , 在相应的 cuboid CubeV 中查询与 δt 相应的元组 t , 即 δt 与 t 在所有维属性上的值都相同. 如果找不到满足条件的元组, 则将 δt 插到 CubeV 中; 如果找到元组 t , 且 $t.\text{count} + \delta t.\text{count} > 0$, 则更新元组 t , 使得 $t.m = t.m + \delta t.m$; 如果 $t.\text{count} + \delta t.\text{count} \leq 0$, 则删除元组 t .

文献[5]对上述方法进行了改进, 使得在计算 delta CUBE 时不再计算整个 CUBE, 而是只计算其中的 $\left(\begin{smallmatrix} n \\ \lceil n/2 \rceil \end{smallmatrix}\right)$ 个 delta cuboid, 然后利用这些 delta cuboid 去更新原 CUBE. 在更新过程中, 当根据某个 delta cuboid 更新原 CUBE 中相应的 cuboid 时, 同时更新原 CUBE 中其它相关的 cuboid, 例如: 更新

一个具有 a, b, c 3 个维的 CUBE 时, 若采用文献[12]的方法, 需要计算 $\Delta abc, \Delta ab, \Delta a, \Delta bc, \Delta b, \Delta ca, \Delta c$ 和 Δall 8 个视图; 采用文献[5]的方法只须计算 $\Delta abc, \Delta bc$ 和 Δca 3 个视图即可. 在使用 Δabc 更新 abc 视图时, 同时更新 ab, a, all 视图; 使用 Δbc 更新 bc 和 b 视图; 使用 Δca 更新 ca 和 c 视图. 该算法减少了生成 delta CUBE 的时间, 从而提高了 CUBE 的维护效率.

但是这两个更新算法仅适合于使用 SUM、COUNT 等分布式聚集函数生成的 CUBE, 而对于由其它类型的聚集函数生成的 CUBE 则不能进行维护. IBM DB2 UDB 中同样采用 delta 方法对 summary 表进行维护^[15], 它也仅限于维护那些使用 SUM 及 COUNT 作为聚集函数的 summary 表.

基于事实表的 what-if 分析可以支持 CUBE 中任意维度、任意层次的 what-if 分析.

3.1.3 基于维表假设更新的 what-if 分析

基于维表假设更新的 what-if 分析是指当维表发生改变时, 与该维表相关的 CUBE 将如何变化, 以保证查询的正确性.

在 OLAP 应用中, 数据存储在一组维表和事实表中, 通常假设事实表中的数据是数据仓库的动态反映, 会随时间发生变化, 而维表中的数据不易发生变化. 然而, 该假设经常与事实相违背^[16-18], 维表也是会随时间而变化的.

维表的更新可以分为两大类: 结构更新和实例更新. 结构更新是指修改维的模式; 实例更新是指修改维的实例. 结构更新包括以下操作:

(1) Generalize: 对于一个给定的维模式, 在一个已存在的维层次 l 上增加一个新的维层次 l_{new} , 并给出从 l 到 l_{new} 的映射关系, 使得可以从 l 到 l_{new} 进行 roll up 操作.

(2) Specialize: 在一个给定的维模式中, 在某个维层次下增加一个新的维层次 l_{new} , 并给出从 l_{new} 到原维层次的映射关系, 使得可以从 l_{new} 到原维层次进行 roll up 操作.

(3) Relate Levels: 在给定维模式中的两个独立的维层次 l_1, l_2 之间定义一个 roll up 函数, 使得可以从 l_1 到 l_2 进行 roll up 操作.

(4) Unrelate Levels: 删除维模式中两个关联维层次之间的关联关系.

(5) DelLevel: 删除维模式中已存在的某个维层次, 同时删除该维层次到其上层维层次的 roll up 函数.

实例更新包括:

(1) AddInstance(d, l, i, P): 在维模式 d 的某个维层次 l 中增加一个实例 P , 并说明实例 P 对应于 l_1 的上层中哪个实例, 例如, P_1 , 即对实例 P 进行 roll up 操作的结果是 P_1 .

(2) DelInstance(d, l, i): 删除维模式 d 的维层次 l 中的一个实例 i .

文献[19]通过扩展 MDX 语言实现了一个支持维更新的 OLAP server: TSOLAP.

当维模式更新时, 应更新基于该维模式的 CUBE, 以保持 CUBE 和维表的一致性. 不同的维模式更新方式对 CUBE 的影响不同, 因此需要针对不同的维模式更新方式进行不同的处理. 对维模式的 Relate、UnRelate 以及 Generalize 等更新操作不会对 CUBE 产生影响, 不需要更新 CUBE. 如果维模式的更新是 DelLevel(l, i), 且删除的是维模式中的最底层, 则需要重新生成 CUBE 的基本事实表, 新的基本事实表是在原基本事实表的基础上通过在维 l 上进行聚集运算得到的; 如果删除的是最底层之外的其它维层次, 则只需删除与该删除维层次相关的 cuboid 即可. 若维更新操作是 Specialize, 则 CUBE 的基本表则成为 CUBE 中的一个 cuboid, 应通过新的基本表重新生成 CUBE.

对于维实例的更新, 如果该实例更新所在的维层次不是最底层, 则任何 cuboid 都不需要更新, 此类维更新称为无关更新^[20]. 如果该维实例更新不是无关更新, 则针对需要更新的 cuboid, 根据该维实例更新计算一个 delta cuboid, 再使用该 delta cuboid, 刷新原 cuboid. Delta cuboid 计算规则如下:

给定一个维实例更新 u 以及一个 cuboid $f = \text{CubeView}(D, f_{base}, GB)$ (其中: D 表示一个维模式集合; f_{base} 表示基本事实表; GB 表示一个维层次集合, 每个维层次都来源于集合 D 中的维模式):

(1) 若 $u = \text{DelInstance}(d, l, i)$:

① 若 $GBottom_D \setminus GB = \{l\}$,

$$\Delta f = \pi_{GB \setminus \{l'\}}(l' = \delta^- d(l'), m = -m, \sigma_{l=i} f_{base},$$

其中, $l' = GB - GBottom_D$; m 表示基本事实表中的度量值属性. $\delta^- d(l')$ 表示所删除的维实例 i 在从维层次 l 到维层次 l' 进行 roll up 操作时的结果; $GBottom_D$ 表示 D 中维模式的最底层次的集合.

② 若 $GBottom_D \setminus GB \neq \{l\}$,

$\Delta f = \pi_{GB \setminus \{l'\}}(l' = \delta^- d(l'), m = -Ag(m), \sigma_{l=i} f_{base} \bowtie d_1 \bowtie \dots \bowtie d_n$, 其中, l' 表示 GB 中属于维模式 d 的维层次, d_1, \dots, d_n 表示集合 $GB \setminus \{d\} \setminus GBottom_D$ 中的维层次所属于

的维模式; Ag 表示一种聚集运算.

(2) 若 $u = \text{AddInstance}(d, l, i, P)$:

$$\Delta f = \pi_{GB \setminus \{l'\}}(l', l' = \delta^+ d(l), m = Ag(m) \sigma_{l=i} f_{\text{base}} \bowtie d_1 \bowtie \cdots \bowtie d_n,$$

其中, $\delta^+ d(l)$ 表示所增加的维实例 i 在从维层次 l 到维层次 l' 进行 roll up 操作时的结果.

上述规则说明了在维实例变化时, 如何根据基本事实表计算 delta cuboid.

不论是维数据更新, 还是事实数据更新, 都可能对已物化的 CUBE 产生影响, 因此, 在更新操作之后, 都需要对 CUBE 进行维护. 目前, CUBE 的增量维护策略多采用 delta 方法. 在计算 delta CUBE 时, 可以利用 CUBE 中 cuboid 之间的关系进行 delta cuboid 的计算, 即高层次的 delta cuboid 可以在低层次的 delta cuboid 的基础上计算得到, 而不必都通过基本事实表来计算. 由于 delta cuboid 中的记录远远少于基本事实表中的记录, 所以, 这种计算方法可提高系统的性能. 采用 delta 方法增量维护 CUBE 的优点是可以利用现有的物化视图, 尽量减少计算量. 但是, 当数据的更新量较大时, delta CUBE 的计算以及与原 CUBE 的合并都需要较大的计算量, 此时更适合对 CUBE 进行完全重新计算. 且 delta 方法仅适用于通过可自维护的函数生成的 CUBE, 在应用时具有很大的局限性. 当假设更新的数据量增大时, 增量 CUBE 的生成和合并也会降低算法的性能.

3.1.4 基于 cuboid 假设更新的 what-if 分析

OLAP 分析包括 ad-hoc 多维分析和基于支持物化视图机制的 CUBE 上的多维分析, 在存储空间许可的情况下, 基于物化 CUBE 的多维分析可以大大提高 OLAP 分析的性能和效率. OLAP DB 面向企业中不同层次的管理人员, 不同用户会根据自身的业务需求在不同的聚集层次上进行 what-if 分析, 需要在不同的层次上进行假设更新操作. CUBE 中的每一个物化视图是特定维、特定层次上的聚集数据, 当前物化视图的假设更新是直接对聚集值的更新, 可以从当前聚集层次上进行上卷操作. 因此对物化视图进行假设更新后, what-if 分析在从当前聚集层次向上所形成的子 CUBE 范围内有效. 如果需要在当前层次上进行下钻操作, 则需要指定数据从上向下的分配模式, 而这往往与复杂的业务逻辑有关. 若能够实现数据从上向下的分配, 则基于 cuboid 假设更新的 what-if 分析可实现上卷和下钻操作.

3.2 根据 what-if 更新的数据存储方法分类

3.2.1 基于 delta 表的 what-if 分析

基于 delta 表的 what-if 更新数据存储是指将 what-if 更新的数据存储在独立的 delta 表中, 通过将 delta 表与当前数据对象合并来完成查询处理.

解决基于假设的 what-if 查询的典型方法是采用差分文件(differential file)来存储对基本文件的更新数据, 多个版本的更新数据可以单独存储, 当访问某个版本的假设数据文件时, 将基本文件与对应的差分文件合并来提供一致的数据访问视图. Stonebraker 将差分文件的概念引入数据库系统, 提出了“Hypothetical Relations”(假设关系, HR)的概念^[21]. HR 的第一次实现是通过扩展 INGRES 的视图机制, 使其支持 UNION 和 DIFFERENCE 操作而实现的. 它将差分文件转换为差分表, 将对关系表(R)的更新操作(insert, delete, update)的结果保存在差分表 S(插入记录表)和 T(删除记录表)中, 其中 update 操作被分解为删除原记录和插入更新后的记录两个操作, 产生两个差分记录, 分别加入差分表 T 和 S 中, 当访问 what-if 数据视图时, 通过 $HR = R \text{ UNION } S \text{ DIFFERENCE } T$ 产生 what-if 数据视图. 在文献[22]的工作中, 通过时间戳机制解决了重插入(re-insertion)问题, 解决了基于集合运算的合并操作在处理删除某记录后重新插入该记录时所导致的记录丢失问题, 并且将两个独立的差分表合并为一个统一的差分表, 在该差分表中增加一个 type 列, 说明更新的类型: APPEND、REPLACE 和 DELETE, 分别表示增加一个新记录、更新原有的一个记录和删除一个记录.

文献[4]提出使用独立更新视图(Independently-Updated Views, IUV)方法表达 what-if 数据库中的假设场景. IUV 是指视图的更新数据独立于任何基本表进行存储, 且在该视图被检索时与该视图一起使用. 通常, 将没有更新的视图称为父视图(parent view), 更新过的视图成为 IUV. 使用 IUV 的 what-if 数据库中的查询处理过程如图 3 所示.

IUV 视图在 what-if 应用中提供了更大的灵活性, 它可以对聚集结果进行操作, 而不仅仅是详细记录. 也就是说, 若使用 IUV 表达 what-if 数据库中的假设场景, 那么不仅可以在基本表的基础上进行假设更新, 也可以在物化视图或计算结果表的基础上进行假设更新. 增加了 what-if 数据库的应用范围.

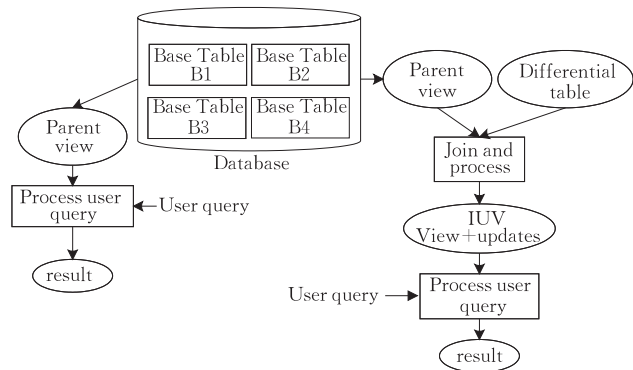


图 3 IUW 模式

这种方法通用性强,没有使用上的限制,能很好地支持多用户、多版本的 what-if 分析.但当差分数据量很大时存在着占用存储空间大、合并速度慢等缺点.

将假设更新数据存储在新的 delta 表或差分表中具有如下优点: (1) 独立存储不影响原来的基本表. 当不需要这些假设数据时,直接删除 delta 表即可. 如果将假设数据存储在新基本表中,那么当不需要这些假设数据时,需要对基本表进行复杂的 roll-back 操作. (2) 假设数据的独立存储有利于建立多版本假设场景. 当需要多个假设条件时,将不同假设条件产生的假设数据存储在不同的 delta 表中,互不影响. 所以对于多用户、多版本的 what-if 分析也能很好地支持. (3) 仅存储假设更新数据可节省存储空间. 由于假设场景与基本表差异较小,所以仅存储其差异数据,即假设更新数据,只占用较少的存储空间. 若存储由基本表和假设更新数据构成的一个完整假设场景数据,由于基本表很大,且是重复存储,必造成存储空间的极大浪费,尤其是在存在多版本假设场景时的情况下.

3.2.2 基于查询重写规则的 what-if 分析

基于查询重写规则的 what-if 分析是指存储 what-if 更新条件,而不是实际存储假设数据,在查询时根据查询重写规则将对假设数据的访问转化为对实际数据的查询.

对 what-if 查询的处理方式可以分为 Eager、Lazy 以及 Hybrid 三类. Eager 方式是指,将数据库的假设状态或相应的 delta 数据进行物化,并在这些物化的假设数据上进行 what-if 查询,例如 Heraclitus 数据库编程语言原型系统^[23]、Stonebraker 等提出的假设关系的实现方法以及前面提到的 CUBE 维护算法. Eager 方式对于在一种假设状态下进行多个假设查询具有一定的优越性. Lazy 方式是指,

将假设状态转换为等价的显式替代,并在 what-if 查询中应用这些替代,从而将 what-if 查询转换为纯粹的关系代数查询. 一个替代可表示为如下形式^[7]:

$$\mu = \{Q_1/S_1, \dots, Q_j/S_j\},$$

它表示用查询 Q_i 的值代替关系表 S_i .

例如,若有数据库的假设更新表示为

$$\mu = \{\sigma_{A>10}(R)/R, \sigma_{B<50}(S)/S\},$$

则在该假设更新上的查询 $Q = \pi_{A,B}(R \times S)$ 可表示为

$$Q = \pi_{A,B}(\sigma_{A>10}(R) \times \sigma_{B<50}(S)).$$

基于查询重写规则的 what-if 分析即是指 Lazy 的 what-if 查询处理方式.

Hybrid 方式是指上述两种方式的结合,即某些假设更新采用 Eager 方式,实际存储假设数据,而另一些假设更新则采用 Lazy 方式进行计算.

文献[7]提出了一种 Lazy 的 what-if 查询处理方法以及将 Lazy 方式与 Easy 方式进行融合的 Hybrid 方式. 首先, Lazy 方式是利用等价理论,将形如 “ Q when $\{U\}$ ” (其中, Q 是一个关系代数查询, U 是一个假设更新表达式,可包含一系列原子的更新操作,表示一种数据库的假设状态) 的假设查询,通过一组查询重写规则,转换为等价的非假设查询 Q' ,得到一个纯粹的关系代数查询,并执行 Q' . 其次,若某些假设更新多次出现在查询中,为提高查询性能,可采用 Eager 方式计算这些假设更新,而其它假设更新仍采用 Lazy 方式进行计算,即采用 Hybrid 方式计算 what-if 查询. 例如,若有如下查询:

$$Q = ((Q_1 \text{ when } \eta_1) \text{ when } \eta_3) - ((Q_2 \text{ when } \eta_3) \text{ when } \eta_2) + (Q_3 \text{ when } \eta_3).$$

由于假设更新 η_3 出现次数较多,可以采用 Eager 方式计算假设更新 η_3 ,采用 Lazy 方式计算假设更新 η_1 和 η_2 .

文献[6]也提出了一种使用查询重写方式在 OLAP 应用中进行假设查询的方法. 即将对假设数据的查询转换为对真实数据的查询. 如有数据表 ValueCTV:

C	T	V
John	MSFT	12540
Mike	CATP	4580
John	MOTO	6890
Gage	MSFT	15000

表中, C 表示客户; T 表示股票名称; V 表示股票价值. 假设 MSFT 公司的股票上涨 10%, 其它股票价值不变, 则针对该假设的假设数据表 ValueCTV' 为

C	T	V
John	MSFT	13794
Mike	CATP	4580
John	MOTO	6890
Gage	MSFT	16500

现在,查询客户 John 的所有股票的情况,则该查询可表达为

```
SELECT C, T, V
FROM ValueCTV
WHERE C='John'
```

因为假设数据表 ValueCTV'在系统中没有真正存储,所以必须把该查询转换为对真实数据的查询:

```
SELECT C, T, (V * 1.1) AS V
FROM ValueCTV
WHERE T='MSFT' AND C='John'
UNION
SELECT C, T, V
FROM ValueCTV
WHERE T!='MSFT' AND C='John'
```

查询重写方法的优点是只存储假设规则,不需要存储假设数据,对假设数据的操作全部通过查询重写的方式转换为对真实数据的操作,可节省大量的存储空间.但是它只能表示那些可以通过规则来表达的假设.如果所进行的假设无法通过规则来表达,例如插入某些虚拟的无规则数据,则这些假设数据必须实际存储在系统中,而且在这些假设数据基础上的查询无法转换成对原基础数据的查询,所以该方法的应用具有一定的局限性.另外,查询转换后形成的查询往往比原始的查询操作复杂,例如上例中的查询,如果假设数据已经存储在系统中,则在假设数据上的查询只须执行一次 SELECT 操作,而转换为在真实数据上的查询之后则需要执行两次 SELECT 操作,所以这种查询重写方式可以看作是用时间换空间的 what-if 查询方式.该方法的另一个缺点是当假设条件非常复杂时,查询重写将变得很复杂,且重写后的查询也会非常复杂,从而影响系统的查询效率.对于多用户、多版本的 what-if 分析也很难提供高效率的解决方案.

另外,在 OLAP 分析中,CUBE 计算是其主要操作.由于一个 CUBE 结构中包含多个 cuboid 视图,其计算非常耗费系统资源.若将 what-if OLAP 中的 CUBE 计算都通过查询重写转换为对基本事实表的操作,可能会带来两个问题:(1)查询重写的复杂程度;(2)大量增加对其基本事实表的操作.这两个问题都将带来严重的系统性能问题.

3.2.3 基于 Hybrid 混合模式的 what-if 分析

基于 delta 表的 what-if 更新与基于查询重写

的 what-if 更新在不同的应用模式与数据模式下具有不同的空间效率和执行时间效率,Hybrid 混合模式根据空间与执行时间效率进行权衡,根据 what-if 更新的特点动态选择假设数据的存储模式.文献[7]实现了 Hybrid 方式的 what-if 查询处理.

3.3 根据 what-if 分析的使用模式分类

(1)单用户单版本

每个用户独立进行 what-if 更新和基于假设数据的 what-if 分析,what-if 更新不能在多个用户之间共享,每个假设只与当前的 what-if 分析相关联,只能串行地执行多个 what-if 更新与 what-if 分析,不能在其它假设的基础上进行多版本 what-if 分析.

(2)单用户多版本

What-if 更新不能在多个用户之间共享,只对当前用户可见,用户可以进行多版本的 what-if 更新,支持多个并行的假设更新和 what-if 数据视图上的假设更新.多版本 what-if 更新支持阶段性的增量 what-if 分析功能.

(3)多用户多版本

What-if 更新是用户基于假设进行决策分析的基础,其中包含了大量的专业知识,在多用户间共享 what-if 更新具有重要的意义.另一方面,OLAP DB 的用户是多层次的,对应企业中不同层次的管理人员,在 what-if 分析时对 CUBE 访问的权限和兴趣数据具有一定的层次性,因此,假设分析也具有一定的层次性.企业级的假设决策分析需要多个层次、多个部分的管理用户协助完成,因此,协同进行 what-if 分析的用户间 what-if 更新的共享访问和多版本控制是实现协同决策分析的重要保证.多用户协同的 what-if 分析需要 what-if 更新共享访问机制、多用户权限管理机制、what-if 更新 workflow 控制机制等协同完成.

3.4 根据 what-if 分析的数据范围分类

(1)Global what-if 分析

what-if 分析对应的数据对象包括整个 CUBE 数据空间,可以根据任意的维、任意的层次进行基于假设的多维查询分析.Global what-if 分析要求假设更新的数据在任意 CUBE 物化视图对象中都能够有效访问.

(2)Local what-if 分析

what-if 更新数据具有局部性,只在某个 CUBE 子空间有效,上卷和下钻操作只在该局部子空间内有效,what-if 分析只能在部分 what-if 数据视图上进行.

3.5 根据 what-if 分析的应用特点分类

(1)自底向上(Bottom-up)的 what-if 分析

它是对 OLAP DB 的底层数据进行 what-if 更新,并通过 what-if 分析查询各个上层数据空间中的聚集结果.一般用于对具体的业务策略进行效果分析,基于业务策略的假设可以直接反映到底层数据中,可以支持 Global what-if 数据视图上的 what-if 分析.自底向上的 what-if 分析模式的特点是“先假设,再分析结果”,是一种结果驱动的 what-if 分析模式.

(2) 自顶向下(Top-down)的 what-if 分析

它是在 CUBE 数据空间的高层进行 what-if 更新,直接对高层聚集数据进行假设分析,并自上而下地分析高层决策对各底层业务部分的量化影响.由于高层聚集数据上的假设更新需要将假设的增量数据向下层数据对象传播,完整的 what-if 分析功能需要各个层次的管理用户之间协同才能完成,需要多用户、多版本的 what-if 分析机制的支持.自顶向下的 what-if 分析模式的特点是“先确定目标,再选择策略”,是一种目标驱动的 what-if 分析模式.

(3) 混合(hybrid)模式的 what-if 分析

从 CUBE 的某个中间聚集层次开始,分别向上层和向下层数据对象进行 what-if 更新和 what-if 分析.是自底向上模式和自顶向下模式的结合.

3.6 分类小结

如图 4 所示,我们从不同的角度对 what-if 分析进行了分类,在五个类别中,前三个类别由 what-if 更新模式决定,后两个类别由 what-if 分析模式决定.

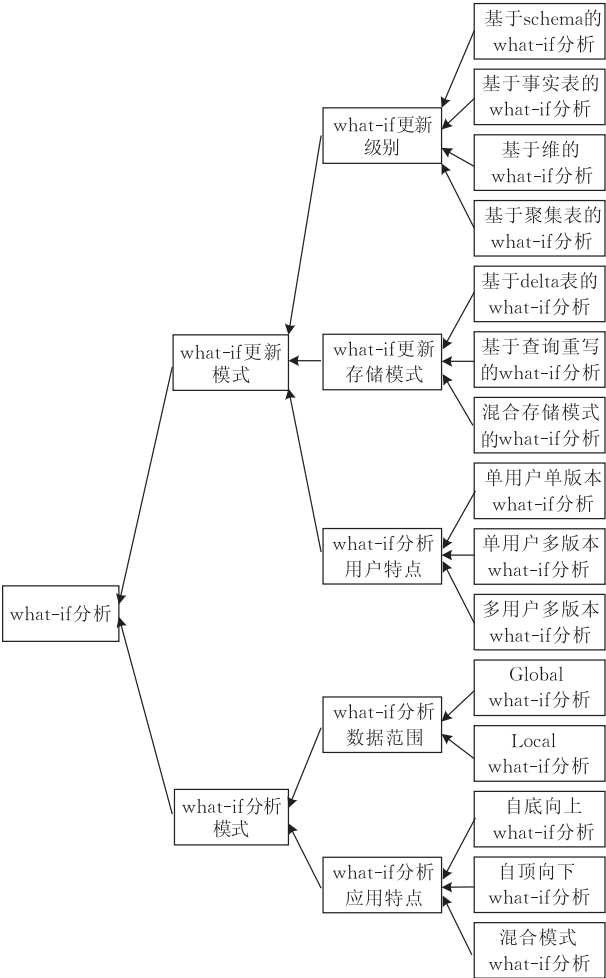


图 4 What-if 分析类型综合分类

表 1 What-if 分析分类

what-if 更新类型		what-if 分析类型以及典型实现方法				
		what-if 分析数据范围		what-if 分析应用特点		
		Global what-if 分析	Local what-if 分析	自底向上 what-if 分析	自顶向下 what-if 分析	混合模式 what-if 分析
what-if 更新级别	基于 schema 的 what-if 分析	MVDW ^[9-10]		MVDW ^[9-10]		
	基于事实表的 what-if 分析	ΔQ ^[5,12]		ΔQ ^[5,12]		
	基于维的 what-if 分析	TSOLAP ^[16,19]		TSOLAP ^[16,19]		
	基于聚集表的 what-if 分析	MDX_rewrite	MDX_rewrite	MDX_rewrite		
what-if 更新存储模式	基于 delta 表的 what-if 分析	DELTA_TABLE ^[21-22]	IUVs ^[4]	DELTA_TABLE ^[21-22] , IUVs ^[4]	DSS ^[2-3]	
	基于查询重写的 what-if 分析	SESAME ^[6]		SESAME ^[6]		
	混合存储模式的 what-if 分析	HQ Framework ^[7]		HQ Framework ^[7]		
what-if 更新用户模式	单用户单版本 what-if 分析	DELTA_TABLE ^[21] , SESAME ^[6] , ΔQ ^[5,12]	IUVs ^[4]	DELTA_TABLE ^[21] , SESAME ^[6] , ΔQ ^[5,12] , IUVs ^[4]		
	单用户多版本 what-if 分析	DELTA_TABLE ^[22]		DELTA_TABLE ^[22]		
	多用户多版本 what-if 分析	Oracle_patent ^[24]		Oracle_patent ^[24]		

为进一步分析现有 what-if 研究中的原型系统和实现技术,我们将“what-if 更新模式”和“what-if 分析模式”作为分类轴,以下的各级分类作为分类轴上的各级层次,则当前研究中比较典型的技术与原型系统可以用表 1 进行分类表示.从表 1 中可以看到,目前大部分技术都属于自底向上的分析类型,而自顶向下的分析方法则较少.这是由于自顶向下的分析涉及到数据从上到下的分配问题,它与具体应用的业务逻辑相关,不是单纯的算法能够解决的问题.

4 支持 what-if 分析的 OLAP 系统

目前,具有 what-if 分析功能的 OLAP 系统比较少.原型系统包括 SESAME^[6]和 TSOLAP^[19].市场上大多数商业数据仓库系统和 OLAP 工具,如 IBM DB2、Sybase Adaptive Server Enterprise、Ingres DecisionBase OLAP Server、NCR Teradata、Hyperion Essbase OLAP Server,都不能对数据仓库的结构更新进行管理,不支持 what-if 分析功能,不能查询多版本数据^[9-10].SAP Business Warehouse 仅能处理简单的维数据变化.Oracle 能够通过 SQL model 子句^[24,26-27](在 Oracle 10g Release 中可用)进行 what-if 分析处理,但其分析仅限于对平面关系表的处理.

电子表格,尤其是 Microsoft Excel,是重要的分析工具.电子表格提供了许多金融、统计、工程和数据方面的函数以及数据转换服务.它们提供了一个图形化的交互式用户界面,易于用户使用的计算模式,并为交互式的 what-if 分析方式,便于分析者进行动态的 what-if 分析.然而,作为一种计算工具,电子表格具有严重的缺陷,主要包括:缺乏良好定义的代数,其计算是隐藏的;缺乏可扩展性;缺乏数据共享能力.另一方面,OLAP 工具以及关系引擎等则具有很好的扩展性、良好的数据共享能力和良好定义的计算代数,然而这些计算引擎在用户交互、图形化界面以及计算语言的通用性等方面存在明显的不足.Oracle 针对关系引擎的上述不足,对 Oracle 10g 进行了分析处理功能的 SQL 扩展,即 SQL Spreadsheet 子句^[28-29],后更名为 SQL Model 子句,并开发了 QBX(Query By Excel)^[24]系统.QBX 系统的目的是将 Excel 电子表格作为关系数据的前端工具,充分利用 Excel 的图形化界面和交互能力.QBX 系统可以自动地把 Excel 计算转化为 SQL.分析者可

以从关系系统中导入数据,使用熟悉的 Excel 公式在这些数据上定义计算方式,将其转换为导入数据上的关系 SQL 视图,并进行存储,所以此时 Excel 计算是由关系系统完成的.

虽然通过 QBX 和 SQL Model 子句,Oracle 有了一定的 what-if 分析能力,且具有友好的用户界面,但其 what-if 分析仅通过 SQL Model 子句进行,对于事实表数据以及维表数据的更新都不能很好地处理,所以其能力受到了一定的限制,并不能完全满足 what-if 分析的要求.

上述基于 what-if 分析的 OLAP 系统的功能还都有待加强.研发功能完善的支持 what-if 分析的 OLAP 系统是我们今后需要努力的方向.

5 我们的研究工作

中国人民大学是国内最早研究和开发数据仓库和 OLAP 分析的单位之一,研制了 PARAWARE 数据仓库系统.目前我们在“内存 OLAP 服务器”研究项目中提出了基于内存数据库架构的 OLAP 服务器体系结构,并且把基于内存数据库的 what-if 分析作为系统的特征功能模块,研究在内存数据库服务器架构下基于假设的企业级 what-if 分析的性能和可行性,并探索建立一个完整的框架结构.

在第一阶段的工作中,我们研究了现有的 what-if 分析的实现技术,并在基于内存数据库的 delta 表模式的 what-if 分析和基于内存数据库的 ΔQ 算法改进以及基于多维查询语言 MDX 的查询重写策略等方面进行了深入的研究.

5.1 基于内存数据库的 delta 表模式的 what-if 分析

根据前面对 what-if 实现技术的研究,我们发现基于 delta 表的 what-if 分析具有最好的适应性,可以适应各种应用的需求,但 delta 表也存在着占用空间大、what-if 数据视图访问性能较低等缺点.对于企业级的海量数据仓库而言,what-if 更新的数据量远远小于原始数据,而由于企业决策的复杂性,what-if 更新操作的运行代价会较高,delta 表相当于物化的 what-if 更新数据,当假设更新后频繁地进行 OLAP 分析时,基于 delta 表的 what-if 更新策略在性能上优越于其它实现方案,delta 表的存储空间远远小于原始数据表,存储空间的开销也在可以接受的范围内.Delta 表不仅可作为 ad-hoc what-if 查询的基础数据,而且还可以作为基于物化视图机制的 ΔQ 增量维护算法的底层数据,具有较好的通

用性.

由于 delta 表的存储独立于原始数据,通过 delta 表生成 what-if 数据视图需要进行两表之间的合并运算.在磁盘数据库系统(DRDB)中,表之间的合并运算的执行代价很大.Join index 作为一种预连接的索引结构,可以提高表连接操作的执行效率,但在 DRDB 模式下,其效率受磁盘块访问策略的限制,很难获得令人满意的性能.在内存数据库系统(MMDB)中,系统数据常驻内存,数据操作性能大大优越于传统的磁盘数据库,对于 join index 而言,在内存数据库中可以使用内存指针代替原来的记录指针(数据块号+记录在块内的偏移地址),可以直接在记录之间建立基于内存地址访问的数据关联访问,delta 表合并性能大大提高.基于以上的分析,我们提出了基于内存数据库的支持 Delta tuple 直接访问的 join index 算法,内存 join index 的工作原理描述如下:

在 MM OLAP DB 中,我们修改 join index 的结构,存储每一条基准表记录和其对应的 what-if 更新的 Delta tuple 指针.假设基准表中的记录数为 n ,join index 定义如下:

$join_index = \{ (Bptr_i, Dptr_i) \mid 1 \leq i \leq n+1, n \in N \}$, 其中,如图 5 所示,join index 中的记录数为基准表的记录数+1,当 $1 \leq i \leq n$ 时, $Bptr_i$ 为指向基准表中第 i 条记录的指针.若基准表中第 i 条记录上没有 what-if 更新时, $Dptr_i$ 为空,否则 $Dptr_i$ 为指向 delta 表中 what-if 更新记录的指针;当 $i = n+1$ 时, $Bptr_i$ 为空,当前记录对应 what-if 更新中的插入记录入口指针. What-if 更新插入的记录在 join index 中只有一个入口指针,插入的多条记录在 delta 表中形成链表结构,并按版本顺序排列,同一版本的插入记录按插入时的自然顺序排列.

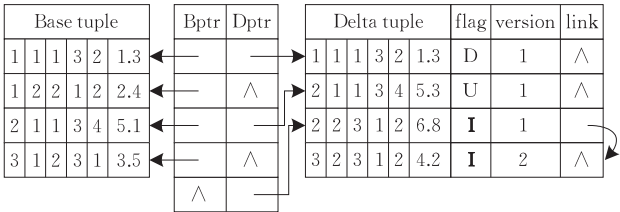


图 5 基于 join index 的 delta 表访问

Version 字段记录 what-if 更新的版本, what-if 更新版本的最小单位是 SQL 更新命令,也可以是一个 what-if 商业规则对应的一系列更新命令的集合.一个 what-if 更新版本中,基准表中的某个记录可能被更新多次,我们只记录当前更新版本中最终的更新结果,如在 what-if 查询中包含如下规则:“在×月

举行促销活动,降低×商品的价格 5%”和“在×月东部地区商品的物流成本增加了 2%”,可能会对事实表中的同一记录(2,1,1,3,4,5,1)进行两次更新,则在第一次 what-if 更新时增加一条 Delta tuple,第二次更新时直接在对应的 Delta tuple 进行再次更新,即在同一个更新版本内,对同一条记录的 what-if 更新只存储最终的更新结果.

我们设计了基于内存数据库的 Delta tuple 直接访问 join index 算法并通过实验验证了它对 what-if 数据视图的访问性能.如图 6 所示,我们设计了基于可写复本的 what-if 更新模式、基于 delta 表合并的 what-if 更新模式(包括索引连接模式)、基于内存 join index 的 what-if 更新模式在 what-if 数据视图访问性能上的测试,测试数据集为 FoodMart,事实表记录量分别为 8 万、80 万和 800 万, what-if 更新的粒度分别为原始事实表记录数的 5%,10%,15%和 20%.

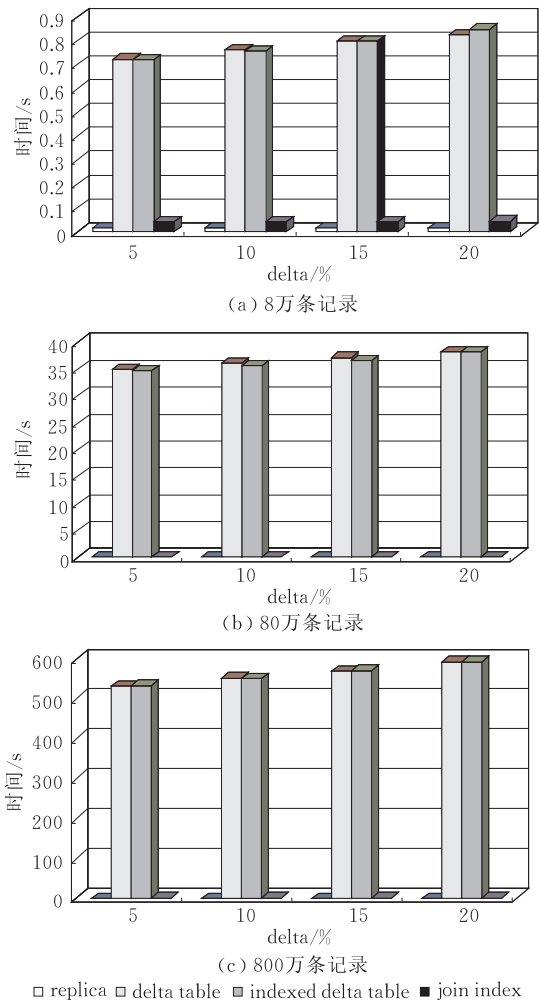


图 6 Writable replica, delta table, indexed delta table 与 join index 算法性能对比

从实验结果我们可以得到这样的结论：基于 join index 的 Delta tuple 访问算法充分利用内存指针的直接数据访问能力,建立了基表记录与 what-if 更新记录的直接联系,在访问 what-if 数据视图时的时间消耗和访问原始表时非常接近,具有非常好的 what-if 数据视图访问性能,远远优越于传统的 delta 表合并算法.

5.2 CUBE 维护算法改进研究

物化视图机制可以有效提高 OLAP 查询的性能,在 CUBE 更新时,为降低实体化 CUBE 的维护代价,最大限度地利用已有的实体化 CUBE,一般采用增量 CUBE 维护的方法,这样可以充分发挥现有资源的作用,缩短计算时间,节省系统资源. 其缺点是应用受限,因为有些聚集函数不可以进行分布式计算(例如,median 函数).

由于 MAX、MIN 函数在数据被删除时是不可自维护的,所以由 MAX、MIN 函数生成的 CUBE,在进行增量计算时比 SUM、COUNT 等函数生成的 CUBE 要复杂. 文献[12]提出了对 MAX、MIN 函数的 CUBE 进行增量计算的方法,即如果当前的最大值(最小值)被删除,则需要从基础事实表中重新计算新的最大值(最小值). 基础事实表中的数据量非常庞大,重新计算新的最大值(最小值)会花费大量的时间.

针对这种情况,我们提出了一种新的 MAX (MIN) CUBE 的增量计算算法 R-NLC(Refreshed by Next-Level Cuboid),能够:

- (1) 自底向上增量计算 CUBE;
- (2) 增量计算底层(level 0)cuboid 时,从基准事实表中计算新的最大值(最小值);
- (3) 增量计算第 n 层的 cuboid 时,使用相邻下一层,即第 $n-1$ 层 cuboid 的计算结果.

R-NLC 算法首先根据 delete_delta 表计算出 delta CUBE,该 delta CUBE 与基准 CUBE 采用同样的聚集函数、维属性和度量值属性,delta CUBE 中每个 cuboid 中 count 属性的值取 delete delta 表中具有该值的元组数的相反数. 然后使用 delta CUBE 中的每一个 delta cuboid 对基准 CUBE 中相应的 cuboid 进行增量计算.

我们通过实验对该算法进行了性能测试,实验结果表明 R-NLC 算法的执行时间明显少于完全从基准事实表中进行重算的方法.

5.3 基于多维查询语言 MDX 的查询改写策略研究

多维查询语言 MDX 是建立在 SQL 语言之上

的面向 CUBE 结构的 OLAP 查询语言,与 SQL 语言相比,其语法结构中以 CUBE 中的维、层次和成员作为直接操作的数据对象,可以更方便地处理基于 CUBE 多维结构的 OLAP 查询. 我们在应用层扩展 what-if 分析功能,通过 MDX 重写的方式表示 what-if 更新,通过 MDX 执行引擎实现动态的 what-if 查询. 与基于 SQL 的查询重写模式相比,基于 MDX 的查询重写具有更高的灵活性,在 what-if 更新的表示上也更加直观,能够有效降低系统开发的难度. 目前原型系统的研制工作正在进行之中.

6 未来的研究方向

What-if 分析是一种有效的多维数据分析与决策支持的辅助工具. 由于应用环境与应用场景存在着巨大的差异,what-if 分析的应用需求也千差万别,而且 what-if 分析的实现及功能受应用需求的影响,具有很大的不确定性.

What-if 分析在以下几个方面需要进行进一步的探索研究:

- (1) what-if 分析框架结构研究
- what-if 分析领域内的研究目前还不成熟,研究成果较少,也没有形成成熟的理论框架体系,很多研究都是基于特殊的需要而设计的,在方法上还不能有较好的通用性. 通过前面的分析,主要原因是:
- ① what-if 分析的需求不明确;
 - ② what-if 分析的解决方案与技术未形成完整的系统;
 - ③ 基于企业的假设分析与企业特殊的业务逻辑紧密结合,假设更新规则具有较强的业务依赖性,难以设计出通用的业务规则转换模块.

完整的体系结构框架、规范化理论和解决方案标准的制订可以更好地促进 what-if 分析研究工作的发展,更好地发挥企业 OLAP 的作用,为企业提供更加有效、灵活的决策评估与支持.

- (2) 多版本 what-if 更新机制研究
- 多版本 what-if 更新机制可以支持假设决策的阶段性与多用户特征. 将一个大的假设决策逻辑分解为多个部门级的业务逻辑会产生版本问题,如某些 what-if 更新是另一些 what-if 更新的基础,某些 what-if 更新数据依赖于其它版本的 what-if 更新等. what-if 更新版本可以是单用户环境下的多个假设更新的数据管理单元也可以是多用户环境下 what-if 更新的共享数据管理单元. 由于 what-if 更新版本之间数据依赖关系的不确定性,what-if 更

新版本形成一个网状的逻辑结构(某个 what-if 更新版本可能依赖于多个 what-if 更新版本),当 what-if 更新版本的祖先发生变化时,该 what-if 更新版本可能存在多条更新路径,基于 delta 表的 what-if 更新可以通过“Eager”方式在祖先 what-if 更新版本变化时重新生成 what-if 更新数据,基于查询重写的 what-if 更新可以通过“Lazy”方式在当前版本的更新数据被访问时生成相应的查询语句. 由于实际应用中业务逻辑的复杂性,what-if 更新版本的管理和维护也变得很复杂,如何保证 what-if 更新版本的高效使用还需要进一步的研究工作.

(3) 多用户协同 what-if 分析研究

在企业级 OLAP DB 上进行的 what-if 分析很难由单一的用户独立完成,基于不同策略的假设分析需要不同部门、不同层次用户的协同处理,而且处理过程也可能是分阶段进行的. 因此,在 what-if 分析中需要多用户、多版本的协同处理机制的支持.

在多用户协同的 what-if 分析中,需要处理两个重要的环节:① what-if 更新数据的共享访问;② 基于协同用户 what-if 更新版本的 what-if 更新以及多个相关 what-if 更新版本再次更新时的级联更新问题.

由于企业决策过程的复杂性与多变性,what-if 更新是一个频繁进行的试探性假设更新过程,通过 what-if 更新版本可以支持多阶段的假设更新,可以支持将一个大的假设决策业务逻辑拆解为多个相对独立的、可分工处理的部门级的假设业务逻辑,多版本的维护是一项重要而复杂的工作,对系统的可用性和灵活性具有重要的作用.

(4) what-if 分析的结果展示和动态分析

与普通的 OLAP 分析工具相比,由于 what-if 分析是基于假设的 OLAP 分析工具,因此在展示分析的结果时需要与原始数据的 OLAP 分析结果建立关联,不仅仅展示出 what-if 分析的结果还要同步展示出基于原始 OLAP 分析结果的差异,在实现中可以通过数据差值、图表等方式直观地向用户展示假设分析对结果的影响程度,从而为用户的决策提供更直观的依据.

普通的 OLAP 分析工具提供静态的多维分析结果,不能在分析结果上做进一步的假设分析. 在实际的应用环境中,假设分析是多层次的,在基于数据层次上 what-if 更新的基础上获取了 what-if 分析的结果后,还需要在最终的结果上进行小范围的、基于可变参数的 what-if 分析,从而将结果驱动的 what-

if 分析模式和目标驱动的 what-if 分析模式有效地结合起来.

从用户界面的角度看,对 what-if 分析的数据视图进行扩展,使之具有动态可假设更新能力,能够提供更灵活的假设分析功能,更好地适应假设策略和应用环境的多变性,提高系统的实用性.

(5) what-if 分析的性能优化

在 survey.com 公司(美国的著名国际市场调查机构)2006 年的 OLAP SURVEY 调查报告中显示,OLAP 产品最受用户关注的问题是“性能太差”. 虽然 OLAP 产品的性能也在逐年提高着,但其提高的速度远远低于用户的需求,从而使得性能问题成为制约 OLAP 应用的最大障碍. 而在传统 OLAP 应用基础上的 what-if 分析必然会进一步恶化 OLAP 应用的性能.

基于内存数据库的 OLAP 系统相比传统的基于磁盘数据库的 OLAP 系统具有更高的底层数据处理性能,会提高 OLAP 系统的整体性能. 由于内存数据库以内存为工作存储设备,基础数据和临时数据都存储在内存中,当采用物化 CUBE 机制时,会进一步造成内存容量的争用,虽然现代硬件技术有了很大的提高,但是内存相对于海量数据而言还不能提供充足的存储空间. 实验表明,当内存数据库系统的数据量超过内存容量时,内存数据库系统的性能会快速下降. 因此,如何有效地利用内存资源,结合内存数据库的高性能数据处理能力和物化视图对 OLAP 查询的支持能力,对于提高内存数据库 OLAP 应用的综合性能具有重要的作用.

在 what-if 分析中,CUBE 的维护代价比较大,是影响 what-if 分析性能的主要问题. 传统的 OLAP 应用和 what-if 分析中的算法是基于磁盘数据库系统而设计的,在基于内存数据处理的内存数据库 OLAP 系统中,可以根据内存数据处理的特点进一步优化 CUBE 的计算和维护算法,从而充分利用内存数据库系统的性能,改善 what-if 分析的性能.

参 考 文 献

- [1] Pendse N, Creeth R. The OLAP report. Business Intelligence Competency Centers, 1995
- [2] Philippakis A S. Structured what if analysis in DSS models// Proceedings of the 21st Annual Hawaii International Conference on System Sciences, Decision Support and Knowledge Based Systems Track. Kauai, HI, USA, 1988, 3: 366-370

- [3] Davis F D, Kottemann J E. What-if analysis and the illusion of control//Proceedings of the 24th Annual Hawaii International Conference on System Sciences. Kauai, HI, USA, 1991, 3: 452-460
- [4] Ramirez R G, Kulkarni U R, Moser K A. The cost of retrievals in what-if databases//Proceedings of the 24th Annual Hawaii International Conference on System Sciences. Kauai, HI, USA, 1991, 2: 136-145
- [5] Lee K Y, Kim M. Efficient incremental maintenance of data cubes//Proceedings of the 32nd International Conference on Very Large Data Bases. Seoul, Korea, 2006: 823-833
- [6] Balmin A, Papadimitriou T, Papakonstantinou Y. Hypothetical queries in an OLAP environment//Proceedings of the 26th International Conference on Very Large Data Bases. Cairo Egypt, 2000: 220-231
- [7] Timothy Griffin, Richard Hull. A framework for implementing hypothetical queries//Proceedings of the ACM SIGMOD International Conference on Management of Data. Tucson, Arizona, USA, 1997: 231-242
- [8] Zhang Y, Shen H. Applying hypothetical queries to E-commerce systems to support reservation and personal preferences//Proceedings of the 11th International Database Engineering and Applications Symposium. Washington, DC, USA, 2007: 46-53
- [9] Bebel B, Eder J, Koncilia C, Morzy T, Wrembel R. Creation and management of versions in multiversion data warehouse//Proceedings of the 2004 ACM Symposium on Applied Computing. Nicosia, Cyprus, 2004: 717-723
- [10] Morzy T, Wrembel R. On querying versions of multiversion data warehouse//Proceedings of the 7th ACM International Workshop on Data Warehousing and OLAP. Washington, DC, USA, 2004: 92-101
- [11] Gray J, Bosworth A, Layman A, Pirahesh H. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-total//Proceedings of the 12th IEEE International Conference on Data Engineering. New Orleans, Louisiana, 1996: 152-159
- [12] Mumick I S, Quass D, Mumick B S. Maintenance of data cubes and summary tables in a warehouse//Proceedings of the ACM SIGMOD International Conference on Management of Data. Tucson, Arizona, USA, 1997: 100-111
- [13] Beyer K, Ramakrishnan R. Bottom-up computation of sparse and iceberg CUBEs//Proceedings of the ACM SIGMOD International Conference on Management of Data. Philadelphia, Pennsylvania, USA, 1999: 359-370
- [14] Han J, Pei J, Dong G, Wang K. Efficient computation of iceberg cubes with complex measures//Proceedings of the ACM SIGMOD International Conference on Management of Data. Santa Barbara, CA, USA, 2001: 1-12
- [15] Lehner W, Sidle R, Pirahesh H, Cochrane R. Maintenance of cube automatic summary tables. ACM SIGMOD Record, 2000, 29(2): 512-513
- [16] Hurtado C A, Mendelzon A O, Vaisman A A. Maintaining data cubes under dimension updates//Proceedings of the 15th International Conference on Data Engineering. Sydney, Australia, 1999: 346-355
- [17] Hurtado C A, Mendelzon A O, Vaisman A A. Updating OLAP dimensions//Proceedings of the 2nd ACM International Workshop on Data Warehousing and OLAP. Kansas City, Missouri, USA, 1999: 60-66
- [18] Mendelzon A O, Vaisman A A. Temporal queries in OLAP//Proceedings of the 26th International Conference on Very Large Data Bases. Cairo Egypt, 2000: 242-253
- [19] Vaisman A A, Mendelzon A O, Ruaro W, Cymerman S G. Supporting dimension updates in an OLAP server//Proceedings of the 14th International Conference on Advanced Information Systems Engineering. Berlin Heidelberg: Springer-Verlag, 2002: 67-82
- [20] Gupta A, Mumick I S. Maintenance of materialized views: Problems, techniques and applications. IEEE Data Engineering Bulletin, 1995, 18(2): 3-18
- [21] Stonebraker M, Keller K. Embedding expert knowledge and hypothetical data bases into a data base system//Proceedings of the 1980 ACM SIGMOD Conference on Management of Data. Santa Monica, CA, 1980: 58-66
- [22] Woodfill J, Stonebraker M. An implementation of hypothetical relations//Proceedings of the 9th International Conference on Very Large Data Base. Florence, Italy, 1983: 157-166
- [23] Ghandeharizadeh S, Hull R, Jacobs D. Heraclitus: Elevating deltas to be first-class citizens in a database programming language. ACM Transactions on Database Systems, 1996, 21(3): 370-426
- [24] System and method for maintaining data for performing "What if" analysis. United States Patent, US 6766325B1, 2004
- [25] Witkowski A, Bellamkonda S, Bozkay T, Naimat A, Sheng L, Subramanian S, Waingold A. Query by excel//Proceedings of the 31st VLDB Conference. Trondheim, Norway, 2005: 1204-1215
- [26] The SQL Model Clause of Oracle Database 10g. An Oracle White Paper, August, 2003
- [27] Witkowski A, Bellamkonda S, Bozkay T, Folkert N, Gupta A, Haydu J, Sheng L, Subramanian S. Advanced SQL modeling in RDBMS. ACM Transactions on Database Systems, 2005, 30(1): 83-121
- [28] Witkowski A, Bellamkonda S, Bozkay T, Dorman G, Folkert N et al. Spreadsheets in RDBMS for OLAP//Proceedings of the ACM SIGMOD International Conference on Management of Data. San Diego, 2003: 52-63
- [29] Witkowski A, Bellamkonda S, Bozkay T, Folkert N, Gupta A, Sheng L, Subramanian S. Business modeling using SQL spreadsheets//Proceedings of the 29th VLDB Conference. Berlin, Germany, 2003: 1117-1120



WANG Shan, born in 1944, professor, Ph. D. supervisor. Her research interests include high performance scalable database, data warehouse and knowledge engineering, database information retrieval.

XIAO Yan-Qin, born in 1974, Ph. D. candidate. Her current research interests include main-memory database, OLAP and high performance databases.

ZHANG Yan-Song, born in 1973, associate professor. His current research interests include main-memory database, OLAP and high performance databases.

CHEN Hong, born in 1965, professor, Ph. D. supervisor. Her main research interests include database, data warehouse and sensor networks.

Background

What-if analysis focuses on analysis on hypothetical scenarios based on historical data, and it is an important type of DSS analytical processing procedure. In the traditional researches, different prototypes and solutions aimed at specific requirements and scenarios, no complete and global solutions can meet the general requirement of what-if analysis. In this paper, the authors present a framework of what-if analysis which is essential for the future research work, also develop several what-if analysis solutions to fill the gap in the framework. Improving the performance of what-if analysis by the

character of main memory OLAP is one of key problems of the study. The work is supported by the National Natural Science Foundation of China (grant Nos.60473069, 60496325), the joint research of HP Labs China and Information School of Renmin University, and the joint research of Beijing Municipal Commission of Education. The target of research is to establish a Main-Memory OLAP system based on what-if analysis which can provide high performance and powerful what-if analysis function.