用于聚集值近似查询的基于密度的树索引结构*)

许 俭 吴天轶 王 晨 汪 卫 施伯乐
(复旦大学计算机与信息技术系 上海 200433)

摘 要 如何快速有效地对数据立方体上的聚集查询给出近似的回答,是数据挖掘和数据仓库研究领域中的核心问题之一。现有大多数聚集查询算法在同一个数据立方体上只能支持某种特定的而非多种类型的聚集查询。本文给出了一种新的框架 AdenTS,即基于密度的自适应树结构,它可以回答同一数据立方体上的各类聚集查询,也提出了一些近似和启发式技术,改善了查询结果和精度。实验结果表明,这种方法在支持的查询种类和性能上是更好的。

关键词 聚集查询,近似查询,密度,树结构

A Density-based Tree Structure for Approximating Aggregate Queries

XU Jian WU Tian-Yi WANG Chen WANG Wei SHI Bai-Le
(Department of Computing and Information Technology, Fudan University, Shanghai 200433)

Abstract In many fields and applications, it is critical for users to make decisions through OLAP queries. How are accuracy and efficiency promoted while answering multiple aggregate queries, e. g. COUNT, SUM, AVG, MAX, MIN and MEDIAN? It has been the urgent problem in the fields of OLAP and data summarization recently. There have been a few solutions such as MRA-Tree and GENHIST for it. However, they could only answer a certain aggregate query which was defined in a particular data cube with some limited applications. In this paper, we develop a novel framework ADenTS, e. g. Adaptive Density-based Tree Structure, to answer various types of aggregate queries within a single data cube. We represent the whole cube by building a coherent tree structure. Furthermore, several techniques for approximation and heuristic approaches are proposed to improve the accuracy of query answering. The experimental results show that the method outperforms others in effectiveness and efficiency.

Keywords Aggregate query, Density-based, Approximation, Tree structure

1 引言

快速有效地对数据立方体上的聚集查询给出近似的回答是数据挖掘和数据仓库研究领域中的核心问题之一。典型的聚集函数有 COUNT、SUM、AVERAGE、MAX、MIN、MEDIAN 等。

给出近似回答而不是精确回答的原因有两个方面:首先,在大多数情况下,用户更关心的是数据集带来的趋势性信息,在这种情况下,近似回答是完全可以被接受的。其次,当数据集非常庞大并且查询负担非常重的时候,给出精确回答的计算代价是非常高的。

大多数现有的算法在建立数据立方体时只能支持某种特定的聚集查询,但现实应用中可能期望在同一个数据立方体上进行多种类型的聚集查询。

本文的方法为数据立方体建立一种高效的树索引结构表示,并提出了一些关键技术来提高回答各种聚集查询的精度和效率。我们的方法有如下几个特征:(1)能在同一个数据立方体上对各种聚集查询做出回答;(2)算法允许用户在"值"上指定一个查询范围,而大多数已有的算法不允许用户根据查询需要自己设定属性值的范围。(3)实验表明,我们的算法在回答查询的精度方面表现很好。

本文第 2 部分介绍相关工作,第 3 部分是建立树索引结构算法的描述,第 4 部分给出了在同一数据立方体上进行多种聚集查询的新技术以及它们的理论基础,第 5 部分提出了两种优化策略,而实验结果以及和现有算法的比较将在第 6 部分给出,最后是结论。

2 相关工作

已有的聚集值近似查询算法包括 quadtree structure[4], histogram techniques[2,3], kernel density estimator[2,5], clustering techniques[5] 和 wavelet decomposition[1] 等。

Multi-Resolution Aggregate Tree (MRA-Tree)[4] 提出了一种四叉树的多维索引结构,通过自上而下有选择地遍历树中结点来回答 COUNT、SUM、MIN、MAX、AVG 等查询。不同层上的结点代表了对空间不同粒度的划分,因此从树的根结点开始向下访问的结点越多,结果越精确。但该算法给出的查询结果的置信区间太大(100%),使得查询结果并不总能很精确。此外,该算法中的树索引结构有很大的信息冗余,因为树中双亲结点的信息完全包含在它们的子结点中了。

另一种常见的回答聚集查询的方法是利用概率密度函数来代表整个数据集。例如,高斯混合模型[5] 就被用来做数据拟合,给定参数 k 后,算法根据原始数据集生成 k 个聚类,并用高斯混合模型代表每个聚类,这样就达到了很高的数据压缩率。对于一个 n 维的数据集,这种方法只需要使用 $1+n+n(n+1)/2$ 个数字来表示一个聚类。然后通过计算概率分布函数的积分来给出如 COUNT、SUM 和 AVG 的近似查询结

*)本课题得到国家自然科学基金重点项目(69933010 和 60303008)和国家 863 高科技项目(2002AA4Z3430 和 2002AA231041)资助。

果。尽管这种方法达到了很高的数据压缩率，但缺点是它支持的查询种类是有限的，而且它的查询结果的精度依赖于高斯分布的假设和原始数据的概率分布情况的吻合程度，这使得该算法不具备很好的通用性。

柱状图技术 GENHIST[2] 被用来拟合多维数据集上的密度分布，可以回答关于 COUNT 的查询。它的核心是根据数据集的情况产生许多不同大小的桶(bucket)，并且允许这些桶相互重叠，通过从密度大的格中移除数据点的方法来使每个桶中的数据点分布趋于均匀。这种方法的优点是可以在只储存相对较少的参数的情况下达到较高的精确度。

3 新型树索引结构

3.1 基本定义和符号

用 $D(n,d)$ 表示含有 n 个数据点的 d 维数据集，其中每个数据点 $\langle P_0,P_1,\cdots,P_{d-1}\rangle$ 具有形式〈loc，value〉，其中 loc 代表 $\langle P_0,P_1,\cdots,P_{d-2}\rangle$，value 代表 P_{d-1}，即 $\langle P_0,P_1,\cdots,P_{d-2}\rangle$ 被认为是在 $d-1$ 维空间 R^{loc} 中的点(代表空间信息)，而 P_{d-1} 则被认为是这个点上的属性值，其值域是 R_{value}，聚集值查询就是针对这一维进行的。所有维都是连续有限的。

本文提出的树索引结构是二叉树的一种变体：树中每个结点 N 有 5 个域，分别是 Area、Count、Max、Min 和 Distrb。给定一个结点 N，Area(N)表示结点 N 所代表的(超)矩形区域，这里 Area(N)$\subseteq R^{loc}$。PointSet(N)表示那些落在区域 Area(N)里面的所有数据点的一个子集。形式化地说：(1) PointSet(N)$\subseteq\{P\mid \forall P\in D(n,d)\wedge P_0P_1\cdots\wedge P_{d-2}\in$ Area(N)$\}$；(2)Count(N)$=\parallel$ PointSet(N) $\parallel$；(3)Max(N)$=\max\{P_{d-1}\mid \forall P\in$ PointSet(N)$\}$；(4)min(N)$=\min\{P_{d-1}\mid \forall P\in$ PointSet(N)$\}$。

树的根结点代表了整个 R^{loc} 空间，每个非叶结点可能有一个或者两个子结点。对应于树中每个结点的 PointSet 是互不相交的，但是它们的并恰好是整个数据点集(在后面将看到如何实现)，因此在我们的树结构中不存在冗余信息。即：$\forall N_1\,\forall N_2\Rightarrow$PointSet($N_1$)$\cap$PointSet($N_2$)$=\varnothing$；$D(n,d)=\{P\mid \exists N\Rightarrow P\in$PointSet($N$)$\}$。

另外，我们还在每个结点上存储关于该结点上的 PointSet(N)中所有数据点在属性值维 Rvalue 上的分布信息 Distrb(N)。Distrb(N)的存储方式可以是多样化的，典型地我们可以把分布所服从的模型和该模型上的参数存储在树结点中。

3.2 基于密度的自适应建树过程

首先设定两个参数 h 和 p，其中 h 是允许建立的最大树高，p 是决定是否要把当前结点分裂的一个密度阈值。初始时，只有一个代表 $d-1$ 维全空间 R^{loc} 的根结点 Root，首先把它沿第一维分成两个同样大小的子空间 R_1、R_2。如果 R_1 中的数据点密度大于阈值 p，那么就为根结点创建左子结点 N_1(代表空间 R_1)，否则不创建。同样，如果 R_2 的密度大于 p，那么为根结点创建右子结点 N_2。我们再用同样的方式沿第二维分裂 R_1 和 R_2，如果它们的子空间的密度超过阈值 p，就为 N_1 和 N_2 建立相应的子结点。分裂的停止条件是树高已经达到 h，或者已经不存在密度超过阈值 p 的结点了。

在树的不同层次上，我们分裂空间时所沿的维是不同的，实际上是按照先沿第一维，再沿第二维，接着第三、第四，一直到第($d-1$)维再沿第一维这样的顺序循环进行的。这样，给定树中的任意一个结点 N，我们能很快知道它所代表的(超)矩形区域空间的形状和范围。值得注意的是，在同一层上的所有结点代表的空间的形状是一样的。算法框架见图 2。

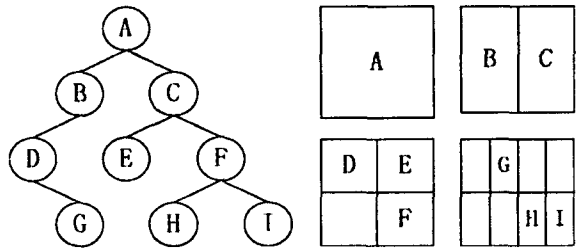


图 1 基于密度的建树过程

输入：$D(n,d)$，最大树高 h，分裂阈值 p
输出：基于密度的二叉树 TR
方法：ADTreeCons($D(n,d)$,h,p)
1. depth 当前树深度
2. IF depth$\geq h$ or $n<p$
3. RETURN Null;
4. 创建与 $D(n,d)$ 对应的结点 N 并设置 Area(N)
5. $k\leftarrow$depth mod ($d-1$)
6. 沿第 k 维将 $D(n,d)$ 分裂为等体积子空间 D_1D_2;
7. N. LeftChild$\leftarrow$ADTreeCons(D_1,h,p)
8. N. RightChild$\leftarrow$ADTreeCons(D_2,h,p)
9. RETURN N;

图 2 基于密度自适应建树过程

3.3 自底向上的更新过程

从上一步中得到了树的基本结构，我们就可以用一种自底向上的方法来为每个结点填充 Count、Min、Max 和 Distrb 信息了。具体方法如下：

从树的最底层开始，通过一种类似于 GENHIST[2] 中的方法：给定一个结点 N，如果 Area(N)中数据点的密度大于它周围区域数据点的平均密度，那么就从 Area(N)中随机地取出一些点，使得剩余的密度和它周围的区域保持一致，并把取出来的这些点存放在 PointSet(N)中。这个过程可以被形象地称为“削山峰”。通过这样的一个步骤，那些密度很大的区域会变得平坦，剩余的那些数据点在 R^{loc} 中的分布更趋于均匀(图 3)。

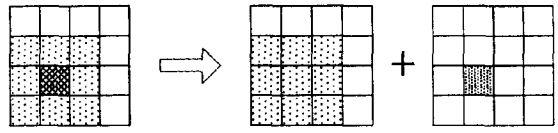


图 3 削山峰技术

当我们获得了和结点 N 对应的 PointSet(N)时，我们就很容易计算出这个结点上的 Count、Min、Max 和 Distrb。特别地，分布函数的信息既可以由用户事先输入，也可以通过 χ^2 测试得到，参数也可以通过无偏估计的方法计算出来。

每一轮的迭代，我们考虑更上一层的结点。在每一轮中，每个结点代表的区域是上一轮的两倍。迭代的终止条件可以是结点代表的区域足够大了，或者所有的数据点都已经通过“削山峰”的方法被移出并存储在相应结点中，或者是已经达到根结点。算法的细节见图 4。

输入：二叉树 TR，数据集 $D(n,d)$，最大格阈值 S
输出：更新后的树 TR
方法：ADTreeUpdate(TR,$D(n,d)$,S)
1. level 树的最底层;
2. REPEAT
3. FOR 当前层的所有结点
4. $p\leftarrow$Area(N)的密度;
5. $q\leftarrow$Area(N)周围的密度;
6. IF $p>q$ // 削山峰

```
7.      PointSet(N)←从 Area(N)中随机选取 (p−q) * Count(N) 个点;
8.      更新 Count(N),Max(N),Min(N),Distrb(N);
9.      level←level-1;
10.  UNTIL gridSize>S or level=0;
11. 用剩余的数据点更新第 level 层
12. RETURN TR
```

图 4　自底向上更新结点信息

4　聚集值的近似查询

一个聚集查询包括三个部分:(1)查询类型,用来指定对属性值上的哪种聚集值进行查询(如 SUM);(2)一个用户指定的($d-1$)维的查询区域 $Q(Q\subseteq R^{loc})$;(3)用户指定的属性值上的查询区间 $T(T\subseteq R^{value})$。最后那个部分是可选的(主要用在 COUNT 和 SUM 的查询上,在 MIN 和 MAX 的查询上是没有意义的),它的默认值是 R^{value}。比如,前面所述的慈善捐款数据库上的一个典型的查询可以表示为:{AVG, Q(40-50⋃10K-20K), T(1K-2K)}。

4.1　COUNT

对计数(COUNT)查询的回答 E_{count} 首先被初始化为 0,然后从树的根结点开始遍历树中的每个结点。记第 i 个遍历到的结点为 $N^{(i)}$。如图 5 所示,查询区域 Q 和 Area($N^{(i)}$)只有 4 种可能的位置关系[4]:(a)嵌入,(b)包含,(c)部分重合,(d)不相交。

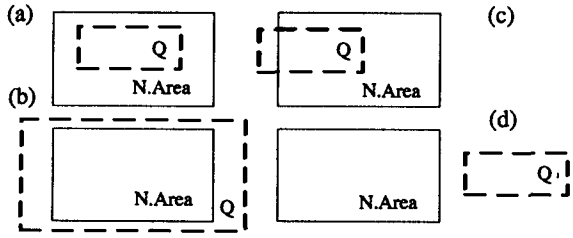


图 5　结点和查询的关系

对于情况(d),我们简单地返回 0,并且停止对 $N^{(i)}$ 的左右子结点的遍历;对于其余的三种情况,基于对 PointSet($N^{(i)}$)在 R^{loc} 上是均匀分布的假设(这种假设的根据是我们前面采用了"削山峰"的技术),我们可以给出对应于 $N^{(i)}$ 的 $E^{(i)}_{count}$ 的一个估算:

$$E^{(i)}_{count}=\mathrm{Count}(N^{(i)})\times\int_T p^{(i)}(\xi)\mathrm{d}\xi\times\frac{\mathrm{Area}(N^{(i)})\cap Q}{\mathrm{Area}(N^{(i)})}$$

其中 $p^{(i)}(\xi)$ 是从 Distrb($N^{(i)}$)得到的密度分布函数。这样,如果在查询过程中总共有 k 个结点被访问到,那么最后返回的查询结果就是 $E_{count}=\sum_{i=1}^{k}E^{(i)}_{count}$。

4.2　SUM 和 AVERAGE

和(SUM)及平均值(AVG)的查询与计数(COUNT)的查询是类似的,我们有

$$E^{(i)}_{sum}=\mathrm{Count}(N^{(i)})\times\int_T \xi p^{(i)}(\xi)\mathrm{d}\xi\times\frac{\mathrm{Area}(N^{(i)})\cap Q}{\mathrm{Area}(N^{(i)})}$$

$$E_{sum}=\sum_{i=1}^{k}E^{(i)}_{sum}$$

平均数:$E_{average}=\dfrac{E_{sum}}{E_{count}}$。

4.3　MAX 和 MIN

在考虑最大值和最小值时,由于我们前面已经假设属性值域是连续的,在查询的时候没有必要对 T 设置一个界限。不失一般性,下面我们只讨论如何回答最大值(MAX)查询。

从树根结点开始自顶向下依次访问各个结点。在每个结点上做查询时,对于 4.1 节中所述情况(d)我们简单地置 $E^{(i)}_{max}=-\infty$;对于情况(b),很显然 $E^{(i)}_{max}=\mathrm{Max}(N^{(i)})$;而对于情况(a)和(c),无法保证给出正确的最大值,但问题可以转化为"如果给定 $n=\mathrm{Count}(N^{(i)})$ 个数据点,并且从中随机选取 $m=\dfrac{\mathrm{Area}(N^{(i)})\cap Q}{\mathrm{Area}(N^{(i)})}\times n$ 个点,那么这些点中最大属性值的数学期望是什么?"假设这 n 个数据点上的属性值分别是 V_1, $V_2,\cdots,V_n$ 且 $V_i\leqslant V_{i+1}$,可以利用组合数学的原理得到最大值的期望

$$E^{(i)}_{max}(m)=\sum_{i=m}^{n}P(\max=V_i)\times V_i=\frac{\binom{m-1}{m-1}\times V_m+\binom{m}{m-1}+\cdots+\binom{n-1}{m-1}\times V_n}{\binom{m}{n}}$$

但这种方法的计算代价太大。注意到 $E^{(i)}_{max}$ 是关于 m 的单调递增函数,且

$$E^{(i)}_{max}(1)=E^{(i)}_{max}(\max(V_1)=E^{(i)\xi}(V_1)=E^{(i)\xi}$$

这里是 $E^{(i)}\xi$ 是 $N^{(i)}$ 中属性值分布的数学期望;又 $E^{(i)}_{max}(n)=\mathrm{Max}(N^{(i)})$,因此我们可以对最大值的数学期望做出一个线性估计:

$$E^{(i)}_{max}=E^{(i)}_{max}(1)+(m-1)\times\frac{E^{(i)}_{max}(n)-E^{(i)}_{max}(1)}{n}=\frac{1}{n}\times\mathrm{Max}(N^{(i)})+\frac{n-m+1}{n}\times E^{(i)}\xi$$

这种估算方法的好处是可以直接利用保存在结点中的关于属性值分布的信息,而不需要由 $V_1,V_2,\cdots,V_n$ 重新计算最大值。这样,最终查询结果就是 $E_{max}=\max\{E^{(i)}_{max},1\leqslant i\leqslant k\}$,其中 k 是被访问的结点的总数。

4.4　MEDIAN

属性值的概率分布信息在回答其他各种查询时是非常有用的。这里,我们讨论如何利用概率分布信息来回答关于中位数(MEDIAN)的查询。

仍然记第 i 个被访问的结点为 $N^{(i)}$, PointSet($N^{(i)}$)的中位数为 $E^{(i)}_{median}$,那么关于未知数 x 的方程 $\int_{\xi\leqslant x\wedge\xi\in T}p^{(i)}(\xi)\mathrm{d}\xi=\int_{\xi>x\wedge\xi\in T}p^{(i)}(\xi)\mathrm{d}\xi$ 的解就是我们所要求的 $E^{(i)}_{median}$。这样,中位数查询的最终结果就是

$$E_{median}=\frac{\sum_{i=1}^{k}\mathrm{Count}(N^{(i)})\times E^{(i)}_{median}}{\sum_{i=1}^{k}\mathrm{Count}(N^{(i)})}$$

其中 k 是访问的结点数。

5　建立树索引结构的优化方法

5.1　双削技术

在第 3 部分中提出了"削山峰"技术,使得记录在每个树结点中的数据点分布趋于均匀。这里提出一种改进,目的是使"削山峰"的效果变得更好。

原来的算法中是从树的最底层开始对每个结点依次进行"削山峰"的。这样做的一个明显弱点就是,往往会遇到一个结点具有左右两个子结点,这时如果先对其中一个子结点进行"削山峰"(这时它的兄弟结点被看作是它的周围环境),那

么由于它的兄弟结点也是密的，将影响到该子结点“削山峰”的效果。改进方法是对这样的两个子结点同时进行“削山峰”。

200	200	200	200
200	N_1 1000	N_2 600	200
200	200	200	200

图 6　双削技术

例如，有一个数据集的密度分布如图 6 所示，按照原先的算法，将得到 Count(N_1)＝1000－(200×7＋600)/8＝750，Count(N_2)＝600－(200×7＋750)/8＝331。经过使用这里提出的“双削”技术后，将得到更好的结果 Count(N_1)＝800 和 Count(N_2)＝400。

5.2　树结构的调整

在前面所述的 ADTreeUpdate 算法中，很可能存在 PointSet(N)为空的结点，这会在决定结点该存在哪个 I/Q 页面时造成不利的影响。可以把那些有子结点的空结点删去，并把它的子结点上提，作为它的双亲结点的子结点。经过调整后，树的结构变得更紧凑，但可能不再是二叉树了。可以设置每个结点的最大扇出数 β 来避免结点有太多的子结点。

6　实验

实验使用的数据是描述美国森林覆盖类型的真实数据集，查询集是通过程序随机生成的。算法用标准 C++ 实现，运行环境是 IBM PM1.5GHz 256M/DDR333。

查询精度可以用相对误差的形式来衡量：相对误差＝|正确结果－查询结果|/正确结果。对于 MIN 和 MAX：相对误差＝|正确结果－查询结果|/属性值域。

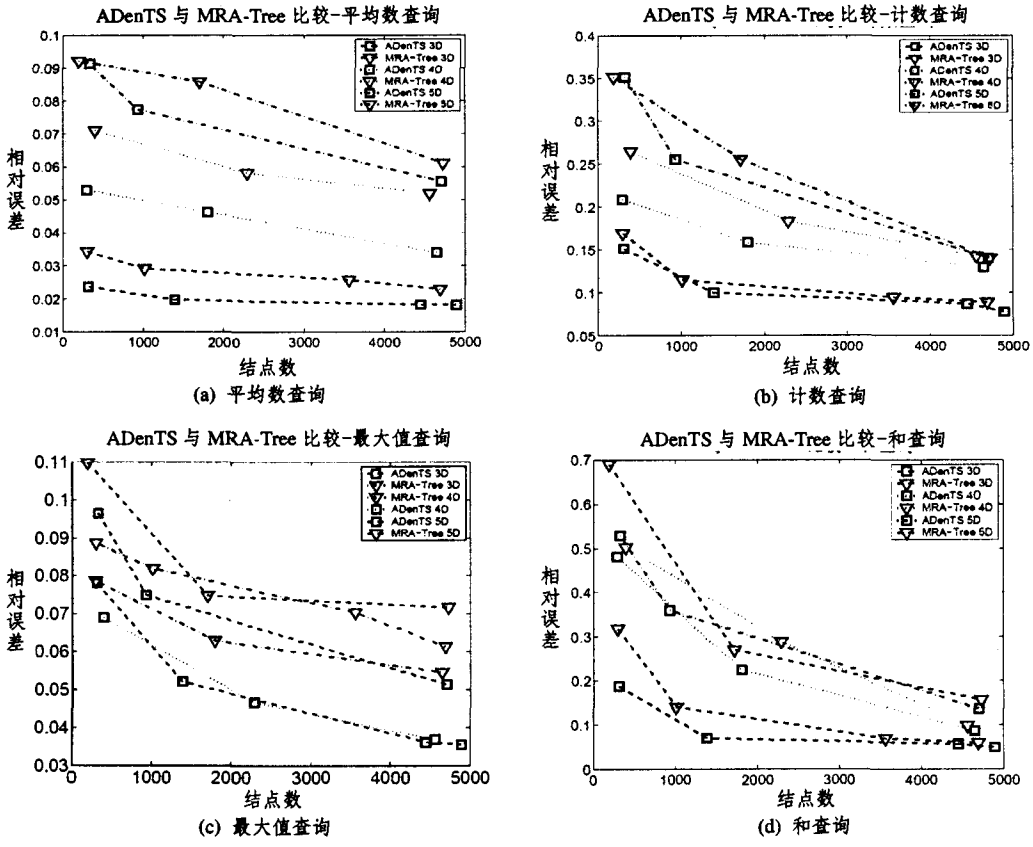
6.1　和 MRA-Tree 以及 GENHIST 的比较

我们把实验分成如下 3 组：(1)只有 ADentTS 所能做的查询；(2)ADenTS 和 MRA-Tree 都能做的查询；(3)ADenTS 和 GENHIST 都能做的查询。

图 8 的(a)、(b)分别说明了在 3 维和 4 维空间上 ADenTS 在对 COUNT、SUM、AVG、MAX、MIN、MEDIAN 查询时的精度和索引树大小的关系，横坐标是索引树中结点的个数，纵坐标是相对误差。

由于 MRA-Tree 不支持带有属性值区间信息的查询，也不支持中位数的查询，我们只生成了 COUNT、SUM、AVG、MAX 和 MIN 这 5 种查询，从图 7 可见，我们的方法在 3 到 5 维空间中均比 MRA-Tree 表现得好，原因是在 ADenTS 中“削山峰”的技术使得在数据点在单个区域中的分布更均匀。另一方面，MRA-Tree 的存储结构是有信息冗余的，而我们方法中的树结构不存在信息冗余，因为每个结点中存储的数据点集是互不相交的。

GENHIST 只能做 COUNT 查询，AdenTS 在回答 COUNT 查询时的基本思想和 GENHIST 是一致的，但由于采用了“双削”的优化技术，使得我们的算法的表现略好于 GENHIST(如图 9 所示)。



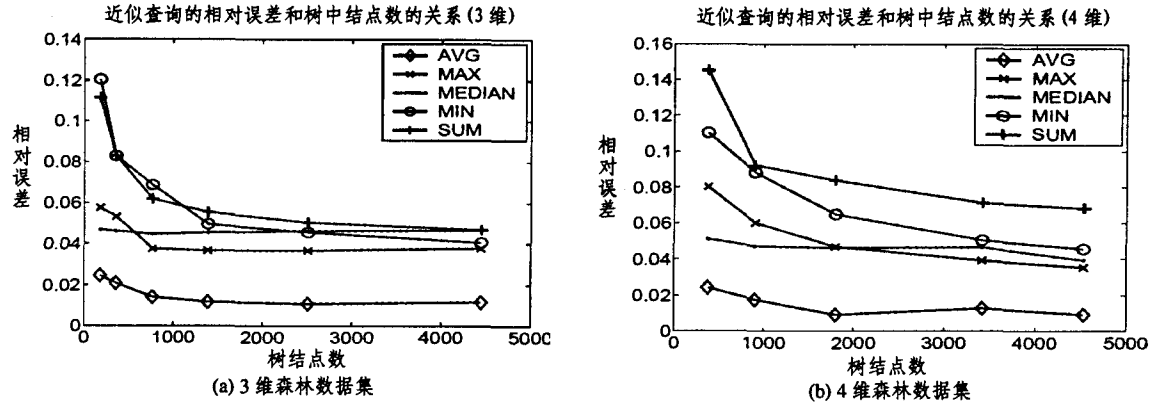
(a) 平均数查询　(b) 计数查询

(c) 最大值查询　(d) 和查询

图 7　AdenTS 和 MRA-Tree 的比较

结论　本文提出了一种高效的树索引结构，用以对多维数据立方体上的各种聚集值查询做出回答。树索引结构是通过基于密度的自顶向下和自底向上相结合的自适应方法建立的，回答查询则基于保存在每个树结点中的概率分布信息。该方法继承了 MRA-Tree 结构和 GENHIST 技术的优点。实验证明，这个方法在支持的查询种类上和性能上均超越了

上述两种方法。



(a) 3 维森林数据集　　(b) 4 维森林数据集

图 8　AdenTS 的性能

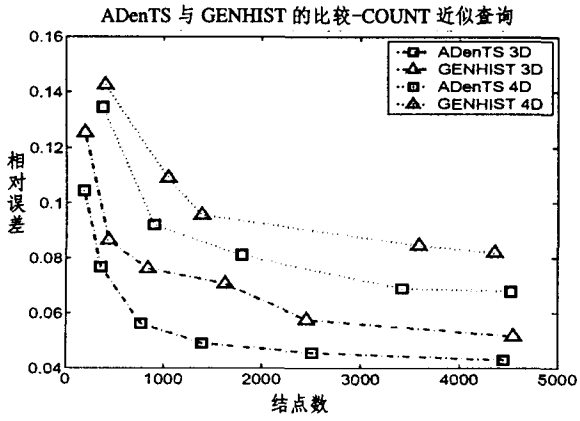


图 9　ADenTS 与 GENHIST 比较

参 考 文 献

1 Chakrabarti M N, Garofalakis R R, Shim K. Approximate Query Processing Using Wavelets. In: Proc 26th Int Conf. on Very Large Data Base (VLDB '00), Cairo, Egypt, 2000

2 Gunopulos D, Kollios G, Tsotras V J. Approximating Multi-dimensional Aggregate Range Queries over Real Attributes. In: Proc. ACM SIGMOD 19th Int Conf on Management of Data (SIGMOD '00), Dallas, USA, May 2000

3 Lee J, Kim D, Chung C. Multi-dimensional Selectivity Estimation Using Compressed Histogram Information. In: Proc. ACM SIGMOD 18th Int Conf. on Management of Data (SIGMOD '99), Philadelphia, USA, 1999

4 Lazaridis I, Mehrotra S. Progressive Approximate Aggregate Queries with a Multi-Resolution Tree Structure. In: Proc. ACM SIGMOD 20th Int Conf. on Management of Data (SIGMOD '01), Santa Barbara, USA, 2001

5 Shanmugasundaram J, Fayyad U, Bradley P S. Compressed Data Cubes for OLAP Aggregate Query Approximation on Continuous Dimensions. In: Proc. ACM SIGKDD 6th Int Conf. on Knowledge Discovery and Data Mining (KDD'99), San Diego, USA, 1999

(上接第 87 页)

表 1　SMDD 部分相似度计算结果

S_1	S_2	Similarity
major	**Falculty_name**	**0.8506**
Student-score	**S_score**	**0.9199**
score	**S_score**	**0.9199**
sex	**S_sex**	**0.9272**
name	**S_name**	**0.9942**
name	S_address	0.9942
score	S_height	0.8913
age	S_score	0.4260
age	Falculty_name	0.1087
…	…	…

SMDD 的匹配质量很大程度上依赖于数据源中数据分布的规律性。在数据分布较为规律的情况下，SMDD 匹配质量较好；在数据分布不规律的情况下，SMDD 的计算效果不理想。实验中，我们通过对现有的一些科学试验、商业销售等方面的专业数据库进行分析，显示在数据分布较为规律的状态下，SMDD 对的匹配质量 F-Measure(0.5)最高可达到 0.65；一般情况下，F-Measure(0.5)介于 0.25～0.65 之间。

结束语　近年来，随着数据共享需求的不断提高，模式映射作为数据管理的基本问题已得到了广泛的重视，出现了很多模式匹配方法。现有的大部分模式匹配方法仍是以利用数据源的元素名称、结构等模式信息为主，对于数据内容信息的利用较为有限。本文提出的 SMDD 方法通过挖掘数据内容信息，根据元素的数据分布特征计算模式元素之间的相似度。SMDD 方法适用于数据分布具有一定规律的异构数据源之间的模式匹配，既可以独立使用，也可以与其它模式匹配方法结合使用，互为补充，从数据内容的角度提高匹配质量。

参 考 文 献

1 Doan A H. Learning to Map between Structured Representations of Data: [PhD thesis]. University of Washington

2 Madhavan J, Bernstein P A, Rahm E. Generic Schema Matching with Cupid. VLDB 2001

3 Doan A H, Domingos P, Halevy A. Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach. In: SIGMOD Record, 2001

4 Melink S, Garcia-Molina H, Rahm E. Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In: Proc 18th Intl Conf. on Data Engineering, 2002

5 Dell'Erba M, Fodor O, Ricci F, et al. Harmonise: A Solution for Data Interoperability. IFIP I3E 2002

6 Rahm E, Philip A. Bernstein. A survey of approaches to automatic schema matching. VLDB Journal, 2001, 10: 334～350

7 Do H H, Rahm E. COMA-A system for flexible combination of schema matching approaches. In: Proc. 28th VLDB Conference

8 史忠植. 知识发现. 北京：清华大学出版社，2004