一种基于 Dwarf 的快速有效增量更新算法*)

印 莹 赵宇海 张 斌

(东北大学信息科学与工程学院 沈阳 110004) (鞍山师范学院数学系 鞍山 114001)

摘 要 数据立方计算是代价非常大的操作,并且被广泛研究。受空间的限制,存储一个完全实例化的数据立方是不可行的。最近提出的一种语义压缩数据立方—Dwarf,通过消除前缀冗余和后缀冗余把一个完全实例化的数据立方压缩存储到一个很小的空间。然而,当数据源发生变化时,它的更新过程是很复杂的。本文通过研究 Dwarf 在更新过程中汇总结点的变化特性,提出了一种基于 Dwarf 的新的增量更新算法,既能完全实例化数据立方又不需要重新计算,大大提高了数据立方的更新效率。实验进一步证明了该算法的效率和有效性,尤其适合数据仓库中的高维数据集。

关键词 数据仓库,前缀冗余,Dwarf,增量更新

A Dwarf-based Fast and Efficient Incremental Update Algorithm

YIN Ying[1] ZHAO Yu-Hai[1,2] ZHANG Bin[1]

(College of Information Science & Engineering, Northeastern University, Shenyang 110004)[1]

(Department of Mathematics and Computer, Anshan Normal University, Anshan, 114001)[2]

Abstract Data cube computation is a well-known expensive operation and has been studied extensively. It is often not feasible to compute a complete data cube due to the huge storage requirement. Recently proposed Dwarf cube compressed the complete data cube into a dramatically condensed data structure through the elimination of prefix and suffix redundancy. However, as changes are made to the data sources, to update such a Dwarf cube in time is non-trivial. In this paper, we propose a new Dwarf-based optimized update algorithm on the basis of studying the characters of summarized node, which can materialize all of data cubes fully and requires no recalculation. The algorithm improves the updating performance of the data cube obviously. Furthermore, the effectiveness and efficiency of this algorithm have been shown by experimental results, especially for the high dimension data warehouses.

Keywords Data warehousing, Prefix redundancy, Dwarf, Incremental update

1 前言

数据立方预计算[1]能够便利 OLAP 操作。但随着元组数和维数的增加,数据立方计算[2]在关系数据库中是代价非常大的操作。语义压缩计算数据立方[3]是新兴的一种完全实例化的数据立方,它用压缩的方式保存数据立方,既保持了语义特性,又能完全实例化数据立方。同其它数据立方压缩方式[4]相比,基于语义的数据压缩方式具有以下的优点:(1)回答查询时不需要解压缩,因为语义数据立方将模式相同的数据压缩后存放在一起,查询时可以直接返回数据而不需要解压缩。(2)可以精确地回答查询。(3)语义数据立方计算给出了完整数据立方的信息,查询时不需要进一步进行聚集计算。(4)不同于那些压缩后的数据只能回答某种类型的查询的方法,语义数据压缩适宜所有的 OLAP 应用。Dwarf[5]数据立方,Condensed[6]数据立方,Quotient[7]数据立方以及改进的 Qc-Trees[8],都是近年来兴起的语义压缩数据立方。Yannis Sismanis 等人提出的 Dwarf 数据立方是一种基于语义压缩的数据立方,它利用数据的前缀和后缀语义信息对数据立方进行压缩,具有很高的压缩比。

然而,当数据源发生变化时,需要对数据立方体(采用各种压缩方体缩减后的数据立方体)中的数据进行相应的更新[9]。一种方法是将待更新数据与原来的 Dwarf 数据合在一起,通过重新计算生成新的 Dwarf 数据立方。这种方法虽然简单,但效率不高。因为它脱离了现有的 Dwarf 数据立方,计算量过大,既浪费时间又浪费空间,不适合现代商业的需求。通常采用另一种方法——增量维护的方式,即在已有的压缩结构上进行调整。这种增量维护方法无疑要优于前者。

本文就是要寻求一种快捷更新 Dwarf 数据立方的方法。通过研究 Dwarf 结构的特点,我们发现,更新的过程使 Dwarf 结构中汇总结点频繁发生变化。汇总结点的处理使更新的代价变得庞大,而汇总结点的处理可以转化成一些嵌套的三角形区域来处理。基于这个特点,我们设计并提出了一种基于 Dwarf 的快速有效的增量更新算法,实验表明了这个算法的有效性。

本文余下部分的组织结构如下:第 2 节介绍 Dwarf 结构分析,第 3 节介绍这种基于 Dwarf 的增量更新算法,第 4 节对实验结果进行理论分析和比较,最后总结全文。

2 Dwarf 结构分析

Dwarf 对于计算、存储来说是一种高度的压缩结构。Dwarf 数据立方将具有相同前缀和后缀的语义信息进行压缩。存储时消除了这两种类型的冗余信息,大大缩减了数据立方的存储空间,将一个完全实例化的数据立方缩减到一个非常密集的数据结构中。

*)国家自然科学基金资助项目(No. 60273079,No. 60473074,2004BA721A05)。印 莹 博士研究生,主要研究领域为数据仓库和数据挖掘。赵宇海 博士研究生,主要研究方向为生物信息与数据挖掘;张 斌 博士,教授,博士生导师,主要研究领域为数据库、Web 服务、网格技术。

表 1　事实表的实例

Store	Customer	Product	Price
S1	C2	P2	70
S1	C3	P1	40
S2	C1	P1	90
S2	C1	P2	50

图 1 是一个从表 1 的事实表计算得到的带有聚集函数 sum 的完整的 Dwarf 数据立方结构图。数据立方具有三个维：商店维(S)、顾客维(C)和产品维(P)。可以看出：Dwarf 数据结构是具有一个根结点的有向无环图，该图具 D 层，D 是数据方体的维数。第 D 层上的结点都具有[关键字：指针]的形式。在叶结点层，每个结点由形如〈关键字：聚集值〉的二元组集合构成。每个叶子结点都包含一个特殊值 ALL，存储该结点所有元素的聚集值。整个 Dwarf 的汇总路径〈ALL，ALL，ALL〉指向所有元组聚集值的和。

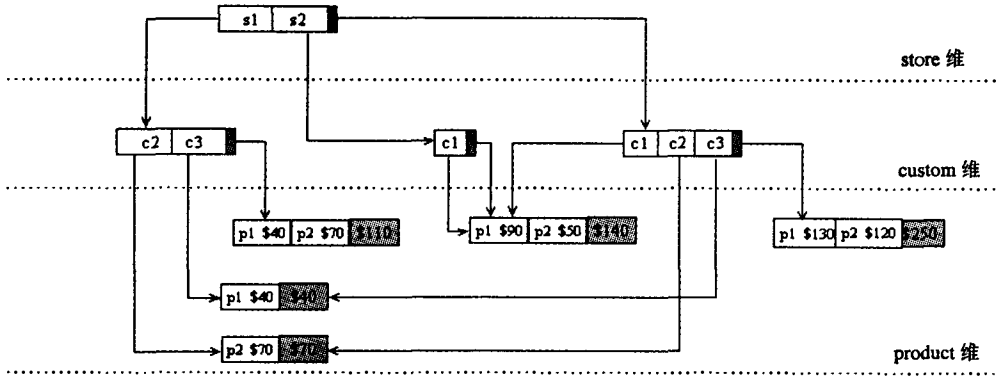


图 1　Dwarf 数据方体的示例

在由该事实表组成的一个数据立方中，S1 将出现 7 次。确切地说，S1 将出现在数据元素〈S1，C2，P2〉，〈S1，C3，P1〉，〈S1，C2，ALL〉，〈S1，C3，ALL〉，〈S1，ALL，P2〉，〈S1，ALL，P1〉，〈S1，ALL，ALL〉中。所谓前缀冗余是指某个维值或某几个维值的组合会同时出现在多个元组的前缀中。*Dwarf* 数据立方将这种类型的冗余提取出来，使得同样的前缀在数据立方中只存储一次。后缀冗余是指多个元组共享相同的后缀。例如：沿着三条不同的路经〈S2，C1，P2〉、〈S2，ALL，P2〉和〈ALL，C1，P2〉所得到的聚集值，全部是从表 1 中最后一条元组得到的，它们都指向了相同的后缀[P2，$ 50]。在图 1 中，凡是有多个指针指向的结点都是后缀合并存储的结点。*Dwarf* 数据立方将这种类型的冗余也提取出来，在数据立方中只存储一次，避免了冗余。去掉前缀和后缀冗余，不仅可以节约大量的存储空间，而且缩短了计算数据立方所需要的时间。

3　基于 Dwarf 的增量更新

当数据源发生变化的时候，需要对数据方体(采用各种压缩方体缩减后的数据方体)中的数据进行相应的更新。通过研究 Dwarf 结构的特点，我们发现，更新的过程中，Dwarf 结构的汇总结点频繁发生变化。汇总结点的处理使更新的代价变得庞大，而汇总结点的处理可以通过转化成对一些嵌套的三角形区域的处理来简化。基于这个特点，设计并提出了一种新的基于 Dwarf 的更新算法。

3.1　增量更新结构图

数据仓库中大多是高维数据集。无论是高维还是低维数据集，更新过程都是把新来的元组沿着根结点开始，循环地更新根结点下面的结点。通过被存储的结点中的 Cell 的关键字和要更新的元组的属性交叉检测比较，快速地扫描不需要更新的单元。如果扫描过程中没找到待更新的结点，进行结点扩展(利用溢出指针)，容纳新的属性值。依此循环，更新那些可能被一个或更多的要更新的元组影响的那些子 Dwarf。如果要更新的信息比已经存在的信息少得多，那么被扫描过的单元个数(Cells)要比需要更新的单元数(Cells)多。结束条件是新的结点被插入到 Dwarf 数据立方中，并且受该插入结点影响的所有汇总单元的处理也都结束。

图 2 所示为一个 4 维待更新 Dwarf 数据方体，最大的三角形区域是原始数据立方。从根结点开始扫描 Dwarf 数据立方的各结点，并与待更新元组进行自顶向下比较，直到找到叶子结点或找到该更新记录应在的位置。更新过程从叶子结点开始，自底向上插入要更新的元组，直到根结点。如果更新的结点是如图所示的 1、2、3、4 结点，那么它所影响的汇总结点将发生改变。这 4 个结点使最大的三角形区域对应的所有结点都发生变化，如图 2(a)所示。首先处理第一维变化所影响的 9、13、15 结点。处理结点 9 时，递归处理 10、11、12 结点，以此类推。然后处理第二维、第三维对应的三角形区域中的各个结点，如图 2(c)所示。随着处理的进行，三角形区域越来越小，即更新的范围越来越小。直到所有记录都被更新结束。

N 维增量更新类似 4 维更新，维数越多情形越复杂。基本算法都是遵照这种递归地处理三角形区域的方式来进行。

3.2　基于 Dwarf 的增量更新算法

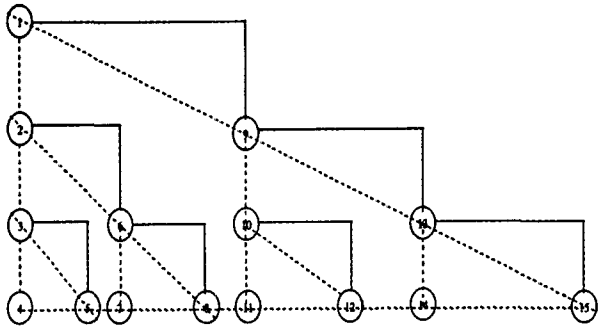
Dwarf 增量更新算法如算法 1 所示。

算法 1：Dwarf 更新算法

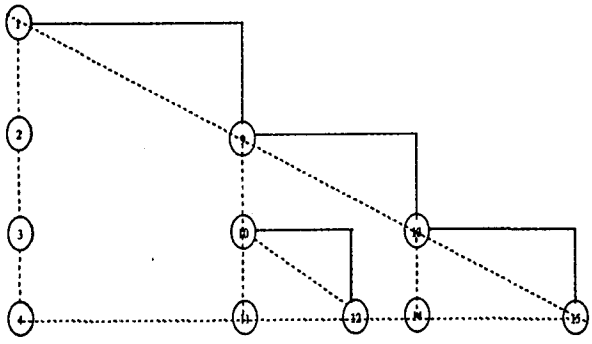
```
Algorithm1() //Dwarf 更新
输入：更新的数据集、已排序的文件
输出：更新后的 Dwarf 数据立方
1：读取第一条记录，从文件中获得维数 D 维
2：while 更新事实表非空 do
3：   取得要更新的第一条元组
4：    自底向上 do
5：    比较关键字与更新记录相同的关键字的结点(叶子)
6：     if 没有找到
7：     创建新的结点(叶子)记录添加到树上
8：     根据该记录影响的 ALL 创建一系列三角形
9：     调用 Algorithm2() //算 ALL 的函数
10：  end do
11：end while
12：end
```

构建过程仅仅要求把一次扫描已排序的事实表映射成字符型数据，读取事实表的第一个元组(步骤 1)，和对应 Dwarf 数据立方的每一维进行比较，相同则继续比较下一维，没有相

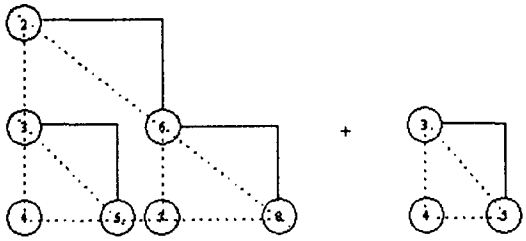
同则创建新的结点添加该单元，直到该元组被更新结束。创建过程见步骤3～7。递归地处理受该插入记录影响的汇总单元(步骤9)。具体结点的处理见算法2。事实表非空，则继续扫描事实表。随着扫描的继续，更新继续。事实表读取结束时，Dwarf数据立方增量更新结束。



(a) 更新过程中影响的一系列三角形区域



(b) 处理第一维变化影响的三角形区域



(c) 处理第二维和第三维变化影响的三角形区域

图2 基于Dwarfk的增量更新示意图

增量更新后，原数据方体结构发生如下变化：(1)仅仅是某一条元组的聚集值发生改变，使得其影响的聚集值也发生变化。(2)不仅仅聚集值变化，而且其它元组中找不到与该元组中的各个结点相同的单元。必须重新申请空间，创建新的单元，使得某些指针发生转移、某些结点发生分裂或合并。

算法2 更新处理三角形

```
Algorithm2 () //更新处理三角形
1:输入:三角形、更新元组
2:输出:当前处理更新完的三角形
3:begin
4:    for 当前处理的三角形
5:            自上而下 do
              // 对三角形中列的每一个结点详细计算
6:          处理三角形的每一列中影响的汇总单元
7:          end do
8:    end for
9:    for 递归的调用其他的三角形
10:             Algorithm2()
11:   end for
12:end
```

算法2是对汇总结点的具体处理过程，是一个递归算法。每一次调用是对各个汇总结点进行处理。处理过程由大三角形区域开始，首先处理最大的三角形区域，然后嵌套处理被其包含的小三角形区域。该算法是一个嵌套的过程，因而能保证处理大三角形区域时，它所包含的小三角形区域都已处理结束。具体过程如下：

(1)如果调用三角形是最小的三角形，就返回上一层的位置。

(2)否则，递归地调用被嵌套的三角形，并且检测在这个三角形的里面是否还有要处理的三角形。如果有要处理的三角形，递归进行调用。

(3)在每个被处理的三角形区域内部，是否存在要结合的单元。如果存在，而且是叶子结点的汇总单元，返回该叶子结点的聚集值。如果是非叶子结点的汇总单元，则对输入的节点调用后缀结合得到的一系列返回单元。

3.3 更新后的聚簇维护

增量更新后的Dwarf数据立方，同样存在潜在的问题：沿着Dwarf结构往下走(即降低结构)的时候，可以从不同的路径到达后缀结点。一旦到达了这个后缀结点，就必须检查这个后缀的路径是否仍然有效，因为新插入的一个或者多个结点可能使后缀的指针变得无效。大量记录的读入会破坏原来已经后缀结合的汇总结点，这主要是由那些溢出结点和被创建的溢出指针引起的。频繁地增量更新，会逐渐恶化原始的Dwarf数据立方结构。这种情况下，由于指针发生了变化，聚簇特性被破坏，查询效率也将大大降低。所以，需要对视图重新进行调整，以维持其聚簇特性。因为Dwarf的目标是为数据仓库应用，即在预定的周期间隔内进行更新。为了能快速地响应查询，必须保持它的聚簇特性。我们设计的聚簇维护策略，是设计一个小程序，随时可由用户启动。这个小程序扫描一遍更新后的Dwarf数据立方，并利用数据的反复移动，重新存储在磁盘上，即把相同的视图放在一个页面或连续的叶面上，所以这个小程序能重新组织Dwarf数据立方，把它转换成一个新的文件，并用聚簇存储起来，使更新过的Dwarf数据立方仍然具有聚簇特性，提高了查询响应时间。

4 实验测试与分析

本节是应用一系列数据集对更新算法做测试并得出一系列测试结果。实验环境是：Windows2000操作系统，384MB主存，2.0GHz CPU。数据集采用方法和参考文[5]中的方法是相同的，采用的是现有的人工数据集。

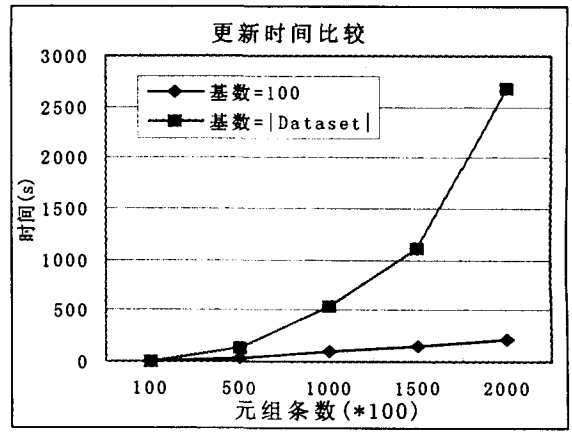


图3 元组数不同更新时间比较

(下转第129页)

```
{ 检查IP包的第51位;
  If(IP包的第51位==0)
    {利用非对称的解密密钥,启动解密算法对IP包得数据段进行解密;
    让解密后的IP包,即是对称密钥及算法通过防火墙II,转发到内部网络的目的站点,并对以前的对称密钥及算法进行更新;
    }
  Else
  {利用该时间段的对称密钥,启动相应的解密算法对IP包的数据段进行解密;
  让解密后的IP包通过防火墙II,转发到内部网络上的目的站点;}
  }
}
```

4.4 传输过程中的加密法

为了进一步提高数据安全性,可以在传输的过程中选择不同的加密方法,这里是对前面的不同内容采取不同的加密方法。先采用端到端加密法,在进入外部网络之前先加密,然后在外部网络中用链路加密法,即在每个节点上将已经在端点加密的密文再加密一次。这样可以防止中间节点的数据是明文。也避免了数据分组头是明文,易受到通信量分析的攻击。

当然这样设计的网络通信提高了数据安全性,但同时,网络资源代价也相当高,硬件设施和软件设施要求高。因此并不能让所有用户的网络通信都这样设置,只适用于对网络数据安全要求相当高的机构使用。也就是那些获得较高权限的用户能采用这种传输机制,没有获得权限的用户就可以直接传送数据。

结束语 本文介绍了防火墙技术和数据加密技术在网络数据库安全中的应用,以及相结合在数据传输过程中的应用的策略。这套方案不仅使内部网络的安全性得到保障,而且还保证了在开放网络中传送敏感数据的安全性,使网络安全得以增强并且在一个更大的范围得以实现。本文提出的防火墙技术和数据加密技术的结合构想,希望能使网络数据库安全性的提高起到抛砖引玉的作用。

参 考 文 献

1 Ziegler R L. Linux 防火墙[M]. 北京:人民邮电出版社,2000
2 罗春荣. 国外网络数据库:当前特点与发展趋势[J]. 中国图书馆学报,2003,3:44~47
3 陈小波. 网络信息安全与防火墙技术[J]. 现代情报,2003,7(7):48~49
4 肖军模,刘军,等. 网络信息安全[M]. 北京:机械工业出版社,2003
5 王睿,林青波. 网络安全与防火墙技术[M]. 北京:清华大学出版社,2000
6 Reed K D. 网络技术[M]. 北京:电子工业出版社,2002
7 袁家政. 计算机网络安全与应用技术[M]. 北京:清华大学出版社,2002
8 李广儒,邹云松. 防火墙技术的几个发展趋势[J]. 河南师范大学学报,2003(2):36~39
9 William T. 防火墙原理与实施[M]. 北京:电子工业出版社,2002
10 雷倩睿,李鹏文. 网络安全防御中数据加密技术的研究[J]. 信息技术,2003(1):6~9
11 孟艳红,秦维佳. 两种密码体制加密技术的研究对比[J]. 沈阳工业大学学报,2003(10):430~432
12 胡予濮. 对称密码学[M]. 北京:机械工业出版社,2002
13 Ford J L. 个人防火墙[M]. 北京:人民邮电出版社,2002
14 Andress M. 计算机安全原理[M]. 北京:机械工业出版社,2001

(上接第 90 页)

该测试集的聚集函数可以选择 SUM、MIN、MAX 进行处理。为了测试方便,所有的测试集只用一个聚集函数做 SUM 处理。图 3 和图 4 分别显示更新时间随着基数不同的变化曲线图。本文基数定义为每个维属性的不同的取值个数。Dwarf 数据集的特点是对基数的不同有明显的效果。

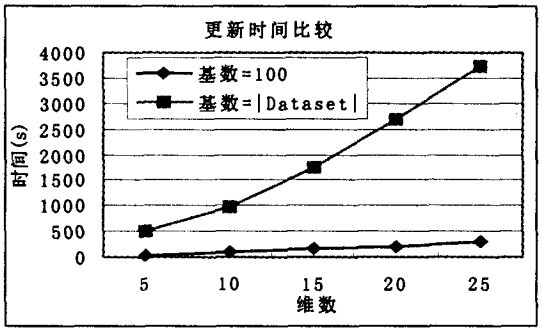


图 4 维数不同更新时间比较

上图分别是对维基数设为 100 的数据集和维基数等于元组数(Dataset)的数据集进行测试得到的更新时间比较图。图 3 是在相同的数据集(200000 条)大小下,更新的数据集分别采用 10000 条、50000 条、100000 条、150000 条、200000 条元组时的算法性能进行测试比较。图 4 是在相同的数据集(200000 条)大小下,在维数采用 5 维、10 维、15 维、20 维、25 维的情况下对 200000 条元组大小的数据集进行更新的性能测试比较。

从实验结果看出:不同的基数对更新性能有很大的影响。基数为 100 的数据集,其响应时间曲线倾斜度无论在数据集变化还是在维数变化的情况下都非常小,而基数等于元组数的数据集的响应时间曲线倾斜度随着维数的增加和元组数的增加急剧上升。从实验结果看出,前缀多的数据集更新性能很好,所以 Dwarf 更新算法在更新性能上对前缀多的数据集有明显的优势,也适用于数据仓库中的高维数据集。

结论 本文主要讨论了 Dwarf 数据立方的增量维护策略,研究并提出了一种新的基于 Dwarf 的增量更新算法。针对更新后聚簇特性被破坏的情况,利用聚簇整理,重新恢复 Dwarf 数据立方的聚簇特性。最后,通过实验给出了在维数不同和元组数不同时,对基数不同的数据集做测试比较的实验结果。实验结果进一步证明了该算法对前缀多的数据集更新性能很好,适用于数据仓库中的高维数据集。

参 考 文 献

1 Agarwal S, Agrawal R, Deshpande P M, et al. On the computation of multidimensional aggregates. In: Proc. of the 22nd VLDB Conf, 1996. 506~521
2 王珊,等. 数据仓库技术和联机分析处理[M]. 北京:科学出版社,1998
3 Li J Z, Rotem D. Aggregation algorithms for Very Large Compressed Data Warehouses. In: Proc. of VLDB Conf, 1999. 51~662.
4 Vitter J S, Wang M, Iyer B. Data cube approximation and histograms via wavelets. In: Proc. of Seventh Intl. Conf. on Information and Knowledge Management, 1998. 96~104
5 Sismanis Y, Roussopoulos N, Deligianannakis A. Dwarf: Shrinking the petacube. In: Proc. ACM SIGMOD, 2002. 464~475
6 Wang W, Lu H Jun, Feng J Lin, et al. Condensed cube: An effective Approach to Reducing Data Cube Size. ICDE, 2002
7 Lakshmanan L V S, Pei J, Han Jia Wei. Quotient cube: How to summarize the semantics of a data cube. In: Proc of VLDB Conf, 2002. 778~789
8 Lakshmanan L V S, Pei J, Zhao Y. QCTrees: An Efficient Summary Structure for Semantic OLAP. In: Proc. ACM SIGMOD 2003
9 Roussopoulos N. Cubetree: Organization of and Bulk Incremental Updates on the Data Cube. In: Proc. ACM SIGMOD, 1997. 88~99