数据立方体聚集范围查询分块方法研究

师智斌　黄厚宽
（北京交通大学计算机与信息技术学院　北京 100044）

摘　要　范围查询是数据立方体数据分析的有效工具，预计算技术通过预先计算并存储范围查询的结果，可以实现快速的用户响应。近年来研究人员对基于 MOLAP 的预计算技术的研究主要以 prefix sum 及分块技术为基础。本文对预计算技术的分块方法进行研究，分析了现有分块技术的方法和性能，并提出了两种新的分块方法：嵌套分块和基于前缀区域边界的分块。本文对这两种分块的方法和特点做了阐述，研究表明这两种方法为分块技术提出了新的思路，是对现有分块方案的有力补充。

关键词　数据立方体，范围查询，划分

The Research on the Partition for Aggregation Range Queries in Data Cube

SHI Zhi-Bin　HUANG Hou-Kuan
(School of Computer and IT, Beijing Jiaotong University, Beijing 100044)

Abstract　A range sum query is one of effective tools to analyze data in data cubes. Pre-computing can speed response times of range query through computing and storing the query result on-the-fly. The researches on pre-computing based on MOLAP are mostly based on technologies of prefix sum and partition recently. This paper works on partition scheme and analyzes the methods and capabilities of current technologies of partition. Two new methods of partition are put forward in this paper. They are nesting partition and partition based on the border of prefix region respectively. This paper discusses constitution and characteristic of two methods and it shows that they bring forward the new means to partition in data cube and supplement the current methods of partition.

Keywords　Data cube, Range query, Partition

数据立方体(data cube)[1]是应用于数据仓库和联机分析处理(OLAP)的多维数据模型，范围查询是数据立方体有效的分析工具，聚集的范围查询选择数据立方体中超立方体区域，计算并返回这个区域的聚集值。由于数据仓库包含的海量数据和在其上进行的查询的复杂性导致范围查询的响应时间过长，不能适应即时交互的决策需求。因此，如何有效地组织、存储海量数据，提供高效的范围查询操作，一直是数据仓库领域的热点问题。

预计算技术是提高数据立方体范围查询响应速度的一种方法，将数据预先进行计算并将结果存储起来，可以大幅提高对正交区域范围查询的响应时间。近年来，研究者在基于 MOLAP(Multidimensional OLAP)数据立方体预计算技术方面开展了许多工作。文[2]首先提出了 Prefix Sum(PS)的方法。在此基础上，文[3]和[4]分别提出了 Relative Prefix Sum (RPS)和 Double Relative Prefix Sum(DRPS)方法，以改善更新代价。文[5]和[6]提出了范围查询的层次结构 Hierarchical Cubes(HC)、Hierarchical Data Cube(HDC)方法。文[7]采用新的思路，在原 PS 的基础上，对更新数据建立 R 树索引，查询在 PS 表和 R 树间同时进行，更新只对 R 树操作。文[8]提出了 Dynamic data cube(DDC)，能同时保证查询和更新与数据立方体的维域成次线性关系，同时占有空间较少。文[9]在 RPS 和 DDC 的基础上进一步改进，提出了 Space-Efficient Relative Prefix Sum(SRPS)和 Space-Efficient Dynamic Data Cube(SDDC)结构，在不占用额外空间的情况下，使查询和更新代价分别达到 $n^{d/2}$ 和 $\log^d n$。文[10]结合了 SRPS 和 SDDC 的技术，利用递归的存储技术获得快速的范围查询响应。

上述预计算技术的研究是以文[2]提出的 Prefix Sum 为

师智斌　副教授，博士生，研究方向：数据仓库、数据挖掘；**黄厚宽**　教授，博士生导，CCF 高级会员，主要研究领域为人工智能、机器学习。

5 Bergamo P, Arco P. Security of public key cryptosystems based on Chebyshev polynomials [J]. IEEE Trans Circuits Syst I, 2005, 52: 1382~1393
6 Ruanjan B. Novel public key encryption technique based on multiple chaotic systems [J]. Phys Rev Lett, 2005, 26: 098702
7 Wang K, Pei W, Zhou L, et al. Security of public key encryption technique based on multiple chaotic system [J]. Phys Lett A, 2006,360:259~262
8 Zhang LH. Cryptanalysis of the public key encryption based on multiple chaotic systems [J]. Chaos, Solitons&Fractions(in press)
9 Kocarev L, Sterjev M. Public key encryption scheme with chaos [J]. Chaos, 2004, 14: 1078~1081
10 王大虎，魏学业，柳艳红. Chebyshev 多项式的公钥加密和身份认证方案的研究[J]. 北京交通大学学报，2005,29(5):40~46
11 刘亮，刘云，宁红宙. 公钥体系中 Chebyshev 多项式的改进[J]. 北京交通大学学报，2005,29(5):40~46
12 Blake I F, Seroussi G, Smart N P. Elliptic Curve in Cryptography [M]. Cambrige University Press, 2002. 185
13 Dai W. Speed comparison of popular crypto algorithms [DB/OL]. http://www.eskimo.com/~weidai/benchmarks.html
14 石熙，廖晓峰. 基于环面自同构的公钥加密算法[J]. 重庆大学学报(自然科学版)，2006,29(3): 62~64

基础，后继的方法，除文[8]外，都采用分块技术改善查询和更新代价。对现有技术分析后发现，分块方案的优劣以及划分后实施 prefix sum 技术对单元的不同组织方式是影响预计算方法的关键，特别是分块方案，是影响查询和更新时间复杂度的决定因素。

本文对预计算技术中的分块方案进行研究，对现有的各种分块方法进行总结，分析其实现方法和性能，并提出两种新的分块方案，嵌套的分块和基于前缀区域边界的分块。文中给出了这两种方法的实现和特点。研究表明，这两种分块方法有各自的优点，是对现有分块方案的有力补充。

1 引言

J. Gray 将聚集函数分为分布型、代数型和全局型[1]。分布型函数可以将数据划分为若干部分，对每一部分分别进行聚集计算，然后对每一部分的聚集结果再进行一次聚集计算得出最终结果。标准的分布函数有：COUNT、SUM、MIN、MAX。代数型聚集函数可以表示为分布型聚集函数的代数函数，例如 AVG 是代数型函数，可以表示为 SUM/COUNT。全局型聚集函数不能通过分块聚集的方法进行计算，如 MEDIAN。

本文讨论的分块方法需要对数据进行划分处理，因此聚集函数只能是分布型和代数型。以下讨论的聚集函数都以 SUM 为例进行说明。

给定 d 维数据立方体 DC，采用 $\prod_{i=1}^{d} n_i$ 的 d 维数组存储。不失一般性，可以假定每维的度都为 n，则原数据立方体占用 n^d 大小的空间。

研究人员对范围查询预计算方法的研究以 Prefix Sum (PS)[2]算法为基础。在 PS 中每个单元存储立方体中该单元前缀区域的累加和。即

$$p[x_1,x_2,\cdots,x_d]=Sum(A[0,0,\cdots,0]:A[x_1,x_2,\cdots,x_d])$$
$$=\sum_{i_1=0}^{x_1}\sum_{i_2=0}^{x_2}\cdots\sum_{i_d}^{x_d}A[i_1,i_2,\cdots,i_d]$$

对任意范围的数据进行查询时，可以通过图 1 所示的方法完成。

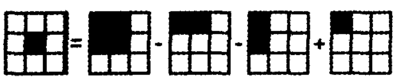

图 1 PS 范围查询

PS 方法需占用大小为 n^d 的存储空间，查询代价为 O(1)。但在 PS 中由于每个单元存放的是前缀区域的累加和，当某单元改变时，其后继所有单元的累加和都要随之改变，因此其更新代价为 $O(n^d)$。

PS 方法的更新代价过高，其原因在于数据之间的联系过于紧密，数据相互的依赖关系过高，解决的办法是使数据之间保持一定的独立，一般采用分块技术对问题进行解决。

2 现有分块方法研究

为改善查询和更新代价，在范围查询、数据更新及占用空间三者中实现很好的平衡，可以采用将数据立方体分块的方法。分块的目的是使块间的数据保持相互独立，当某一块的数据发生变化，不会影响其他块的数据或只影响少量数据。在设计分块方案时必须同时考虑数据立方体的高维及海量数据的特性。

现有的分块方案有以下几种。

2.1 固定分块

Relative Prefix Sum (RPS)[3] 和 Double Relative Prefix Sum(DRPS)[4]是典型的固定分块方法。该方法将数据立方体分成若干个固定大小的子块，在每个子块内部进行独立的的数据累加，同时为保持块间联系，增加附加空间存储块的综合数据。以 2 维 9×9 数据立方体为例，RPS 方法将立方体划分为 3×3 个独立的子块，每个子块包含 9 个单元。在每个子块内部各自实施 Prefix Sum 方法计算前缀累加和，同时为方便范围查询，RPS 采用额外的空间存储每一子块的边界以获取综合数据。当进行范围查询时，利用子块单元值以及子块边界值可以很快计算出区域的聚集值。图 2 给出其划分方法，并给出子块边界值的计算方法，图中带叉处单元为子块的边界单元。

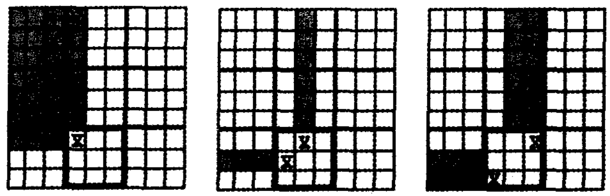

图 2 RPS 固定分块以及子块边界值的计算

由于采用了分块的方法，当立方体中单元内容发生改变时，只影响单元所在子块的累加和数据以及其他有限的几个边界值，不会引起整个立方体的变动，降低了更新代价。

固定分块方法的查询和更新时间复杂度与子块每维的度有关。分析 RPS 方法，假定在每一维，子块度为 k（简化起见，可以假定维长度 n 整除 k），在最坏情况下，更新效率可近似为 $k^d+dnk^{d-2}+(n/k)^d$，当 $k=\sqrt{n}$时，更新时间复杂度降为 $O(n^{d/2})$。

固定分块方法一般实现起来比较简单，但由于受到分块大小的限制，查询或更新效率不会提高很多。

2.2 *m* 叉树分块

m 叉树分块方法利用了著名的回归分割技术，其方法是先将数据立方体划分为 m 块，然后将每个子块继续划分为 m 块。这样反复分割，直到不能再划分为止。这种划分方法产生划分的层次，随着层次的增高，子块范围越来越小。

m 叉树的构造方法除了上述采用的自上而下法，还可以采用相反的自下而上的合成方法。这种方法开始时将立方体按最小分块大小分块，然后按规则将同一父块的相邻 m 个子块合并为一个块。反复进行直到不能再合并为止。

现有的 m 叉树分块方法又分为不带边界分块和带边界分块。

2.2.1 不带边界的 *m* 叉树分块

Hierarchical cubes (HC)[5] 和 Hierarchical data cube (HDC)[6]是两种不带边界的 m 叉树分块方法。在这种方法中，可以灵活地规定维及每层的划分数目。当划分不规则时，要将不同层次、不同维的各个划分点或划分块区域等相关参数存放在一个划分序列中，当进行划分时，按照划分序列中定义的点或位置进行划分。以二维数据立方体为例，采用四叉树分块，对应的划分层次如图 3 所示。

建立不带边界的 m 叉树划分方法，一般要分为两个步骤。首先将数据立方体分解，即进行分层、分块处理，根据划

分序列将数据立方体划分为若干子块，先划分为第一层，接着进行第二，三层…的划分，形成层次结构。第二步要进行数据立方体的映射，即考虑层次组织和单元的位置，对数据立方体中的每一个单元赋予不同层次、不同区域的累加值。数据立方体的映射方法可以有多种，HC 和 HDC 分别采用了各自的映射方法。当查询和更新时，可根据数据立方体的映射规则在不同层次取出相应的值，进行相应的运算即可。

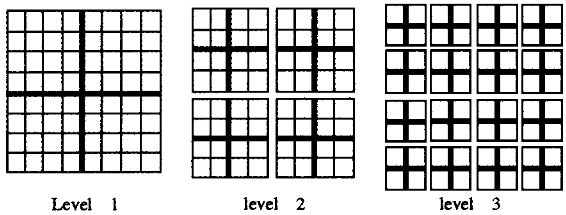

图 3 无边界四叉树的划分方法

不带边界的划分方法在最好情况下，可以实现查询和更新复杂度都为 $O(\log^d n)$，效率较高。但这种方法需要额外的存储空间存放立方体的映射数据和划分序列，占用较大存储空间。另外在这种方法中数据立方体的映射比较复杂，不利于实现。

2.2.2 带边界的 m 叉树分块

Dynamic data cube(DDC)[7] 和 Space-Efficient Dynamic Data Cube(SDDC)[9] 是带边界的 m 叉树分块的方法。这种方法在进行 m 叉树划分之后，定义子块的边界，再划分时除去每一块的边界，对子块剩余的单元继续划分为 m 块。这样反复分割，直到不能再划分为止。每个子块的边界一般存放块的综合数据，如块前缀区域的累加和等，也可以自行进行定义。以二维立方体为例，SDDC 方法的划分如图 4 所示，图中带叉单元为块的边界。

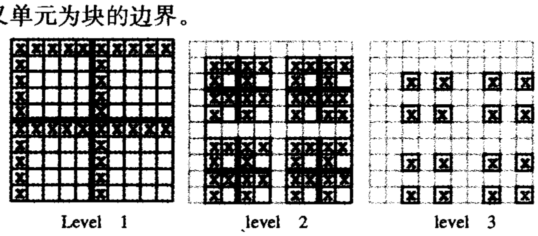

图 4 带边界四叉树的划分方法

带边界的 m 叉树分块方法，块的边界一般存放综合数据，是对块数据的综合描述。随着层次的划分和块范围的缩小，不同层次的边界形成了具有不同范围、不同粒度的聚集数据。在进行查询和更新时，可按照分块层次取出不同综合粒度的边界数据，再进行相应的计算即可。

采用带边界的 m 叉树分块方法，也可以实现查询和更新复杂度为 $O(\log^d n)$，效率较高。与不带边界分块比较，不带边界的分块方法中单元的值要通过复杂的映射规则计算得到，而带边界的 m 叉树分块方法其单元所处的层次位置非常清楚，如其值可为相应层次数据的累加，因此数据立方体的构造以及查询、更新都容易实现。更具有可取的是，采用带边界的 m 叉树分块方法在最好情况下，不需要额外的存储空间存放聚集数据，原则上只需和原数据立方体同样大小的存储空间就可以实现，存储代价低。这种方法是目前分块方案中较好的一种方法。

3 两种新的分块方案

本文在分析现有分块方案基础上进一步研究，提出两种新的分块方法。

3.1 嵌套分块方法

嵌套的分块方法其实是 m 分块方法中当 $m=1$ 时的特殊的分块，这种方法通过形成块间的嵌套结构，构造父块与子块的相似结构，使得父块包含子块，子块蕴含于父块。

其具体构造方法如下。先构造基本块结构，根据需要，可以选择不同数目、不同结构的方体作为基本块。然后采用固定分块的方法，将数据立方体按基本块大小分成若干个大小相同的子块，每个基本块的单元值被赋予不同范围的累加和。接着按规则合并子块，合并时从立方体的起点 $A[0,0,\cdots,0]$ 开始，将相邻基本块合并，形成较大的子块，同时根据子块单元所处位置，对部分单元值做相应修改。再继续合并时，以合并后的子块为基本块，将与其大小相同的相邻的子块合并成更大的子块。这样反复进行，直到不能再合并为止。

以二维立方体为例，如图 5 所示说明嵌套的分块方法。基本块取 $A[0,0]:A[1,1]$ 包含四个单元的方体区域，可以规定块内的单元值采用 Prefix Sum 方法，存放累加和，即 $p[0,0]=A[0,0]$，$p[0,1]=\sum_{j=0}^{1}A[0,j]$，$p[1,0]=\sum_{i=0}^{1}A[i,0]$，$p[1,1]=\sum_{i=0}^{1}\sum_{j=0}^{1}A[i,j]$。先将数据立方体划分为若干个大小相同的基本块(简化起见，可假定正好划分)，每个基本块中的单元值都与 $A[0,0]:A[1,1]$ 中相应位置的值相同。然后将数据立方体从起始位置 $A[0,0]$ 处开始，将相邻的四个基本块合并为包含 16 个单元的子块，范围为 $A[0,0]:A[3,3]$，合并后的子块为保持单元结构与基本块相同，部分单元值要做相应的修改。如 $p[1,3]$ 位置与基本方体 $p[0,1]$ 相同，因此 $p[1,3]=\sum_{i=0}^{1}\sum_{j=0}^{3}A[i,j]$；同理，$p[3,1]$ 与 $p[1,0]$ 位置相同，$p[3,1]=\sum_{i=1}^{3}\sum_{j=0}^{1}A[i,j]$；$p[3,3]$ 与 $p[1,1]$ 位置相同，$p[3,3]=\sum_{i=0}^{3}\sum_{j=0}^{3}A[i,j]$。继续使用同样的方法，将 $A[0,4]:A[3,7]$，$A[4,0]:A[7,3]$，$A[4,4]:A[7,7]$ 的区域也按上述规则合并，这样就有了四个包含 16 个单元的子块。再继续将这四个合并后的子块合并为包含 64 个单元的方体，范围区域为 $A[0,0]:A[7,7]$，同时修改部分单元值，$p[3,7]=\sum_{i=0}^{3},\sum_{j=0}^{7}A[i,j]$，$p[7,3]=\sum_{i=0}^{7}\sum_{j=0}^{3}A[i,j]$，$p[7,7]=\sum_{i=0}^{7}\sum_{j=0}^{7}A[i,j]$。照此反复合并下去，直至整个数据立方体。图中阴影部分表示与基本块 $p[0,1]$ 和 $p[1,0]$ 位置相同的单元，网格阴影部分表示与 $p[1,1]$ 位置相同的单元，其值要修改为对应位置的值。

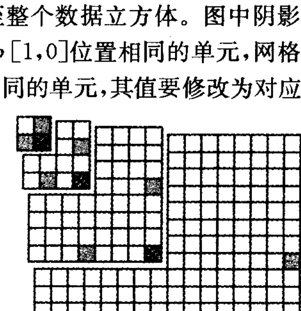

图 5 嵌套式分块方法

从上述分块过程可以看出，与 m 叉树分块方法不同的是，嵌套的分块方法每次只产生一个父块，而不是 m 个父块，这样保证了父块与子块之间的包含关系，形成了块的嵌套结

构。在合并子块时修改相应单元值,使父块与子块的单元组织结构相似,实现了部分与整体之间的有机联系。

嵌套分块方法的查询和更新时间复杂度与选取基本块的大小和结构有关,也与合并后对块单元值的不同设定有关。采用嵌套的分块的最大特点是可以利用块间的嵌套关系,即单元值累加和的包含关系,实现数据立方体快速的近似查询。近似查询是为了提高查询的响应时间,给出有一定误差的近似查询结果的一种方法。由于在嵌套分块单元中存放了不同综合粒度的聚集值,例如上述二维立方体分别存放包含 2^2, 2^4, 2^6, 2^8…个单元累加值的不同粒度的聚集值,因此近似查询时可以先取大范围的粗粒度的聚集值取平均作为查询结果快速返回精度不高的数据,然后进一步缩小范围,逐步取较细粒度的数据返回精度较高的查询结果,直至最后给出精确结果。

3.2 基于前缀区域边界分块

在数据立方体数组中,从起始单元 $A[0,0,\cdots,0]$ 开始的一片区域 $A[0,0,\cdots,0]$: $A[x_1,x_2,\cdots,x_d]$ 是前缀区域(Prefix Region),简称 $PR[x_1,x_2,\cdots,x_d]$。我们定义前缀区域的最外层数据为前缀区域的边界,边界数据包围前缀区域的内部数据。

前面提到的所有的分块方案,包括嵌套的方法都是规则的方体(hyper-rectangle)分块,这些方体分块大小或者相同,或者相差一定的倍数。基于前缀区域的边界数据分割方法,是一种不规则的分块方法,这种方法划分后的块大小、形状不同,利于从起始单元开始的前缀区域聚集值的计算。下面进行说明。

基于前缀区域边界的分块方法也是一种带边界的回归分割方法。不失一般性,可假定数据立方体每维的度都为 n。首先将原数据立方体作为第0层,然后对每维的度折半,以前缀区域 $PR[n/2,n/2,\cdots,n/2]$ 的边界作为第一次划分的分割单元,将立方体划分为两个没有交迭的子块,作为第一层。划分后的两个子块大小不相同,其中一个为规则方体,但另一个不再是规则的方体区域。继续将每个子块按维度折半分割为两部分,分割单元分别为:前缀区域 $PR[(n/2-1)/2,(n/2-1)/2,\cdots,(n/2-1)/2]$ 和前缀区域 $PR[(n/2+1+n)/2,(n/2+1+n)/2,\cdots,(n/2+1+n)/2]$ 的边界,形成立方体的第二层。继续采用回归分割的方法,对子块反复进行上述的划分,形成树状层次结构,得到不同层次的子块,直至不能划分为止。其树的深度,即层次数最多为 $\log n$。

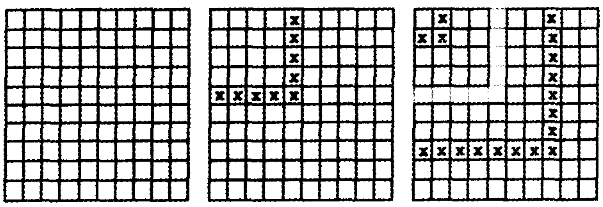

图6 基于前缀区域边界的划分

以2维10×10(下标为0~9)数据立方体为例说明基于前缀区域的分块方法。如图6所示,图中带叉处单元表示块的分割边界。先将原数据立方体作为第0层,然后将维度折半,用前缀区域 PR[4,4]的边界作为分割单元划分第一层,将原立方体分割为两个子块。除去边界继续划分,将每一块的维度再折半,对划分后的两个子块分别用前缀区域 PR[1,1]与 PR[7,7]的边界作为分割单元将立方体划分为四个子块,形成第二层。采用上述的方法反复进行回归分割,形成第三层、第四层,…,直到不能分割为止。

基于前缀区域边界的分块方法,其最大特点是不规则的划分。从上述分块过程可以看出,如果规定各层的边界值存放该层次前缀区域的累加和,则这种分块最接近 PS 的方法,但比 PS 优越的是边界中存放的是不同层次的累加和,因此不存在 PS 中的更新灾难问题。从前面的讨论可知,在数据立方体中任意区域的聚集范围查询都可以转化为计算若干个从起始单元开始的一片区域单元,即采用图1所示的方法。因此基于前缀区域的划分方案与前述方法相比,能充分适应这种区域特点,进行的划分能加快从起始单元开始的区域聚集值的计算速度,从而提高对正交区域范围查询的响应时间,查询响应速度较高。

结论 本文对基于 MOLAP 范围查询的预计算技术采用的分块方案进行研究,分析了现有分块方法的技术和性能,并提出了两种新的分块方法,嵌套分块和基于前缀区域边界的分块。这两种方法为分块方案提供了新的思路,是对现有分块方案的有力补充。今后我们继续在分块方案以及近似查询、联机聚集等方面进行研究,实现对范围查询的快速响应。

参考文献

1 Gray J, Bosworth A, Layman A, et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and subtotals. In: Proc. of the 12th International Conf. on Data Engineering, 1996. 152~159

2 Ho C T, Agrawal R, Megiddo R, et al. Range queries in OLAP data cubes. In: Proc. of the International ACM SIGMOD Conf., 1997. 73~88

3 Geffner S, Agrawal D, Abbadi A, et al. Relative prefix sums: an efficient approach for querying dynamic OLAP data cubes. In: Proc. of the 15th International Conf. on Data Engineering, 1999. 328~335

4 Liang W, Wang H, Orlowska M E. Range queries in dynamic OLAP data cubes. Data and Knowledge Engineer, 2000, 34(1): 21~38

5 Chan C Y, Loannidis Y E. Hierarchical cubes for range-sum queries. In: Proc. of the 25th VLDB Conf., 1999. 675~686

6 高宏,李建中,李金宝. 数据仓库系统中层次式 cube 存储结构. 软件学报, 2003, 14(7): 1258~1266

7 Geffner S, Agrawal D, Abbadi A. The dynamic data cube. In: Proc. of the EDBT, 2000. 55~77

8 Chun S J, Chung C W, Lee J H, et al. Dynamic update cube for range-sum queries. In: Proc. of the 27th VLDB Conference, 2001. 521~530

9 Riedewald M, Agrawal D, Abbadi A E, et al. Space efficient data cubes for dynamic environments. In: Proc. of the International Conf. on Data Warehousing and Knowledge Discovery, 2000. 24~33

10 Bengtsson F, Chen J. Space-efficient Range-sum Queries in OLAP. In: Proc. of 6th International Conf. on Data Warehousing and Knowledge Discovery, 2004. 87~96