ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.4, April 2006, pp.806−813
DOI: 10.1360/jos170806

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

无须附加空间的数据立方体联机聚集*

李红松+, 黄厚宽

(北京交通大学 计算机与信息技术学院,北京 100044)

Online Aggregation on Data Cubes Without Auxiliary Information

LI Hong-Song+, HUANG Hou-Kuan

(School of Computer & Information Technology, Beijing Jiaotong University, Beijing 100044, China)

+ Corresponding author: Phn: +86-10-51683602, E-mail: hongsongli@msn.com, http://www.njtu.edu.cn

Li HS, Huang HK. Online aggregation on data cubes without auxiliary information. *Journal of Software*, 2006,17(4):806−813. http://www.jos.org.cn/1000-9825/17/806.htm

Abstract: Typically, online aggregation algorithms on multi-dimensional data need additional auxiliary data for estimation, which make the performance of the storage and maintenance of the data cube worse. This paper presents the PE (progressively estimate) and HPE (hybrid progressively estimate) to progressively estimate the answers for range queries in the QC-Trees. MPE (multiple progressively estimate) is also proposed to simultaneously evaluate batches of range-sum queries. The difference between the algorithms and other online aggregation algorithms on data cubes is that these algorithms do not need any auxiliary information. The idea of this estimation method is to utilize the data stored in the QC-Tree itself. As a result, this algorithm will not deteriorate the performance of the storage and maintenance of the data cubes. Analysis and experimental results show that the algorithms provide an accurate estimation in far less time than the normal algorithms.

Key words: data cube; OLAP (on-line analytical processing); query approximation; online aggregation

摘 要: 以往在数据立方体上实现的联机聚集往往需要附加空间来存储联机聚集估算所需要的信息,极大地影响了数据立方体的存储和维护性能.提出了基于 QC-Tree 的用于范围查询处理的联机聚集 PE(progressively estimate)算法以及它与简单聚集算法相结合的混合聚集算法 HPE(hybrid progressively estimate);还提出了一种能够同时处理多个范围查询的联机聚集算法 MPE(multiple progressively estimate).与以往联机聚集算法不同,这些算法不需要任何附加空间,而是利用 QC-Tree 自身保存的聚集数据和语义关系来估算聚集结果.由于 QC-Tree 是一种极为高效的数据立方体存储结构,因此能够以较理想的性能实现数据立方体上的联机聚集.对算法的分析和实验结果表明,所提出的算法具有较好的性能.

关键词: 数据立方体;联机分析处理;近似查询处理;联机聚集
中图法分类号: TP311 文献标识码: A

联机分析处理(on-line analytical processing,简称 OLAP)是数据仓库的一种基本的数据分析服务.数据立方

* Supported by the Key Technology R&D Programe Foundation of China under Grant No.2002BA407B01-2 (国家科技攻关计划); the Special Science Foundation of Beijing Jiaotong University of China under Grant No.2003SZ003 (北京交通大学科技专项基金)
Received 2005-03-30; Accepted 2005-10-10

体是实现 OLAP 的主要手段.QC-Tree[1]是一种新型的数据立方体的高效存储结构,它在极大减少数据存储容量的同时,还保持了原数据之间存在的语义关系.这使得我们可以很方便地进行上卷和下钻等 OLAP 操作.

尽管 QC-Tree 在拥有良好的存储性能的同时,对点查询的处理能力也非常强,但 QC-Tree 对范围查询(range queries)的处理能力并不突出.例如,在对 6 维气候数据[2](1 百万条最细节记录)进行的 1~3 维的范围查询中,其平均响应时间约为 1s.而且当数据的维数增加时,响应时间呈指数增长[1].

另一方面,在进行范围查询时,完全精确的结果并不总是必需的.在这种情况下,查询处理可以采用联机聚集(online aggregation)的策略[3].也就是说,可以先花较少的时间向用户返回一个比较粗略的估算结果,然后不断修正并返回更为精确的估算.对于大数据量的查询处理,联机聚集是一种比较合算的查询策略.

然而,过去在数据立方体上实现的联机聚集算法往往需要使用许多附加空间以存储在联机聚集过程中所必需的信息.这些附加空间增加了数据立方体存储空间和维护操作的负担,影响了整体性能.

本文所做的贡献是:(1) 提出了不需要附加存储空间的联机聚集 PE(progressively estimate)算法,并对其性能进行了分析;(2) 将 PE 与普通聚集算法相结合,提出了一种混合算法 HPE(hybrid progressively estimate),该算法可以进一步提高联机聚集算法的性能;(3) 提出了一种可以同时高效处理多个范围查询的算法 MPE (multiple progressively estimate);(4) 使用模拟和真实数据,对 PE,HPE 和 MPE 的查询性能进行了实验.实验结果说明,我们的联机聚集算法可以显著地改善 QC-Tree 的范围查询处理能力.

本文第 1 节介绍 QC-Tree 的特点,并对与本文相关的工作进行介绍.第 2 节提出一种单纯的通过概率来估算聚集结果的 PE 算法,并对其进行了分析.为提高算法的综合查询性能,第 3 节提出一种混合的查询处理算法 HPE 和处理多范围查询的算法 MPE.第 4 节给出实验结果及其分析.第 5 节是结论.

1 相关工作

随着进入数据仓库的数据越来越多,数据立方体的体积也越来越大.单纯压缩算法带来的问题是,在取数据时常常需要解压缩,损坏了原结构中的数据间的语义关系,难以提高查询效率.

浓缩立方体(condensed cube)[4]、侏儒立方体(dwarf cube)[5]、商立方体(quotient cube)[6]以及其后继 QC-Tree[1]是近些年来出现的一系列新型的数据立方体的存储结构,则部分地解决了这个问题.它们可以认为是为数据建立的索引,并将由相同底层数据聚集生成的不同 *cell* 用同一个存储单元表示,这样就在实质上提高了数据立方体的存储效率.同时,这些结构还能够保持数据间的语义关系,从而使这些结构的查询性能也得到提高.QC-Tree 是这些结构中综合性能最好的一个.

QC-Tree 是使用树状结构进行存储的商立方体.在 QC-Tree 中,每一个节点都有一个维值与之相联系,并将它与其兄弟区分开.而且,任何节点的每一个孩子的维标号(维值所在维的序号)都小于这个节点的维标号.并且,该节点任何一个维的所有孩子节点值的聚集值等于该节点的值.例如,对于一个维为(*A*,*B*,*C*)的立方体,如果以 *A*,*B*,*C* 为序构建一个 QC-Tree,若某一节点维值所在维为 *A*,则其孩子节点维值所在的维只能是 *B* 或 *C*.若聚集函数是 *sum*,则该节点的值等于它所有 *B* 维孩子节点值的和,也等于它所有 *C* 维孩子节点值的和.

为了说明 QC-Tree 在结构上的特点,我们使用图 1 显示了一个 QC-Tree 的一部分.这个 QC-Tree 上维的次序为 *A*,*B* 和 *C*,聚集函数为 *sum*.在后文 PE 算法的第 5 步中,一个节点使用(维标号,维值)的标签形式进行标识.节点本身的聚集值在标签下方显示.例如,节点 2 的标签为(*A*,3),这表示节点 2 的维标号为维 *A*,并且在 *A* 维上的维值为 3.节点 2 本身所保存的聚集值等于 18,这表明 *sum*(3,*,*)=18.

节点在任何一个维上的所有子节点的聚集值的和也等于这个节点本身的聚集值.例如,节点 1(根节点,表示(*,*,*))的聚集值为 27,它在 *A* 维上的子节点包括节点 2(表示 *cell*(3,*,*),聚集值为 18),节点 6(表示 *cell*(1,9,*),聚集值为 9).这两个节点的聚集值之和也等于 27.而节点 1 在 *B* 维上的子节点:节点 4(同时表示 *cell*(*,2,*)和 *cell*(3,2,*)),节点 7(表示 *cell*(*,1,*)),节点 8(表示 *cell*(*,5,*)),节点 5(表示 *cell*(*,2,*)),聚集值之和也等于 27.

从查询处理上来说,对多维数据的查询主要有两种类型:点查询(point query)和范围查询(range query).范围查询是联机分析处理研究和应用中最重要的领域之一.近年来,已经出现了大量的文献探讨如何提高范围查询

的处理效率.这些文献的思路主要分为 3 种:(1) 使用预计算数据返回准确值,主要包括前缀和立方体(prefix sum cube)[7]以及由它延伸出来的其他算法[8];(2) 对立方体中的数据进行采样[9]或近似压缩[10],返回近似结果;(3) 联机聚集[11−13].

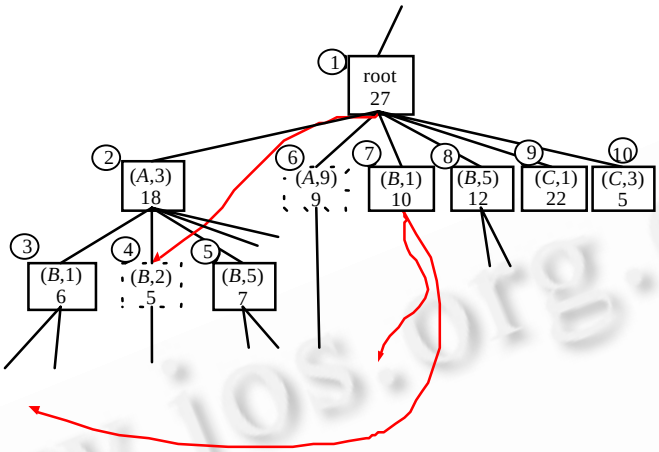


Fig.1 A sample of QC-Tree

图 1 一个 QC-Tree 的例子

尽管以上提到的所有范围查询处理方法可能在某些方面能够很好的工作,但它们通常无法同时满足 3 个条件:(1) 能够很快地提供一个较为精确的答案;(2) 当需要时,提供准确的答案;(3) 所处理的数据能有较好的存储和维护性能.

显然,这 3 个条件对范围查询处理是非常重要的.然而一般来说,应用预计算技术的数据立方体的存储和维护性能较差;近似技术一般不能提供准确的查询结果;而以往提出的联机聚集(有时也称为渐进查询)技术往往需要可观的附加存储空间用于存储多个粒度的预计算结果.这样就影响了数据立方体的存储和维护性能.

2 PE:范围查询的渐进估算算法

在本节中,我们提出用于范围查询处理的渐进估算 PE 算法.

概括地说,PE 首先为范围查询提供一个最粗粒度的概率估计,然后对这个估计结果不断地进行修正,直到这个估算过程被用户终止或者完成最细粒度的估计(此时,估算结果已经是准确值).

其思路是,对于一个针对 d 维数据的范围查询 $q=sum(l_1:h_1,l_2:h_2,...,l_d:h_d)$,我们从 QC-Tree 的根节点开始聚集估算(其中 q 中的 $l_i:h_i$,是指该查询在 i 维上的范围是从 l_i 到 h_i).首先,在根节点上获得对范围查询结果的第 1 次估算;随后,对根节点每一个维标号为 1 的孩子节点进行下一个层次的估算,并据此修正原先的估算;然后,对这些孩子节点的维标号为 2 的孩子节点进行估算.重复这个过程,直到最后一个维(此时也到达了叶子节点),最终得到的结果将是一个精确值.在查询过程中,为提高效率,可以利用 QC-Tree 的特点对查询点进行剪枝,其原理与 QC-Tree 的查询处理相同.

对于一个维标号为k、维值为d_k的节点c,我们在节点c上的估算结果为

$$\hat{q}=sum(d_1,...,d_i,*,...,*)\prod_{i=k+1}^{d}\frac{sum(d_1,...,d_i,*,...,*,l_i:h_i,*,...,*)}{sum(d_1,...,d_i,*,...,*)}=\frac{\prod_{i=k+1}^{d}sum(d_1,...,d_i,*,...,*,l_i:h_i,*,...,*)}{sum(d_1,...,d_i,*,...,*)^{d-k-1}}.$$

下面给出 PE 算法的基本过程.其中,*queueNode* 是一个先入先出的、指向节点的指针的队列,将要在这些指针指向的节点上进行估算.另外,*queueValue* 是与之相对应的队列,保存着过去对这些节点在较粗粒度级别上的估算值.在 PE 算法的第 4 步,我们从队列 *queueNode* 中获得一个节点 *currN*,从队列 *queueValue* 中获得过去对这个节点在较粗粒度级别上的临时估算值 *currV*(在以后的第 13 步,我们将获得比 *currV* 更精确的估算);在第 5 步,

如果 *currN* 是一个叶子节点,那么 *currV* 就是准确的,我们不需要对 *currN* 进行更精确的估算;第 7~第 11 步计算 *rate* 的值,其中第 9 步中,*temp* 的值就是 *currN* 的所有维标号为 *i*、维值在(l_i:h_i)的孩子节点上的值的总和.假设 *currN* 的维标号为 *k*,那么在执行第 12 步之前,*rate* 的值是 $\frac{\prod_{i=k+1}^{d} sum(d_1,...,d_i,*,...,*,l_i:h_i,*,...,*)}{sum(d_1,...,d_i,*,...,*)^{d-k-1}}$;在第 12 步中,对于 *currN* 的每个维标号为 *next*、维值在(l_{next}:h_{next})的孩子节点,我们将其放入队列 *queueNode*,并将对这个孩子节点的聚集值×*rate*×*currN.valu*/*temp* 后放入队列 *queueValue*.这个放入队列 *queueValue* 的值,也就是我们在当前的粒度级别对这个孩子节点所覆盖的聚集值的临时估算;在第 13 步,q'的值获得了修正.值得注意的是,由于有了队列 *queueValue* 的存在,我们实际上在对每个节点进行估算后,就可以修正 q'的值,而不是将同一个粒度级别的所有节点进行估算后才能对 q'进行修正.

PE 算法.

输入:QC-Tree 和范围查询 $q=sum(l_1:h_1,l_2:h_2,...,l_d:h_d)$.

输出: $\hat{q}$.

Method:

1. 清空 *queueNode* 和 *queueValue*; $\hat{q}$ =*root.value*; /*初始化*/
2. *queueNode.push*(*root*);*queueValue.push*(*root.value*);
3. while(*queueNode* 不为空){
4. *currN*=*queueNode.pop*(); *currV*=*queueValue.pop*();
5. if(*currN* 不是叶子节点){
6. *next*=*currN* 的维标号+1;
7. *rate*=1;
8. for(*i*=*d*;*i*≥*next*;*i*--){
9. $temp=\sum_{child_i \in l_i:h_i} child_i.value;\ rate=rate\times\frac{temp}{currN.value}$;
10. if(*rate*==0) break;
11. };
12. 将 *currN* 的每个维标号为 *next*、维值在(l_{next}:h_{next})的孩子节点放入队列 *queueNode*;将其中的每个孩子节点的聚集值×*rate*×*currN.valu*/*temp* 放入队列 *queueValue*;
13. $\hat{q}=\hat{q}$ –*currV*+*rate***currN.value*;输出 $\hat{q}$;
14. }; };
15. 返回 $\hat{q}$; /*此时, $\hat{q}$ 是准确的查询结果*/

2.1 估算过程中估算值的精度

在这里,我们暂时假定数据立方体中的数据是非负的.这样,我们可以将数据立方体中度量值的 *sum* 聚集看作在由维所确定空间的特定区域中点的计数.例如,可以将 $sum(l_1:h_1)=m$ 的含义解释为在区域 l_1:h_1 中有 *m* 个点.

首先,我们考虑在数据立方体中只有两个统计独立的维 *A*,*B* 的情况.假定我们已经知道 *sum*(*)的值为 *n*.如果 *x* 表示某点的 *A* 维在 l_1:h_1 范围内,*y* 表示该点的 *B* 维在 l_2:h_2 范围内,*z* 表示该点的 *A* 维在 l_1:h_1 范围内,并且 *B* 维在 l_2:h_2 范围内(即该点在区域(l_1:h_1;l_2:h_2)中),那么,*z*=*xy*.

设 *Z* 为 *n* 个独立点 *z* 值的总和,那么,*Z* 将服从二项式分布 *B*(*n*;*P*(*x*)*P*(*y*)),则对 *Z* 的估算为

$$E(Z)=sum(*,*)P(xy)=nP(x)P(y)=\frac{sum(l_1:h_1,*)sum(*,l_2:h_2)}{sum(*,*)}.$$

为了考察这种估算的准确性,我们观察到 $D(Z)=sum(*,*)P(z)(1-P(z))$.

根据 Chebyshev 不等式,我们可以得到

$$P(|Z-E(Z)|\geq\varepsilon)\leq\frac{D(Z)}{\varepsilon^2}=\frac{sum(*,*)P(z)(1-P(z))}{\varepsilon^2} \quad (1)$$

这个结果可以很容易地扩展到更高维的情况.

在前面,我们假定维之间是相互独立的.然而在现实世界中,数据的维常常是相互关联的.正是在这种情况下,我们的估算结果是逐步靠近精确值的.原因在于:在较细粒度上的估算所需要的前提条件更少,从而可以对过去较为粗略的结果进行修正.由于篇幅所限,我们省略了具体的推算过程.一般来说,QC-Tree 的维按照相关度逆序排列,将有助于提高 PE 算法的质量.

PE 算法计算到第 g 个维时的时间复杂度 $Cost_{pe}(g)$和普通算法结算完毕的时间复杂度 $Cost_{normal}$ 在数据极为稠密时可以分别看成

$$Cost_{pe}(g)=\sum_{k=1}^{g}\left[\prod_{i=0}^{k-1}\left(\sum_{j=k}^{d}m_j\right)\right];\ Cost_{normal}=\sum_{k=1}^{d}\prod_{i=1}^{k}m_i,$$

其中,d为维的总数;m_i为第i维中在查询范围内的维值的个数,即$m_i=h_i-l_i+1$,且定义$m_0=1$.虽然上式只有在数据极为稠密时才是准确的,但仍然能反映出PE算法的时间消耗与结果精度之间的关系.虽然PE算法的总时间复杂度大于普通算法,PE算法可以随时输出估算值而不需要等待全部计算完成,而当g较小时,所需时间要少得多.

算法所需的空间主要是队列 *queueNode* 和 *queueValue* 所占用的临时空间.这个空间在 PE 算法计算到第 g 个维时最多约为 $2\times\prod_{i=1}^{g+1}m_i$,也就是在查询范围内需要在第 g+1 维进行估算的节点的个数乘以 2.

对 PE 算法计算到第 g 个维时的精度,可以采用如下方法进行判断:若 $V(g)$表示 PE 计算到第 g 个维时输出的估算,则此时的误差为 $Err(g)=V(g)\times|[V(g-1)-V(g)]/V(g)|^{d-g-1}$.

值得注意的是,我们的算法对度量的要求为所有数据不小于 0,或者所有数据不大于 0.对于更一般的情况,可以用以下方法解决:将原先的数据立方体按照度量值分为两部分——正数和负数.这样,每一个部分都可以看成是一个独立度量.在进行估计时,分别对两个度量进行估计,然后将估计值相加并作为最终的估算结果.由于 QC-Tree 本身是一种非常有效的数据存储结构,因此执行这种策略还是值得的.

3 HPE(混合渐进估算算法)和 MPE(多查询渐进估算算法)

在等式(1)中我们注意到,当范围查询的最终聚集值较小时,PE 的估算准确率不高.我们观察到,当这个真实值较小时,被这个范围查询所覆盖到的节点数通常也比较少.此时,由于普通的查询处理算法(深度优先搜索)的查询速度与范围查询所覆盖到的节点数成正比,所以普通查询处理算法将能够比较快地返回查询结果.也就是说,普通算法在真实值较小时速度快,而 PE 在真实值较大时效率高.

我们提出了一种混合型的估算算法 HPE(hybrid progressively estimate).这种算法将 PE 和普通的查询处理算法结合起来,以达到一个均衡的查询性能.HPE 在整体结构上与 PE 类似,两者的不同在于,HPE 有一个阈值.在某节点 N 上进行估算时,两算法使用的方法相同.在这个估算后,HPE 对刚刚估算出的 N 节点所覆盖的聚集值 V 进行判断,以确定是否要对 N 的子节点继续进行估算.如果 V 比阈值小,我们就不需要对 N 的子孙节点进行估算,而是直接用普通的方法在 N 上进行聚集计算;否则,如果 V 比阈值大,就把 N 的相关子节点放入队列,进行以后的估算.作为 PE 和普通算法的结合体,HPE 的整体性能更为均衡和合理.当查询结果较小时,HPE 的性能与速度较快的普通算法相差不多;而当查询结果较大时,HPE 又与效率较高的 PE 类似.第 4.3 节的实验结果也证实了这一点.

在本节中,我们还将考虑同时对多个范围查询进行估算的问题.多个同时进行的范围查询往往在大多数维上的范围是相同的.例如,多范围查询常常以如下形式出现:$sum(l_1:h_1,...,\{l_{g1}:h_{g1},l_{g2}:h_{g2},...,l_{gn}:h_{gn}\},...,l_d:h_d)$,这里,除了第 g 个维的范围不相同以外,其他维上的范围都是相同的.我们称 g 维为“group-by 维”.

我们提出了 PE 的多查询版本 MPE(multiple progressively estimate)算法来处理这样的多范围查询问题.在实际的处理过程中,PE 的相当一部分工作是计算估算中所用到的 rate.而 MPE 算法的思路是同时对这多个范围

查询进行估算,在估算过程中,共享临时计算出的rate值.这样,MPE算法可以减少多范围查询估算中的大量重复工作,从而提高处理的效率,使得查询的速度比分别对这些查询进行处理要快得多.

由于篇幅关系,我们在这里将不给出HPE和MPE的详细算法描述.

4 算法性能测试实验

4.1 实验设置

实验的软、硬件环境是,CPU型号为Pentium 4,主频为2.4G,256M内存,操作系统为Windows XP.

实验使用了模拟数据和真实数据.其中,模拟数据包括6个维、3个度量,每个维的秩均为100.与文献[1]类似,模拟数据符合Zipf分布,参数为2,维之间没有关联.底层数据共有1 000 000条记录.真实数据选用了1985年9月陆地上的天气情况数据[2],该数据也出现在了文献[1,4]等许多相关文章中.该数据包含了1 015 367条记录,选用了6个维和3个度量.每个维的含义及其秩为:时间(24)、经度(352)、纬度(152)、海拔(179)、当前气候(101)、气候改变码(10).

4.2 PE的查询性能实验

在这组实验中,本文将PE的查询处理性能与文献[1]中的普通算法进行了比较.为了检测范围查询的效果,我们随机生成1 000个范围查询.对于模拟和真实数据,一个范围查询包括有6个维,每个维的范围是随机选出的.我们将相对误差定义为误差与精确值的比,即|精确值–估算值|/精确值×100%.在估算过程中,如果相对误差的值在某一时刻t以后一直小于某一阈值(一般为0.05),则我们认为在t时刻我们获得了对查询的足够准确的估计.为方便起见,当查询精确结果为0时,我们将相对误差从无穷大调整为300%.

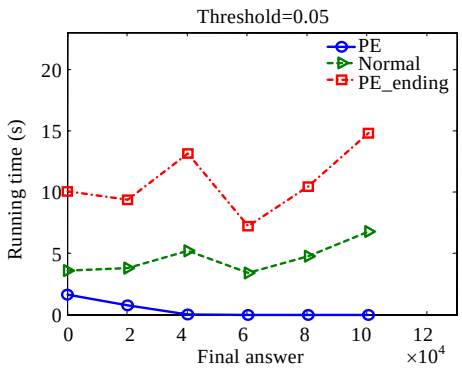
在模拟数据集上的实验结果显示在图2(a)中,在真实数据集上的实验结果显示在图2(b)中.图中的x轴表示查询的精确值,y轴表示最终结果的返回时间(普通算法)或获得足够准确的估算值所需的时间.

图2(a)和图2(b)的结果显示,尽管PE算法的完全计算(计算出精确值)所花费的时间(图中为PE_ending)比普通算法(图中为normal)要长,但是PE在运行过程中获得足够准确的值的时间(图中为PE)明显少于普通算法.而且,聚集值越大,PE算法的优势就越明显.另外,图2(b)中,当聚集值非常大时,几种方法的查询时间均有所下降的原因是,此时QC-Tree的查询剪枝处理作用的结果.

4.3 HPE和MPE的查询性能实验

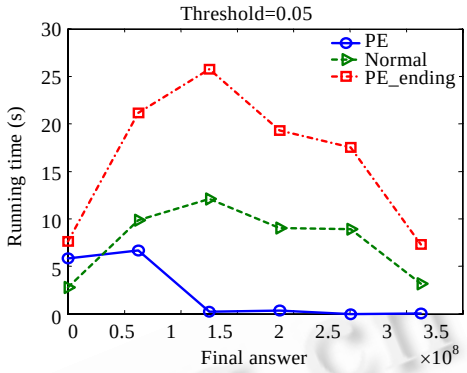
在HPE的实验中,对于每类数据,我们也都生成了1 000个范围查询.实验结果显示在图2(c)中.实验结果显示,HPE能够显著提高渐进算法在聚集值较小时的速度,同时,当聚集值较大时,HPE的性能与PE基本相同.因此,HPE的整体性能更为稳定.此外,HPE算法的完全运行时间要少于PE算法的完全运行时间.

对于每个数据集,我们使用不同的group-by维来检测MPE的性能.对于每个group-by维,我们随机生成100个多范围查询,并分别使用MPE和普通算法进行处理.实验结果显示在图2(d)中.图2(e)和图2(f)显示了查询时间与group-by维之间的关系.实验结果说明,普通算法的平均运行时间基本上不随group-by维发生变化,而group-by维在整个维次序中排得越靠后,MPE算法的平均运行时间就越短.这是很自然的,因为group-by维排得越靠后,在MPE的查询过程中就有更多的rate被共享,这样被节省的操作就越多,MPE也就越快.



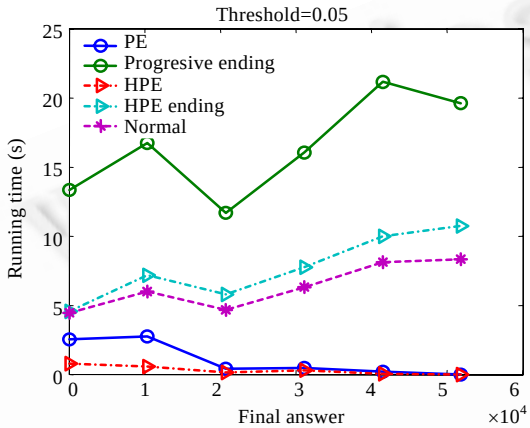
(a) PE with the synthetic data

(a) PE 模拟数据



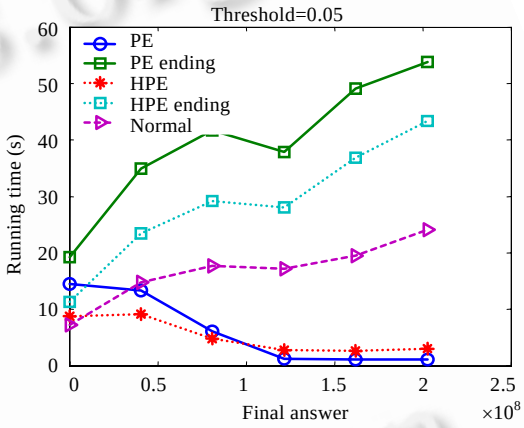
(b) PE with the real data

(b) PE 真实数据



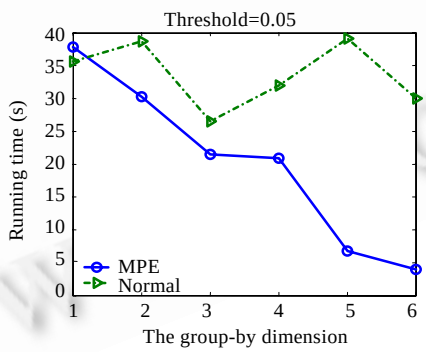
(c) HPE with the synthetic data

(c) HPE 模拟数据



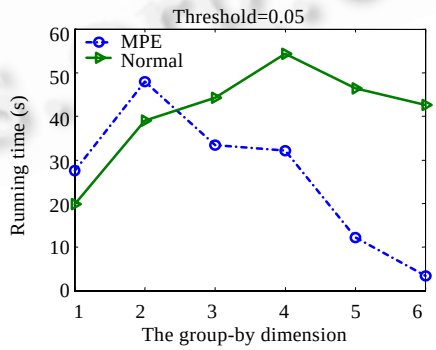
(d) HPE with the real data

(d) HPE 真实数据



(e) MPE with the synthetic data

(e) MPE 模拟数据



(f) MPE with the real data

(f) MPE 真实数据

Fig.2 Experimental results

图 2 实验结果

5 结论和进一步的工作

本文提出了在 QC-Tree 上的多种联机聚集算法.这些联机聚集算法可以明显改善 QC-Tree 的范围查询处理

性能.而且,与以往其他数据立方体的联机聚集算法不同,我们的算法不需要任何附加空间,而是利用 QC-Tree 中的保存的语义关系来估算聚集结果.性能分析及实验结果也说明,我们的联机聚集算法可以较好地改善 QC-Tree 的范围查询处理能力.

References:

[1] Lakshmanan LV, Pei J, Zhao Y. QC-Trees: An efficient summary structure for semantic OLAP. In: Halevy AY, ed. Proc. of the 2003 ACM SIGMOD Int'l Conf. on Management of Data. San Diego: ACM, 2003. 64−75.

[2] Hahn CJ, Warren SG, London J. Edited synoptic cloud reports from ships and land stations over the globe, 1982−1991. 1996. http://cdiac.est.ornl.gov/ftp/ndp026b/SEP85L.Z

[3] Hellerstein JM, Haas PJ, Wang HJ. Online aggregation. In: Peckham J, ed. Proc. of the 1997 ACM SIGMOD Int'l Conf. on Management of Data. Tucson: ACM Press, 1997. 171−182.

[4] Wang W, Lu HJ, Feng JL, Yu JX. Condensed cube: An efficient approach to reducing data cube size. In: Proc. of the 18th Int'l Conf. on Data Engineering. San Jose: IEEE Computer Society, 2002. 155−165.

[5] Sismanis Y, Deligiannakis A, Roussopoulos N, Kotidis Y. Dwarf: Shrinking the PetaCube. In: Franklin MJ, Moon B, Ailamaki A, eds. Proc. of the 2002 ACM SIGMOD Int'l Conf. on Management of Data. Madison: ACM Press, 2002. 464−475.

[6] Lakshmanan LV, Pei J, Han JW. Quotient cube: How to summarize the semantics of a data cubes. In: Bressan S, Chaudhri A, Lee ML, Yu JX, Lacroix Z, eds. Proc. of the 28th Int'l Conf. on Very Large Data Bases. Hong Kong: Springer-Verlag, 2002. 778−789.

[7] Ho CT, Agrawal R, Megiddo N, Srikant R. Range queries in OLAP data cubes. In: Peckham J, ed. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. Tucson: ACM Press, 1997. 73−88.

[8] Poon CK. Dynamic orthogonal range queries in OLAP. Theoretical Computer Science, 2003,296(3):487−510.

[9] Acharya S, Gibbons PB, Poosala V, Ramaswamy S. The aqua approximate query answering system. In: Delis L, Faloutsos C, Ghandeharizadeh S, eds. Proc. of the 1999 ACM SIGMOD Int'l Conf. on Management of Data. Philadelphia: ACM Press, 1999. 574−576.

[10] Chakrabarti K, Garofalakis MN, Rastogi R, Shim K. Approximate query processing using wavelets. The VLDB Journal, 2001, 10(2−3):199−223.

[11] Lazaridis I, Mehrotra S. Progressive approximate aggregate queries with a multi-resolution tree structure. In: Aref WG, ed. Proc. of the 2001 ACM SIGMOD Int'l Conf. on Management of Data. Santa Barbara: ACM Press, 2001. 401−412.

[12] Riedewald M, Agrawal D, Abbadi AE. pCube: Update-Efficient online aggregation with progressive feedback and error bounds. In: Gunther O, Lenz HJ, eds. Proc. of the 12th Int'l Conf. on Scientific and Statistical Database Management. Berlin: IEEE Computer Society, 2000. 95−108.

[13] Schmidt RR, Shahabi C. How to evaluate multiple range-sum queries progressively. In: Popa L, ed. Proc. of the 21st ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems. Madison: ACM Press, 2002. 133−141.

李红松(1973－),男,河南渑池人,博士生,主要研究领域为数据仓库,联机分析处理,数据挖掘.

黄厚宽(1940－),男,教授,博士生导师,CCF 高级会员,主要研究领域为人工智能,机器学习.