ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.7, July 2009, pp.1785−1799
doi: 10.3724/SP.J.1001.2009.03350

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

压缩数据库中一种自适应直方图的构建*

骆吉洲+, 李建中, 王宏志

(哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

Construction of an Adaptive Histogram in Compressed Database

LUO Ji-Zhou+, LI Jian-Zhong, WANG Hong-Zhi

(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

+ Corresponding author: E-mail: luojizhou@hit.edu.cn

Luo JZ, Li JZ, Wang HZ. Construction of an adaptive histogram in compressed database. *Journal of Software*, 2009,20(7):1785−1799. http://www.jos.org.cn/1000-9825/3350.htm

Abstract: Histograms can be used to estimate the selectivity of queries in query optimization. It is an unsolved problem using batched queries in compressed databases to construct an adaptive histogram to optimize query processing or answer queries approximately. This paper proposes to track hot data in compressed databases by scheduling these batched queries and use the feedback in query results to accelerate the convergence speed of the constructed adaptive histogram which can be maintained incrementally. A parametric method is also proposed to estimate the tuples falling in query area which is not covered by any bucket in the histogram. Experimental results show that the adaptive histogram has more average accuracy, higher convergence speed and better adaptability than STHoles.

Key words: adaptive histogram; compressed database; parametric method; convergence speed

摘 要: 直方图在查询优化过程中起着重要作用.在压缩数据库中利用查询处理的特点构建自适应直方图以便于查询优化或近似回答查询是尚待解决的问题.通过对查询缓冲池内的查询进行调度来追踪热点数据,并用查询结果中的反馈信息构建自适应直方图以加快自适应直方图的收敛速度.另外,还提出一种参数化方法来估计未被任何桶覆盖的区域中元组的个数.该直方图可以增量式地被维护.实验结果表明,这种直方图具有良好的平均精度、更快的收敛速度和更强的自适应能力.

关键词: 自适应直方图;压缩数据库;参数化方法;收敛速度

中图法分类号: TP311 **文献标识码**: A

信息技术的发展,使得高于 10^{12} 字节的海量数据层出不穷.为了有效管理海量数据,人们提出了压缩数据库技术[1−5].压缩数据库技术可以提高海量数据的存储效率,也是提高数据库性能的重要途径.目前,压缩数据库技术的研究主要包括适应于数据库随机存取特征的数据压缩方法[3,4]和压缩数据上的操作算法[1,2]两个方面,有关压缩数据库上的查询优化技术[4]的研究很少见.文献[3]给出的查询优化技术在传统的查询优化技术中增加了

* Supported by the National Natural Science Foundation of China under Grant Nos.60533110, 60773068 (国家自然科学基金); the National Basic Research Program of China under Grant No.G1999032704 (国家重点基础研究发展计划(973))

Received 2007-09-29; Revised 2008-01-29; Accepted 2008-04-15

对解压缩时机的选择操作,但并未讨论压缩数据库中查询选择性估计这一问题.本文讨论压缩数据库查询优化中的查询选择性估计问题.

在查询计划生成时,常用直方图来估计查询的选择性,而估计的精确程度将影响连接、选择等基本关系代数操作的执行顺序.在压缩数据库中还将影响到解压缩时机的选择,最终影响查询计划的生成.此外,在特定应用中查询不需要精确结果时,直方图的好坏还将影响近似回答查询的精确程度.直方图在查询处理过程中的重要作用已经得到广泛认识[6].

绝大多数商用数据库系统在关系上维护一个或多个直方图[6,7].建立直方图要对数据集进行扫描、抽样、排序,再将数据划分到桶中.这种直方图建立后将在一段时间内保持不变(无论数据集是否发生变化),因而是静态的.对静态直方图的维护策略是,等到直方图对查询选择性的估计误差达到阈值后(或周期性地)重建,其代价极高.在压缩海量数据方面,还需考虑重建直方图时对数据的解压操作,批量数据的压缩、加载使得海量数据集上数据分布频繁变化,这使得上述策略在压缩数据上更加不可行.

自适应直方图可以减小在海量数据上建立和维护直方图的代价,并得到了广泛的研究[8,9],其基本方法是用查询反馈来建立和维护直方图,其最大特点是建立和维护代价独立于数据集的大小.这恰好是我们在压缩海量数据上建立直方图时所希望具备的性质.

另一方面,由于压缩海量数据库中处理查询的时间相对较长,查询缓冲池内可能滞留多个未处理查询.尽管数据库中的查询相互之间相对独立,但压缩海量数据库中存储的数据覆盖了很长时间或很大空间.一个查询往往只对数据集的某个子集进行查询,并且在相对短的时间段内,后到来的查询很可能再访问先前查询访问过的数据.这样,仅有部分数据被频繁访问.一段时间内被频繁访问的数据称为热点数据.我们希望直方图能够跟踪热点数据,即在直方图中,热点数据的信息被维护得更加详细、准确,且当热点数据转移时,直方图能够根据查询反馈及时反映这种转移.调度查询缓冲池内的查询并根据查询反馈来建立和维护高质量的直方图,使其快速反映数据集和用户查询兴趣的变化,这正是本文研究的问题.

本文提出了查询的在线调度方法来加速满足上述要求的直方图准确度的收敛速度.该方法使得自适应直方图能够迅速、准确地估计查询选择性,快速适应海量压缩数据的批量加载和用户查询兴趣的转移.我们的直方图由于追踪热点数据而不能覆盖整个数据空间.当访问冷数据的查询到来后,如果未被直方图覆盖的数据空间内数据被看成是均匀分布的,则直方图对查询选择性的估计非常不准确.这种现象在压缩海量数据批量加载完成时非常严重.针对这一问题,我们提出一种参数化方法来估计查询选择性,其中的参数可以增量式地计算和维护.我们提出的自适应直方图既适用于压缩海量数据库中查询选择性估计,也适用于在任何数据分布和查询兴趣频繁变化的数据集上估计查询选择性.

本文第 1 节讨论相关工作.第 2 节给出预备知识.第 3 节给出直方图建立和维护算法.第 4 节用所建立的直方图和参数化方法来估计查询的选择性.最后是实验结果和结论.

1 相关工作

人们已经提出了很多方法来估计查询的选择性,包括抽样方法[10−12]、直方图方法[8,9,13−15]、参数方法[16].元组抽样方法[10,11]在查询到来后,在关系上均匀地抽样,然后根据样本大小和样本中数据的分布来估计查询选择性.如果将这样的方法用到压缩数据库中,则在每个查询到来之后均要对压缩数据进行解压缩,操作代价极高.参数方法利用一定个数的参数函数来近似拟合数据的分布函数.有的参数方法采用一般的多项式函数并用最小二乘法来计算参数;有的参数方法用线性函数的组合作为分布函数.在这种方法中,分布函数中的参数往往用查询提供的反馈信息来更新.参数方法最大的问题在于,很难找到通用分布函数来刻画实际数据分布.在压缩数据库中,这样的问题仍然存在.

直方图方法通过建立静态或动态直方图来拟合数据分布,文献[6]综述了建立直方图的各种方法.静态直方图构建方法被广泛研究,如基于小波的直方图[17]、*V-optimal*(*F*,*F*)直方图族[18]和 *V-optimal*(*V*,*F*)直方图族[19].如本文开始部分所述,这种直方图(在压缩数据库中)建立和维护的代价极高.

自适应直方图利用查询反馈建立和维护直方图,其建立和维护代价独立于数据集的大小.这一特征使得它非常适合于压缩海量数据库.这种方法可以发现数据分布不均匀的桶并将它分裂成若干个桶,同时将相邻的小桶合并,以便将有限的空间用到临界数据区域.STGrid 方法[3]利用查询的反馈信息来改善基于网格方法的直方图.这种方法对数据划分过于生硬,数据分布中存在的不规则小聚类将导致无用的桶.Lim 等人提出了一种基于查询反馈建立和维护的自适应直方图 SASH(a self-adaptive set of histograms)[7].

Bruno 等人在文献[9]中提出了一种称为 STHole(self-tuning holes)的自适应直方图.它是一种很有代表性的自适应直方图. STHoles 假设初始直方图为空.一旦一个查询被执行,就可以从其返回结果中抽取统计信息,用以识别现有直方图的桶与查询相交的部分数据分布是否符合均匀分布.如果发现某个小区域中数据分布很不均匀,就在现有的桶中挖一个洞,建立新桶并修改相关的桶.这种方法不需要系统地扫描关系中所有的数据,其建立和维护的代价与数据集合的大小无关.STHole 用嵌套桶结构与查询区域的相交情况和查询的反馈来找出数据分布不均匀的区域,这将有助于我们跟踪热点数据.本文的工作将基于这种嵌套的桶结构.但由于这种直方图不能覆盖整个数据空间,在未被直方图覆盖的区域内对查询选择性的估计不准确.它也不考虑当前查询缓冲池内查询的调度问题,使其精确度的收敛很慢.

2 预备知识

将数据库中具有 n 个属性的关系 R 表示成 $R(X_1,\ldots,X_n)$,不区分关系模式和关系实例.关系 R 中的一个元组记为 r.对于 i=1,…,n,将属性 X_i 的值域记为 D_i.不失一般性,假设对任意 i,值域 D_i 是可数的并用整数编码.将属性 X_i 在关系 R 中的实际取值集合记为 V_i,即 $V_i=\{x\in D_i|\exists r\in R$ 使得 $r.X_i=x\}$.$R(X_1,\ldots,X_n)$确定了一个多维数据空间 $D=D_1\times\ldots\times D_n$.下面首先讨论 D 中超方体,然后给出直方图的概念.

定义 1. 给定 $v_l=(v_l^{(1)},...,v_l^{(n)})\in D$, $v_u=(v_u^{(1)},...,v_u^{(n)})\in D$ 且 $v_l^{(i)}\le v_u^{(i)}$ 对 i=1,…,n 均成立,则称 $B=\{(x_1,\ldots,x_n)\in D|v_l^{(i)}\le x_i\le v_u^{(i)},i=1,\ldots,n\}$为 n 维数据空间 D 中的一个超方体,记为 $B\langle v_l,v_u\rangle$.

定义 2. 超方体 $B\langle v_l,v_u\rangle$的体积定义为 $vol(B\langle v_l,v_u\rangle)=\prod_{i=1}^{n}(v_u^{(i)}-v_l^{(i)})$.

定义 3. 一个区域查询 q 由一个超方体 $B\langle v_l,v_u\rangle$构成,其查询结果为$\{r\in R|v_l^{(i)}\le r.X_i\le v_u^{(i)}$ 对 i=1,…,n 成立$\}$.超方体 $B\langle v_l,v_u\rangle$称为查询 q 的查询区域.

为方便计,用 B_q 表示区域查询 q 对应的超方体,用 q_B 表示超方体 $B\langle v_l,v_u\rangle$对应的区域查询.

超方体 B 和 B'之间的拓扑关系有包含($B\subseteq B'$或 $B'\subseteq B$)、不相交($B\cap B'=\varnothing$)和部分重叠($B\cap B'\neq\varnothing$但不无包含关系).如果 $B'\subseteq B$,则称 B'嵌套在 B 中.给定一列超方体 $B_1,\ldots,B_w$,可以构造一棵树来反映这些超方体间的拓扑关系.这棵树满足如下条件:(1) 根节点是整个多维数据空间 D;(2) 任意孩子节点均嵌套在其父亲节点中;(3) 任意兄弟节点不互相嵌套.如果将查询缓冲池内区域查询对应的超方体组织在这样一棵树中,就可以判断哪些查询访问了热点数据并对查询执行顺序加以调度,以改善要构建的直方图的性能.

定义 4. 对于超方体 $B\langle v_l,v_u\rangle$,$|\{r\in R|v_l^{(i)}\le r.X_i\le v_u^{(i)},i=1,\ldots,n\}|$称为超方体 B 的频率,记为 f_B.

定义 5. 对于超方体 B 和 B',如果 $B'\subset B$ 且不存在 $B''\subset B$ 使得 $B''\subset B'$,则称 B'是 B 的直接子超方体,B 的所有子超方体构成的集合记为 $child(B)$.

定义 6. 超方体 B 的净体积定义为 $vol_{net}(B)=vol(B)-\sum_{B'\in child(B)}vol(B')$,净频率定义为 $f_{net}(B)=f_B-\sum_{B'\in child(B)}fB'$.

定义 7. 对于净频率为 $f_{net}(B)$的超方体 B,称 $B\backslash(\cup_{B'\in child(B)}B')$为由超方体 B 诱导的桶,超方体 B 的净频率称为该桶的频率,超方体 B 的净体积称为该桶的体积.通常,仍然用 B 来表示一个桶.由 $child(B)$中超方体诱导的桶称为桶 B 的子桶.桶 B 的数据密度定义为 $d(B)=f_{net}(B)/vol_{net}(B)$.

简单地说,桶是有洞的超方体.由于超方体的外边界和桶的外边界一致,故用同一个记号来表示.

定义 8. 一系列互不相交的桶 $B_1,\ldots,B_m$ 以及相应的频率值构成的集合 $H=\{\langle B_i,f_{net}(B_i)\rangle,i=1,\ldots,m\}$称为关系 R 上的一个直方图.

直方图有很多类型,它的桶可能构成了整个数据空间的划分、覆盖或者部分覆盖.

直方图建立后,人们通常假设数据在直方图每个桶内均匀分布,然后估计关系 R 中满足查询 q 的元组个数.用记号 $est(H,q)$表示用直方图 H 估计得到的满足查询 q 的元组个数,则 $est(H,q)$可如下计算:

$$est(H,q)=\sum_{B\in H}\frac{vol_{net}(B\cap B_q)}{vol_{net}(B)}f_{net}(B) \tag{1}$$

对于自适应直方图,文献[9]提出了衡量其性能的度量.设 W 是一个由$|W|$个查询构成的查询序列,则可以用如下的均值绝对误差来衡量直方图的精确度:

$$E(R,H,W)=\frac{1}{|W|}\sum_{q\in W}|est(H,q)-act(R,q)| \tag{2}$$

其中,$act(R,q)$是执行查询 q 后得到的数据集 R 中实际满足查询 q 的元组个数.

在 STHoles 中,直方图的所有桶并不一定覆盖整个数据空间.因此,作者假设在未被覆盖的数据区域内数据是均匀分布的,由此得到关系 R 中满足查询 q 的元组个数的一个估计值 $est_{uni}(R,q)$.并由此建立了一个衡量直方图精确度的如下度量:

$$E'(r,H,W)=\frac{E(R,H,W)}{E_{uni}(R,W)} \tag{3}$$

其中,$E_{uni}(R,W)=\frac{1}{|W|}\sum_{q\in W}|est_{uni}(R,q)-act(R,q)|$.

为直观计,本文下面的讨论中假设关系 R 的维数等于 2,但所讨论的方法均适用于多维数据空间.

3 压缩数据库中自适应直方图的建立

本节首先建立两个度量:一个用来衡量查询的热度,另一个用来衡量某数据区域的数据密度与其周围区域数据密度的差异.然后,根据这两个度量和已经执行的查询以及查询缓冲池内未被执行的查询提供的信息来对查询进行调度,以加速所建立的自适应直方图精确度的收敛.

3.1 查询调度中的两个度量

定义 9. 查询 q 的查询区域 B_q 与其他查询的查询区域相交,B_q 被划分成体积分别为 $s_1,\ldots,s_k$ 的 k 个互不相交的部分,其中体积为 s_i 的部分被 n_i 个查询(包括 q)覆盖,定义查询 q 的热度 $hot(q)=\sum_{i=1}^{k}s_i\times n_i$.

查询的热度综合反映了查询区域的大小和用户对该区域内数据感兴趣的程度.如果一个查询的查询区域不与其他查询的查询区域相交,则该查询的热度即为该查询的查询区域的大小;如果多个查询的查询区域相交,则查询的热度等于相互重叠的子区域体积的加权和,权值的大小即为覆盖该子区域的查询的个数.显然,在查询调度时应该选择热度值较大的查询先执行,这样可以尽可能地将其他查询要访问的数据区域的数据分布状况刻画出来反映到直方图中,在用直方图来估计其他查询的选择性时就会更加精确,从而加快自适应直方图精确度的收敛速度.

然而,查询的热度值随着新查询的到来会频繁变化.这对于已完成执行的查询来讲是无关紧要的;对于未处理的查询,动态维护查询的热度值将影响到对该查询的执行调度.关于查询热度值的动态维护,我们将在第 4.2 节中加以讨论.关于热度值,定理 1 将有助于第 4.3 节中建立辅助结构以加速查询调度过程.

定理 1. 设 q 和 q'是两个查询,且 $B_q\supseteq B_{q'}$,则 $hot(q)\geq hot(q')$.

定义 10. 超方体 B_1 和 B_2 满足 $B_1\subseteq B_2$,gap 值定义为

$$gap(B_1,B_2)=\frac{d(B_1)}{d(B_2)}=\frac{f_{net}(B_1)}{vol_{net}(B_1)}\times\frac{vol_{net}(B_2)}{f_{net}(B_2)} \tag{4}$$

直观地讲,$gap(B_1,B_2)$是两个区域的密度差异,刻画了两个区域数据分布不均匀的程度.gap 值的取值范围是$(0,+\infty)$.$gap(B_1,B_2)=1$ 表示数据区域 B_1 的数据分布密度与其周围区域的数据密度大致一样,没有必要用两个桶来刻画该区域的数据分布;$0<gap(B_1,B_2)<1$ 表示数据区域 B_1 的数据分布密度小于其周围区域的数据密度,gap 值越小,表明数据分布越不均匀;$1<gap(B_1,B_2)<+\infty$表示数据区域 B_1 的数据分布密度大于其周围区域的数据密度,gap

值越大,表明数据分布越不均匀.

3.2 查询树的构造

连续到来的查询缓存在固定大小的查询缓冲池内.先不考虑缓冲池的溢出问题(见第 4.3 节).缓冲池内所有查询被组织成一棵树,称为查询树.树中每个节点对应一个查询,记录的信息见表 1.查询树的根节点被初始化为整个数据空间.将新来的查询 q 插入查询树时,以深度优先的方式访问查询树,找出所有完整包含 B_q 的查询.同时,相应地更新这些查询的热度值.将查询 q 插入查询树 QT 的方法如算法 1 所示.对于 QT 中的节点 t,如果 $B_q \subseteq B_t$,由定义 9 知 t 的热度值将增加 $q.hot$,然后以深度优先搜索方式类似地考虑 t 的所有子节点(第 4~6 行).否则,q 作为 t 的子节点被插入 QT,同时,在这个分支上,深度优先搜索将终止,转而搜索其他分支,直到所有分支被搜索(第 7~15 行)到为止.将 q 插入 QT 作为 t 的子节点时有两种不同的情况:一是 t 有某个孩子节点 q'的查询区域 $B_{q'}$被 B_q 完全包含.这时,q'将下移作为 q 的孩子节点,同时将 q'的热度值转移给 q(第 10~12 行);二是 t 的孩子节点 q'的查询区域不完全在 B_q 中.此时,q'和 q 的重叠部分的体积将同时被追加到 $q'.hot$ 和 $q.hot$ 上(第 13~15 行).

算法 1 可能将查询 q 插入到查询树 QT 的多个位置.这是必须的,因为插入 q 可能导致多个查询的热度值发生变化.显然,算法 1 动态维护了查询树 QT 中所有查询的热度值.尽管用深度优先的方式来遍历查询树,但由于查询缓冲池内的查询数量不多,从而 QT 的规模不会很大,算法 1 是非常有效的.

Table 1 Information maintained in each node of query tree

表 1 查询树节点中维护的信息

Notation	Description	Notation	Description
id	Query ID	*rst*	Actual number of tuples in query area
B	Query area	*children*	Pointer to children node
hot	Hotness scale	*parent*	Pointer to parent node
gap	*gap* value	*next*	Pointer to the next execution candidate node
est	Estimated number of tuples in query area		

算法 1 没有讨论节点的 *gap* 值如何维护.由于只有等到查询及其父查询被执行后才能用公式(4)来计算 *gap* 值,因此 q 被插入 QT 时 *gap* 未知.为查询调度时充分利用 *gap* 值反映的数据分布状况,我们希望 q 被执行前 *gap* 值被计算.为此,初始化 QT 时将根节点的 *gap* 值置为 1;q 被插入 QT 时,q 将从其父节点继承得到一个 *gap* 值.q 被执行后,将查询结果中元组的个数记录到 $q.rst$ 中,并令 $q.gap=gap(B_q,B_q.parent)$,同时让 q 的所有孩子节点继承 $q.gap$.因此,q 被执行前,$q.gap$ 实际上已经表示了其父查询的查询区域与其祖父查询的查询区域之间数据密度的差异大小.

算法 1. *InsertQuery* /*查询树构造算法*/

输入:新到来的查询 q,查询树 QT;

输出:将 q 插入 QT 并更新相应信息后得到的查询树.

```
1.  令 t 指向 QT 的根节点;
2.  q.hot=vol(B_q);
3.  WHILE t≠NULL DO
4.      IF B_q⊆B_t THEN
5.          t.hot=t.hot+q.hot;
6.          t=t 的第 1 个未访问的孩子节点;
7.      ELSE        /*q 被插入作为 t 的孩子*/
8.          将 q 插入作为 t 的孩子;
9.          FOR t.children 中的每个节点 q′ DO
10.             IF B_q′⊆B_q THEN
11.                 q.hot=q.hot+q′.hot;
12.                 将 q′下移作为 q 的孩子;
```

13. ELSE
14. $q.hot=q.hot+vol(B_q \cap B_{q'})$;
15. $q'.hot=q'.hot+vol(B_q \cap B_{q'})$;
16. t=深度优先访问 QT 的下一个节点;
17. return QT;

3.3 查询调度

查询调度是为了由此建立直方图,使其估计一系列查询的选择性时平均误差最小.因此,先被执行的查询应能在最大程度上优化直方图,使它对未执行查询的选择性估计更加精确.调度策略基于如下观察:

观察结果 1. 为使直方图更适应用户查询兴趣的变化,应首先执行查询热度值较高的查询,因为它所在数据区域被其他查询访问的可能性更大.精确刻画该区域数据分布情况,更有可能使得其他查询的选择性被估计得更加准确.对热度值相同的两个查询,应先执行位于数据分布不均匀的区域(*gap* 值大的区域)的查询.因为在用直方图来估计查询选择性时,将每个桶内的数据看成是均匀分布的,将数据分布不均匀区域的数据分布情况刻画精确后,将使得查询选择性被估计得更加准确,从而加速直方图精度的收敛速度.

观察结果 2. 如果桶 B_1 的数据密度远小于桶 B_2 的数据密度,则桶 B_2 的数据分布情况对直方图估计查询选择性时的影响会更大.因此,先执行超方体 B_2 内的查询对加速直方图精确度的收敛速度更有利.

由观察结果和查询树,我们可以调度查询.由观察结果 1,先从查询树中选择热度值最大的 k 个查询.为快速找出这 k 个查询,在查询树 QT 中利用节点中的 next 指针维护一个线性链表 CQ(candidate query),记录当前每个分支中热度值最大的查询.初始化时,CQ 仅包含查询树的根节点.当查询 q 被插入 QT 时,如果其父节点是已被执行过的查询,则 q 被加入 CQ;否则,q 不被加入 CQ.当 q 被执行后,将 q 从 CQ 中删除,同时将 q 的孩子节点加入 CQ.由定理 1,q 的孙子节点及其后代的热度值均不超过 q 的孩子节点的热度值,故 CQ 实际上维护了 QT 的所有分支中当前热度值最高的查询.

由观察结果 2,应先执行位于数据分布最不均匀区域的查询.为此,在选出的 k 个查询中选择父节点的 *gap* 值最小的查询来执行.考虑父节点的 *gap* 值有两个原因:一是因为查询在被执行之前,其本身的 *gap* 值未知;二是父节点的 *gap* 值反映了查询所在数据区域的数据分布状况.分 3 种情况说明为什么选 *gap* 值最小的查询来执行.设 q_1 和 q_2 是两个查询,其父节点的 *gap* 值分别记为 g_1 和 g_2.

情形 1. $g_1<g_2<1$.这表明,与 q_2 相比,q_1 的父节点的数据密度与其祖父节点的数据密度的差距更大,即 q_1 所在区域的数据分布更不均匀,故选择 q_1 先执行;

情形 2. $g_2>g_1>1$.与情形 1 类似;

情形 3. $g_2>1>g_1.g_2>1$ 表明 q_2 的父节点的数据密度较其祖父节点的数据密度要小,由观察结果 2,不应选择 q_2 来执行;相反,$g_1<1$ 表明 q_1 的父节点的数据密度较其祖父节点的数据密度要大,先执行 q_1 能够更准确地刻画 B_{q_1} 内的数据分布状况.因此,选择查询 q_1 先执行.

查询被执行后,并不立即从查询树中删除.*q.est* 被置为查询执行时用直方图估计 q 的选择性得到的结果.这样做的原因是,执行过的查询仍然对找出热度值较高的查询和位于数据分布不均匀区域的查询是有用的.只有在为查询树分配的内存空间不能容纳新到来的查询时才删除这些被执行过的查询,第 4.4 节将对此进一步加以讨论.在算法 1 中,查询 q 的多个备份可能被插入到查询树的不同位置.查询调度时,它们将被看作不同的查询.但是,q 的这些备份仅被执行 1 次,且将被同时从查询树中删除.

3.4 直方图的建立及相关问题讨论

本节主要讨论直方图建立及查询调度相关的 3 个问题:一是直方图的建立问题;二是如何避免查询饥饿现象的发生;三是讨论查询缓冲池内查询的删除问题.

我们建立直方图的方法是 STHoles[9]建立直方图方法的改进.在查询调度得到查询的优化执行次序后,STHoles 建立直方图的方法仍然可用.我们的方法与 STHoles 建立直方图的方法的显著区别在于,如何在直方图

中反映整个数据空间的数据分布状况.在 STHoles 中,直方图被初始化为空;而在我们的方法中,直方图被初始化为一个覆盖整个数据空间的桶,并且假设这个桶内的数据是均匀分布的.由于数据库数据字典中记录了关系中元组的总数,初始化过程可以有效地被执行.

在很大的数据区域内假设数据是均匀分布的将导致用直方图估计查询选择性的平均误差很大.STHoles 通过假设查询流具有一定的稳定性来避免这个问题,即假设测试直方图精确性的查询不会访问训练生成直方图的查询访问过的数据区域.在压缩数据库上,这种假设是不成立的,而且一个先到来的查询可能访问了一个较大的数据区域,而后到来的一系列查询仅访问一个相对很小的数据区域,这时,直方图的精确性也很低.针对后一个问题,第 4.2 节将提出参数化方法来解决它,这是 STHoles 未讨论过的问题.此外,在合并桶时,我们采用了不同于 STHoles 的方法,这将在后面加以讨论.

当一个查询被调度执行后,需要将它从查询树中删除.否则,查询树的规模将无限增大,超出有限的内存空间并导致新到来的查询无法参与调度.但是,如果查询被调度执行后立即将它从查询树中删除(其子孙节点将上移),则查询树往往趋于一个扁平的结构,从而导致我们无法追踪热度值较高的查询,同时也无法刻画查询所在区域的数据分布状况.此外,如果用户查询兴趣频繁发生变化,我们在直方图中就会频繁地进行合并桶、插入桶等操作,数据区域内的数据分布总不能被精确地刻画,这将使得直方图精确度有所下降.为了解决上述问题,我们将执行过的查询存储在一个固定大小的队列中,当查询被执行后,其反馈结果被反映到直方图中,同时将它压入队列.如果队列满了,则根据先进先出原则取出队首的查询 q',将 q'的所有备份从查询树中删除.

若严格按照我们的调度策略来对查询树中的查询进行调度,则有些查询就不得不长时间等待或者永远不能被执行.尽管这种等待不会对直方图的精确性造成负面影响,但对发布该查询的用户是不可接受的.查询长时间得不到执行还会造成查询缓冲池的空间被长时间占用,查询饥饿问题可以采用如下方式来解决.q 到来时,系统记录其到达时间 t_1,同时为 q 指定一个最长等待时间 t_2,并允许用户修改这个时间.将当前时间与 t_1+t_2 进行比较就可以发现那些等待时间过长的查询,在调度时先执行这样的查询.在自适应直方图的环境下,这种迫于时间限制执行的查询极有可能位于不常被访问的数据区域内.用直方图估计其选择性得到的结果可能很不准确.第 4.2 节讨论的参数化方法有助于提高估计结果的精确度.

4 基于自适应直方图的查询选择性估计

本节讨论用直方图来估计查询选择性的方法.对于未完全被直方图中桶覆盖的查询,本节引入了一种参数化估计方法,讨论了参数函数的选取过程以及这些参数的动态调整方法.

4.1 选择性估计算法

在利用直方图和公式(1)来估计查询选择性时,要假设直方图的每个桶内数据都是均匀分布的.然而,STHoles 的基本想法是通过在桶内挖“洞”来反映桶内数据分布不均匀的状态.当一个覆盖较大区域的桶有很多子桶,并且查询区域距某些子桶非常近时,这些子桶提供了查询区域附近的数据分布.能否利用这些信息来估计查询的选择性,是一个非常有意义的问题.另一种情况是,自适应动态直方图随着查询的到来和执行将会对桶进行增、删、改等一系列操作,在一段时间之后,数据空间内某个子区域可能未被直方图的任何桶覆盖.这时,只能通过直方图中桶提供的数据分布信息来估计位于该子区域内的查询的选择性.我们提出了一种参数方法,利用直方图中桶提供的数据分布信息建立参数函数作为数据密度的分布函数,并结合传统的方法来估计查询的选择性.

由于直方图 H 中的桶存在嵌套关系,有的桶不是其他任何桶的子桶,这样的桶称为极大桶.当访问 H 中的桶时,将从它的极大桶开始.由直方图的构造过程,H 的极大桶互不相交.

给定查询 q,现在考虑如何用直方图 H 来估计 q 的选择性.依次考虑 H 的每个极大桶 B,将超方体 B 和 B_q 求交集.对于 $B\cap B_q$,调用算法 $EstInBucket(B,B\cap B_q)$来估计 $B\cap B_q$ 中的元组个数;对于未被任何桶覆盖的查询区域,用参数方法来估计其中的元组个数.将所有估计结果相加即得到 B_q 中元组个数的估计值.整个过程如算法 2 所示.其中,$\varphi(x,y)$将在第 4.2 节中给出.

算法 2. *EstSelectivity* /*查询树构造算法*/

输入:查询 q,直方图 H;

输出:用直方图 H 估计查询 q 的选择性所得到的结果.

1. E=0;
2. FOR 直方图 H 的每个极大桶 B DO
3. $q_B=B_q\cap B$;
4. $E=E+EstInBucket(B,q_B)$;
5. $q=q-q_B$;
6. $E=E+\iint_q \varphi(x,y)\mathrm{d}x\mathrm{d}y$;
7. return E;

由于 H 中桶的嵌套结构,在桶 B 内估计 q 内的元组数通过递归算法 $EstInBucket(B,q_B)$来实现,如算法 3 所示.对于桶 B 的每个直接子桶 $B'\in child(B)$,先求 B_q 与超方体 B'的交 $B_q\cap B'$,然后递归调用 $EstInBucket(B',B_q\cap B')$来得到一个估计值.对于未被任何子桶覆盖的区域,将其体积与桶 B 的净体积相除得到一个覆盖率 c_r.如果 c_r 大于预先指定的阈值 *threshold*,则用传统方法估计剩余部分中的元组数;否则,用参数函数 $\varphi(x,y)$进行估计.

算法 3. *EstInBucket*.

输入:直方图 $\mathcal{H}$的桶 B,查询 q 且 $B_q\subseteq B$;

输出:在桶 B 内估计查询 q 的选择性所得到的结果.

1. E=0;
2. FOR B 的每个直接子桶 B' DO
3. $q_B=B_q\cap B$;
4. $E=E+EstInBucket(B,q_B)$;
5. $q=q-q_B$;
6. IF $c_r\geq threshold$ THEN $E=E+\dfrac{vol(q)}{vol_{net}(B)}f_{net}(B)$;
7. ELSE $E=E+\iint_q \varphi(x,y)\mathrm{d}x\mathrm{d}y$;
8. return E;

4.2 密度函数

本节给出算法 2 和算法 3 中的函数 $\varphi(x,y)$.我们的方法基于如下观察:

观察结果 3. 由于构建直方图的过程是依据用户的查询反馈不断地在超方体内挖“洞”,“洞”的边界正是用户查询区域的边界,数据分布在每个超方体内具有一定的连续性.因而,桶内的数据分布密度和桶周围区域的数据分布密度差异不大.

根据观察结果 3,桶内的数据分布密度可以用来近似桶周围附近的数据分布.因此,我们不再始终假设桶内的数据是均匀分布的,而是用桶内数据分布状况建立参数函数来估计桶周围附近的数据分布.在远离桶的数据区域,建立参数函数的信息无法获得,此时仍假设数据在桶内是均匀分布的.

在物理上,物体的质量可以看作集中分布在其重心上.如果我们将桶内元组的分布看成超方体质量的分布,则可以计算每个桶的重心.如果桶没有洞且数据在其中均匀分布,则桶的重心就位于相应超方体的中心.但是,由于我们构建超方体的过程要不断地在桶内挖“洞”以刻画桶内数据分布不均匀的地区,因此,桶的重心可能会漂移.同时,桶的合并、删除、净频率的变化也会引起桶的重心漂移.我们用定理 2 和定理 3 来计算桶 B 的重心.

定理 2. 设桶 A 是桶 B 的唯一子桶,在桶内数据均匀分布的假设下,桶 A 和桶 B 的密度函数分别为 $d(A)$和 $d(B)$,则桶 B 的重心可如下计算:

$$\bar{x}_B=\frac{1}{2f(B)}\sum_{X\in\{A,B\}}(d(X)-d(P(X)))F_1(X),\ \bar{y}_B=\frac{1}{2f(B)}\sum_{X\in\{A,B\}}(d(X)-d(P(X)))F_2(X) \tag{5}$$

其中,$P(X)$表示 X 的父桶,$d(P(B))=0$,且

$$F_1(X(v_l,v_u))=(v_u^{(2)}-v_l^{(2)})([v_u^{(1)}]^2-[v_l^{(1)}]^2),\ F_2(X(v_l,v_u))=(v_u^{(1)}-v_l^{(1)})([v_u^{(2)}]^2-[v_l^{(2)}]^2) \tag{6}$$

定理 3. 设 B 是直方图 H 的一个桶,X 是嵌套在 B 中的任意子桶且 $P(X)$表示桶 X 的直接父桶,令 $d(P(B))=0$,$F_1(X)$和 $F_2(X)$如公式(6)所定义,则在桶内数据均匀分布的假设下,桶 B 的重心可如下计算:

$$\bar{x}_B=\frac{1}{2f(B)}\sum_{X}(d(X)-d(P(X)))F_1(X),\ \bar{y}_B=\frac{1}{2f(B)}\sum_{X}(d(X)-d(P(X)))F_2(X) \tag{7}$$

在桶的重心被计算出来以后,我们建立参数函数来近似桶附近的数据分布状况.对于桶 B,设其重心为 $(\bar{x}_B,\bar{y}_B)$,它到超方体 B 的边界的最短距离记为 u_B.根据重心的物理意义和观察结果 3,可以认为桶 B 内数据密度最大的点即为重心.随着到重心的距离 t 的增加,数据密度将按照某种函数关系 $g_\theta(t)$有所下降.其中,θ是待确定的参数.在确定这个函数的过程中,需要考虑以下 3 个问题:第 1 个问题是,根据桶 B 提供的信息确定的函数 $g_\theta(t)$应该有一个有效半径,因为桶 B 的数据分布只能反映桶 B 附近的数据分布;第 2 个问题是,$g_\theta(t)$中参数的个数应该如何确定;第 3 个问题是,$g_\theta(t)$应该采用什么样的函数形式.下面,我们逐一讨论这 3 个问题.

首先,由于直方图是根据用户查询的反馈来建立和维护的,每个桶内元组的个数是精确的.因此,由桶 B 建立的参数函数 $g_\theta(t)$不能表示其他桶内部的数据分布,设距离 B 最近的桶到$(\bar{x}_B,\bar{y}_B)$的距离为 dis_B,则我们有

$$g_\theta(dis_B)=0 \tag{8}$$

公式(8)给出的条件表明,参数函数 $g_\theta(t)$只能用来估计桶 B 附近的数据分布.如果一个查询距离所有的桶均很远,则所有桶的参数函数均不对该查询起作用.这时,我们不得不再假设数据在整个数据空间是均匀分布的.这种处理方式是合理的,因为对某个数据区域的数据分布了解得越多,才能在该区域上获得更精确的估计结果.

除了公式(8)的条件以外,还知道桶 B 的平均密度 $d(B)$,重心处精确的数据密度是不知道的.将桶 B 的平均数据密度看成桶 B 边界上距离重心最近的点的数据密度.这样,即便是桶 B 附近的点,由于它们距离重心的距离不一样,用参数函数估计得到的数据密度也不一样.距离重心越远,由参数函数得出的数据密度越小.根据重心的物理意义,这是合理的.所以,

$$g_\theta(u_B)=d(B) \tag{9}$$

由桶 B 提供的信息仅能建立公式(8)和公式(9)的方程,故阐述函数中的参数最多有两个参数.

原则上讲,带有两个参数的函数均可以选作参数函数.比如,可以选用带有两个参数的多项式函数 $at+b$,at^2+bt,at^2+b 等.如果已知数据在局部范围内近似服从某种分布,则可以选用相应的数据分布函数.例如,若知道数据在局部范围近似服从 Gausss 分布,则可选用形如 ae^{-bt} 的参数函数.事实上,可以给出候选的参数函数集合,根据建立直方图的过程中它们估计查询选择性所得结果的精确性来最终确定参数函数的类型.当参数函数的类型选定之后,可以用公式(8)和公式(9)给出的条件来确定参数.对于桶 B,设最终得到的参数函数为 $g_\theta(t)$,$t\in[u_B,dis_B]$,则可以按照如下操作将它扩展到整个空间内:

$$G_B(t)=\begin{cases}d(B), & t<u_B\\ g_\theta(t), & u_B\le t\le dis_B\\ 0, & \text{否则}\end{cases} \tag{10}$$

如果查询点(x,y)在多个参数函数的有效半径之内,则该点上的数据密度是这些参数函数之和.因此,数据空间内任意一点的数据密度函数可以表达成

$$\varphi(x,y)=\sum_{B\in H}G_B(dis((\bar{x}_B,\bar{y}_B),(x,y))) \tag{11}$$

4.3 直方图的动态维护

直方图所占用的内存空间有限,不能无限地在直方图中添加桶.当直方图的内存空间被占满后,在添加新桶之前,必须合并直方图中两个相邻的桶.我们讨论两种情况:一是子桶与父桶的合并;二是两个兄弟子桶之间的

合并.

对于第 1 种情况,我们找到 *gap* 最小的桶 $B_{\min}$,将它的 *gap* 值记为 $G_{\min}$.*gap* 值的意义表明,$B_{\min}$ 的数据密度与其父桶之间的数据密度最接近,合并它对直方图精确性的影响最小.

对于第 2 种情况,考虑桶 B_p 的两个直接子桶 B_1 和 B_2.在合并 B_1 和 B_2 时,我们首先找出包含超方体 B_1 和 B_2 的最小超方体 B_{new},如图 1 所示.一般来讲,B_{new} 中包含了桶 B_p 中原来不在 B_1 和 B_2 内部的部分 B_{old}(图 1 中的阴影部分).对于 B_{old},我们先计算其净体积,再通过公式(12)得到 B_{old} 中元组数的估计值.然后,从 B_p 中"挖掉"B_{old} 得到 $B_{p'}$,并相应地更新 $B_{p'}$的净体积和净频率.再删除桶 B_1 和 B_2,构建新桶 B_{new},计算其净体积和净频率.根据桶 $B_{p'}$ 和 B_{new} 的信息,分别重新计算其 *gap* 值,将桶 B_{new} 的 *gap* 值记为 G_{new}.兄弟子桶的合并操作有两个缺点:一是确定 B_{new} 是一个代价较高的操作,因为我们必须找出合适的 B_1 和 B_2 以确保 B_{old} 内没有其他桶;二是 $f(B_{old})$是一个估计值,这将影响直方图的平均精度.

$$f(B_{old})=\frac{vol_{net}(B_{old})}{vol_{net}(B_p)}f_{net}(B_p) \tag{12}$$

基于以上讨论,我们优先考虑子桶与父桶的合并,即首先从全局范围找到 *gap* 值最小的桶 B,然后在 B 的所有直接子桶中考虑是否有兄弟子桶合并产生新桶 B_{new} 使得 $G_{new}<G_{\min}$.如果有合适的兄弟子桶,则合并兄弟子桶;否则,将桶 B 与其父桶合并.合并完成后,修改相关桶的频率值和 *gap* 值.

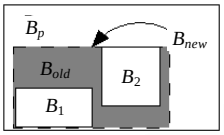


Fig.1 Merge of two sibling buckets

图 1 兄弟子桶的合并

随着直方图中桶的插入、合并等操作,桶的重心也会漂移.假设桶 B 的重心为$(\bar{x}_B,\bar{y}_B)$,且一个新到来的查询反馈使得我们在 B 内部插入了一个新桶 B_I,并删除了桶 B_1 的某个子孙桶 B_D.现在,我们来计算更新之后桶 B 和 B_1 的重心.由于当 $B=B_1$ 时我们可以先计算删除 B_D 后桶 B 的重心然后再计算插入 B_I 后 B 的重心,因此,实际上,我们可以只考虑仅由一个插入或删除操作引起的重心转移.

定理 4. 设 B 是直方图 H 中数据密度为 $d(B)$的桶,其重心为$(\bar{x}_B,\bar{y}_B)$,$F_1(X)$和 $F_2(X)$由公式(6)定义.如果在桶 B 内插入一个数据密度为 $d(B_I)$的子桶 B_I,由此导致桶 B 的数据密度变成 $d'(B)$,则 B 的重心变成:

$$\begin{aligned}\bar{x}&=\bar{x}_B+(d'(B)-d(B))\left[\sum_{X\in cild(B)}(F_1(X)-F_1(B))\right]+(d(B_I)-d(P(B)))F_1(B_I)\\ \bar{y}&=\bar{y}_B+(d'(B)-d(B))\left[\sum_{X\in cild(B)}(F_2(X)-F_2(B))\right]+(d(B_I)-d(P(B)))F_2(B_I)\end{aligned} \tag{13}$$

定理 5. 设 B 是直方图 H 中数据密度为 $d(B)$的桶,其重心为$(\bar{x}_B,\bar{y}_B)$,$F_1(X)$和 $F_2(X)$由公式(6)定义.若在桶 B 内删除一个数据密度为 $d(B_D)$的子桶 B_D,由此导致桶 B 的数据密度变成 $d'(B)$,则 B 的重心变成

$$\begin{aligned}\bar{x}&=\bar{x}_B+(d'(B)-d(B))\left[F_1(B)-\sum_{X\in cild(B)\setminus\{B_D\}}(F_1(X)\right]-(d(B_D)-d'(B))F_1(B_D)\\ \bar{y}&=\bar{y}_B+(d'(B)-d(B))\left[F_1(B)-\sum_{X\in cild(B)\setminus\{B_D\}}(F_2(X)\right]-(d(B_D)-d'(B))F_2(B_D)\end{aligned} \tag{14}$$

5 实 验

5.1 实验数据与运行平台

我们实现了本文给出的自适应直方图 BQHist,同时实现了文献[9]中给出的自适应直方图 STHoles.实验从 4 个方面比较了 BQHist 和 STHoles:一是用公式(3)中的度量比较了 BQHist 和 STholes 的平均精度(见第 5.2 节);二是比较了用户查询区域的变化对 BQHist 和 STholes 的影响(见第 5.3 节);三是比较了数据批量更新对 BQHist 和 STholes 的影响(见第 5.4 节);四是比较了两个直方图对系统整体性能的影响.

实验运行在主频为 2.7GHz、内存容量为 512M 的奔腾 IV 台式机上.后台数据库是 Oracle 9*i*.代码用 Java 语言编写,程序通过 ArcSDE Java API 接口与后台数据库相连接.

实验采用了 1 组真实数据和 4 组人工数据.真实数据是澳大利亚昆士兰州环境保护组织提供的野生动物活动区域的数据 SRE(398 464 条记录)、2 维和 3 维 Gauss 分布和对数分布的数据集.服从 Gauss 分布的数据集由

一系列 Gauss 分布单元构成,每个 Gauss 分布单元具有一定的标准差,包含 1 600 000 条记录.Gauss 分布单元内的元组数服从 Zipf 分布.服从对数分布的数据集的生成方法与此类似,只是将 Gauss 分布替换成对数分布.

5.2 平均精度的比较

为了比较 BQHist 和 STHoles 的平均精度,生成了 3 组查询来模拟查询,每组查询均随机动态生成,见表 2.表格中第 2~5 列分别是查询树中根节点以下第 1~4 层查询的总数,兄弟查询可以相交.在第 1 组查询中,第 1~4 层的每个查询分别至少覆盖整个数据空间的 2%,1%,0.4%和 0.2%.在第 2 组查询中,各层的每个查询分别至少覆盖整个数据空间的 1%,0.5%和 0.25%.在第 3 组查询中,各层每个查询分别至少覆盖整个数据空间的 1%,0.5%,0.25%和 0.01%.

Table 2 Three groups of queries in experiments

表 2 实验中的 3 组查询

Group ID	Level 1	Level 2	Level 3	Level 4
1	50	100	300	600
2	50	200	800	0
3	250	250	250	250

首先,我们在 SRE、2 维 Gauss 数据集和 2 维对数数据集上用第 1 组查询比较了直方图的平均精度.在每个数据集上,先执行 250 个查询,再用剩下的查询测试直方图的平均精度,将结果画成柱状图(如图 2 所示).其中,BQHist-uni 表示在 BQHist 中未使用参数函数估计,并假定桶内的数据均匀分布;BQHist-par 表示在 BQHist 中用参数函数估计.可以看到,用 BQHist 的估计查询选择性的平均误差比 STHoles 低约 30%;在 BQHist 中使用参数估计可以将平均误差降低约 10%.这验证了 3 个观察结果和由此引出的调度策略、参数函数估计方法的合理性.

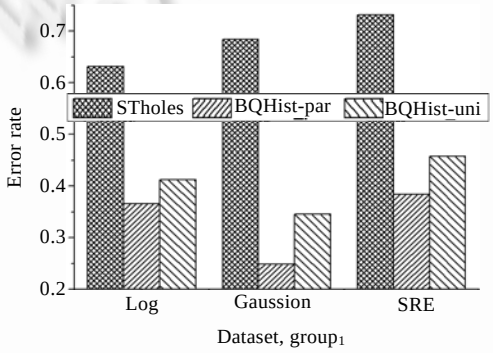


Fig.2 Average accuracy of histograms

图 2 直方图的平均精度

为了进一步比较参数函数类型的选取对平均精度的影响,我们另外生成了一组有 10 个层次的查询,在 SRE 上运行.第 1 个层次含 10 个查询,每个查询覆盖整个数据空间的 3%;每层每个查询下有两个兄弟查询,覆盖率较其父查询降低 10%.分别选用指数函数、线性函数和多项式函数作为参数函数,得到的平均误差率如图 3 所示.可以看到,参数函数的选取对选择性估计的准确性影响也很大.由于澳大利亚野生动物分布比较均匀,桶周围的数据密度与桶周围的数据密度相差不大.指数函数下降太快,得到的效果最差.线性函数和多项式函数下降速度较慢,能够较好地模拟真实的数据分布.将每个桶内的数据看成是均匀的,估计结果也很差.这是由于桶周围估计的结果偏低,而子桶内部估计的结果偏高.

我们还比较了多项式的最高次数对直方图平均精度的影响,如图 4 所示.可以看到,用不同次数的多项式在真实数据集上来拟合桶周围的数据分布,得到的平均精度差别不大.

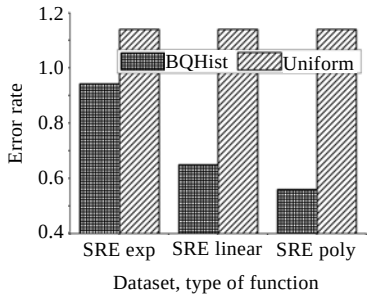


Fig.3 Impact of para-functions on average accuracy

图 3 参数函数类型对平均精度的影响

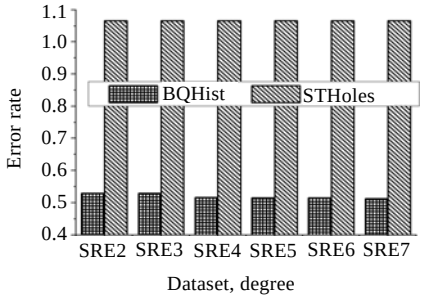
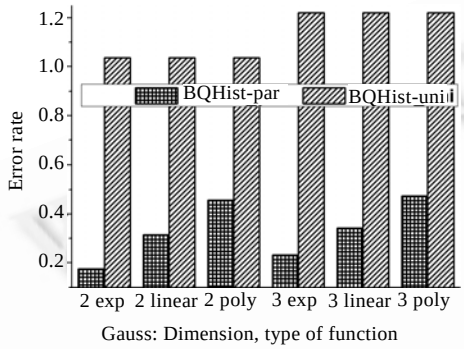


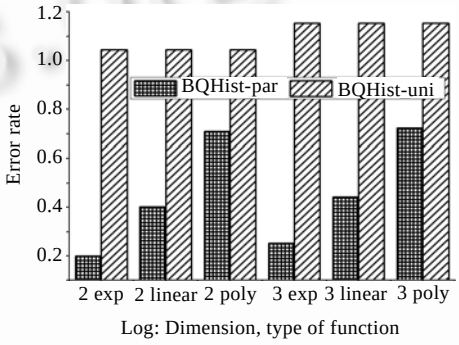
Fig.4 Impact of poly_degree on average accuracy

图 4 多项式次数对平均精度的影响

此外,我们还在 4 个人工数据集上比较了 BQHist 和 STHoles 的平均精度,如图 5 所示.其中,图 5(a)是在 Gauss 数据集上得到的结果,图 5(b)是在对数数据集上得到的结果.注意到,此时选用指数函数的平均误差最小,这是因为它大致等同于数据的分布函数;用多项式函数作为参数函数得到的结果其精度仍然比较稳定.此外,当维数增加时,平均误差的变化不大,这印证了方法的稳健性.



(a)



(b)

Fig.5 Average accuracy of histograms on synsetic datasets

图 5 人工数据集上直方图的平均精度

5.3 用户查询区域的变化对直方图的影响

将整个数据空间等分成 4 个子区域.先在第 1 个数据区域随机生成 100 个查询,将它们组织成查询树,执行完成后再在第 2 个区域内随机生成 100 个查询.第 3 个数据区域和第 4 个数据区域类似地处理.根据公式(3)计算平均误差,每 20 个查询取样 1 次,得到的曲线如图 6 所示.可以看到,在真实数据集和 Gauss 数据集上,BQHist 总比 STHoles 能够获得更好的平均精确度.同时,当查询区域转移时,BQHist 的误差上升高度明显小于 STHole 的误差上升高度,这说明 BQHist 更能适应查询区域的变化.

直方图对用户查询区域变化的适应性将导致直方图具有更快的收敛速度.即用同样的查询序列来建立自适应直方图,达到同样的平均精度,BQHist 较 STholes 用掉的查询要少.为此,我们在 6 个数据集上生成了表 2 列出的前两个查询序列.分别执行它们,并用它们的反馈建立自适应直方图 BQHist 和 STHoles,用公式(3)计算平均精度,每隔 50 个查询采样 1 次平均误差率,得到了如图 7 所示的曲线.对于 Gauss 数据集和对数数据集,得到的曲线与此类似.可以看到,为了达到相同的平均精度,BQHist 用的查询数明显小于 STHoles,这说明 BQHist 的收敛速度快于 STHoles.

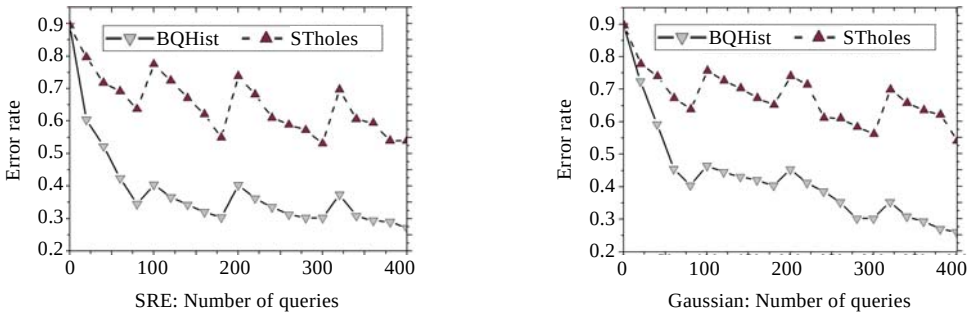


Fig.6 Adaptability of histograms to the change of query ranges

图 6 直方图对查询区域变化的适应性

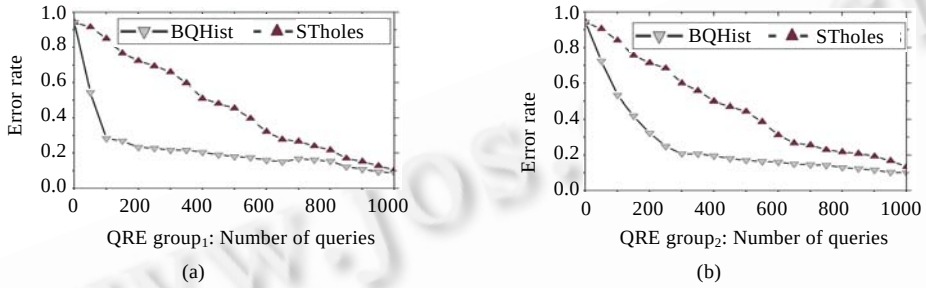


Fig.7 Convergence speed of histograms

图 7 直方图的收敛速度

我们还考察了调度策略中的 k 值对自适应直方图收敛速度的影响.分别令 k=5,10,15 来运行表 2 的查询序列,用公式(3)来计算平均误差率,每隔 20 个查询对平均误差率,得到如图 8 所示的曲线.

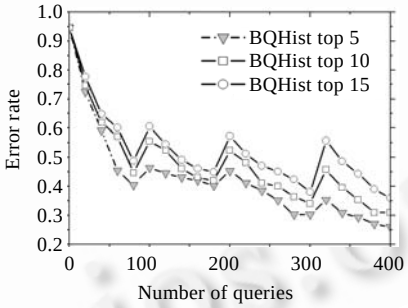


Fig.8 Impact of k on convergence speed

图 8 参数 k 对收敛速度的影响

由图 8 可以看出,k 值越大,收敛速度越快.这是由于,k 越大,候选的热点查询越多,就可以更有效地刻画数据分布不均匀区域的数据分布,使得后执行的查询位于热点数据区域的部分被估计得更准确.

5.4 数据更新对直方图的影响

基于查询反馈来建立自适应直方图的一个优点在于,所建立的直方图能够有效地适应数据集的变化.在压缩数据库中,数据集往往批量压缩加载,因此,这个优点在压缩数据库中显得格外重要.首先生成 25%的 Gauss 单元,执行 500 个随机生成的查询来建立直方图,用公式(3)计算直方图的平均精度;然后再生成 25%的数据加载到

数据库中,执行500个查询,动态维护直方图,并计算其平均精度.重复上述过程,直到整个数据集被生成为止.图9中的实验结果表明,BQHist较STHole更能适应数据集合的变化.

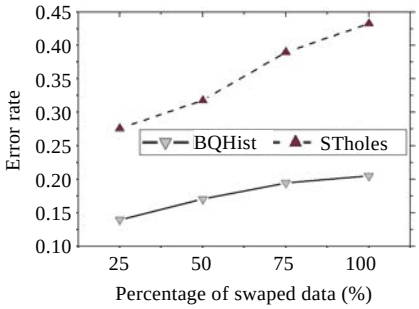


Fig.9 Impact of batched-upload on the performance of histogram

图9 批量数据加载对性能的影响

5.5 直方图对系统性能的影响

我们按照第5.1节的设置在SRE(state regional ecosystem)上对第1组查询分别用BQHist和STHoles来估计查询的选择性,并相应地选用查询计划.由于查询在BQHist和STHoles中执行的顺序不同,直接比较每个查询的查询计划是否相同很困难.实验中,我们分别统计各个层次的查询用BQHist和STHoles选取的查询计划的平均运行时间,计算它们的比值.我们发现,第1~4个层次查询的平均运行时间的比值依次约为0.95,0.93,0.96和0.98.可以看到,由于查询调度追踪特点数据导致查询选择性估计精度的提高,可能导致查询优化器选用更优的查询计划,最终使得查询执行的效率整体上明显提高.此外,我们还统计了查询调度的总时间开销,然后将它平均到每个查询上.我们发现,查询调度的时间开销相对于查询执行时间是可以忽略的.

6 结 论

压缩海量数据库中压缩数据的批量加载会引起数据分布频繁发生变化,且数据空间中热点数据区域可能会频繁发生变化.这导致传统直方图对查询选择性的估计十分不准确,其更新或重建代价很高.针对这些问题,本文提出通过对查询缓冲池内查询进行调度,根据查询反馈建立和更新自适应直方图,以有效跟踪热点数据区域、用户查询区域以及数据集数据分布的变化,提高自适应直方图的平均精度.并提出了参数方法来估计数据空间中未被直方图覆盖的区域中的查询,讨论了参数函数的维护策略.针对压缩海量数据的特征,我们在真实数据集和人工数据集上设计了大量的实验.实验结果表明,我们建立的自适应直方图比STHoles具有更好的平均精度、更快的收敛速度和更强的自适应能力.

References:

[1] Wu WL, Gao H, Li JZ. New algorithm for computing cube on very large compressed data sets. IEEE Trans. on Knowledge and Data Engineering, 2006,18(12):1667−1680.

[2] Luo JZ, Li JZ, Zhao K. An Iceberg cube algorithm for large compressed data warehouses. Journal of Software, 2006,17(8): 1743−1752 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/1743.htm

[3] Chen ZY. Building compressed database system [Ph.D.Thesis]. Cornell University, 2002. 19−132.

[4] Jermaine C, Omiecinski E. Lossy reduction for very high dimensional data. In: Proc. of the 18th Int'l Conf. on Data Engineering. San Jose: IEEE Computer Society, 2002. 663−672.

[5] Pinar A, Tao T, Ferhatosmanpglu H. Compressing bitmap indices by data reorganization. In: Kawada S, ed. Proc. of the 21st Int'l Conf. on Data Engineering. Tokyo: IEEE Comperter Society, 2005. 310−321.

[6] Loannidis Y. The history of histograms (abridged). In: Freytag JC, Lockemann PC, Abiteboul S, Carey MJ, Selinger PG, Heuer A, eds. Proc. of the 29th Int'l Conf. on Very Large Data Bases. San Fransisco: Morgan Kaufmann Publishers, 2003. 19−30.

[7] Lim L, Wang M, Vitter JS. Sash: A self-adaptive histogram set for dynamically changing workloads. In: Freytag JS, Lockemann PC, Abiteboul S, Carey MJ, Selinger PG, Heuer A, eds. Proc. of the 29th Int'l Conf. on Very Large Data Bases. San Fransisco: Morgan Kaufmann Publishers, 2003. 423−434.

[8] Aboulnaga A, Chaudhuri S. Self-Tuning histograms: Building histograms without looking at data. In: Delis A, Faloutsos C, Ghandeharizadeh S, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 1999. 181−192.

[9] Bruno N, Chaudhuri S, Gravano L. Stholes: A multidimensional workload-aware histogram. In: Aref WG, ed. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2001. 294−305.

[10] Lipton RJ, Naughton JF, Schneider DA. Practical selectivity estimation through adaptive sampling. In: Garcia-Molina H, Jagadish HV, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 1990. 1−11.

[11] Wu YL, Agrawal D, Abbadi AE. Applying the golden rule of sampling for query estimation. In: Aref WG, ed. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2001. 299−400.

[12] Wu YL, Agrawal D, Abbadi AE. Query estimation by adaptive sampling. In: Proc. of the 18th Int'l Conf. on Data Engineering. San Jose: IEEE Computer Society, 2002. 639−650.

[13] Chen CM, Roussopoulus N. Adaptive selectivity estimation using query feedback. In: Snodgrass RT, Winslett M, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 1994. 161−172.

[14] Donjerkovic D, Ioannidis Y, Ramakrishnan R. Dynamic histograms: Capturing evolving data sets. In: Proc. of the 16th Int'l Conf. on Data Engineering. San Jose: IEEE Computer Society, 2000. 86−97.

[15] Ioannidis YE, Poosala V. Balancing histogram optimality and practicality for query result size estimation. In: Carey MJ, Schneider DA, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 1995. 233−244.

[16] Konig AC, Weikum G. Combining histograms and parametric curve-ftting for feedback-driven query result-size estimation. In: Atkinson MP, Orlowska ME, Valduriez P, Zdonik SB, Brodie ML, eds. Proc. of the 25th Int'l Conf. on Very Large Data Bases. San Fransisco: Morgan Kaufmann Publishers, 1999. 423−434.

[17] Matias Y, Vitter JS, Wang M. Wavelet-Based histograms for selectivity estimation. In: Haas LM, Tiwary A, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 1998. 448−459.

[18] Poosala V, Ioannidis YE, Haas PJ, Shekita EJ. Improved histograms for selectivity estimation of range predicates. In: Jagadish HV, Mumick IS, eds. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 1996. 294−305.

[19] Jagadish HV, Koudas N, Muthukrishnan S, Poosala V, Sevcik KC, Suel T. Optimal histograms with quality guarantees. In: Gupta A, Shmueli O, Widom J, eds. Proc. of the 24th Int'l Conf. on Very Large Data Bases. San Fransisco: Morgan Kaufmann Publishers, 1998. 275−286.

附中文参考文献:

[2] 骆吉洲,李建中.大型压缩数据仓库上的 Iceberg Cube 算法.软件学报,2006,17(8):1743−1752. http://www.jos.org.cn/1000-9825/17/1743.htm

骆吉洲(1975－),男,重庆人,博士,副教授,主要研究领域为压缩数据库技术,传感器网络,算法设计与分析.

王宏志(1978－),男,博士,讲师,主要研究领域为 XML 查询处理,数据质量.

李建中(1951－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据管理,传感器网络,并行计算,计算生物学.