数据流变化的检测*)

聂国梁　卢正鼎
(华中科技大学计算机科学与技术学院　武汉 430074)

摘　要　通过对数据流的两个相邻窗口的比较,检测出绝对变化较大的元素,以此来描述流数据的变化。把单个窗口中的数据流划分成若干层,在每层上对数据值域进行分段。然后在每层上定义若干分段集合,并对分段集合进行求和运算。通过对两个窗口的概要结构进行合并,采用二分法,利用集合的分解,可以求得变化较大的元素。理论和实验证明,本算法利用对数空间有效地解决了数据流中变化较大元素的检测问题。

关键词　数据流,近似算法,数据流统计

Detecting Change of Data Stream

NIE Guo-Liang　LU Zheng-Ding
(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074)

Abstract　Detecting change of data stream plays an important role in many data stream's decision support systems. The change of data stream is described by detecting the elements whose value difference between two adjoining windows exceeds threshold value. Single window data stream is divided into several levels, each of which partitions all elements into some groups. Some supersets over groups are defined, and the sum is calculated for each group. After combining sketches of two windows, the elements whose value exceeds threshold value are detected by performing binary search. Theory and experiments prove that the algorithm is accurate and effective for detecting change of data stream.

Keywords　Data stream, Approximation algorithms, Data stream statistics

1　引言

近年来,随着互连网、无线电通讯、电子商务和传感网络的蓬勃发展,一种新兴的数据类型——数据流越来越受到人们的重视。数据流是由众多元素构成的无限序列,它在以下三个方面不同于传统的数据类型:(a)连续性;(b)长度未知或无限;(c)无法访问以前到达的元素,或者是访问历史数据代价过大。数据流在其活跃期内,可能变化很大。而这种变化可能反映了一个事实:有异常情况发生。如在网络中,数据流突变可能暗示着某些节点正在发起恶意攻击。对这种变化进行量化,能给决策者提供有益的参考信息。传统的数据变化检测算法并不适用于数据流。因此,研究数据流的变化检测算法是必要的,而且是意义重大的。

本文通过对数据流的两个相邻窗口进行比较,检测变化较大的元素,以此来表示数据流的变化状况。为了描述方便,首先定义一些基本概念。

元素 i 在数据流窗口 W 中的值用 $W[i]$ 表示。sketch 为窗口 W 的概要数据结构,sketch$[i]$表示元素 i 在 sketch 中的值。对集合 d,本文利用 SKL_d 表示 $\sum_{i\in d}$sketch$[i]$。

对尺寸都为 N 的窗口 a 和 b,用 DIF$[i]$来表示元素 i 在这两个窗口中的绝对变化,DIF$[i]=|a[i]-b[i]|$。用 $L_{a,b}$ 来表示流数据在这两个窗口的变化,$L_{a,b}=\sum_{i\in(a\cup b)}DIF[i]$。用 $SD_{m,n}$ 表示 $\sum_{i=m}^{n}$DIF$[i]$。元素 i 的绝对变化率为 DIF$[i]/L_{a,b}$。对集合 d,本文利用 SDL_d 表示 $\sum_{i\in d}$DIF$[i]$。

本文所要解决的问题描述为:

给定阈值参数 $\theta(\theta<1)$,通过对两个相邻窗口 a 和 b 中的数据流进行比较,输出绝对变化率不小于 θ 的元素 e 和 DIF$[e]$。

2　相关工作

通过对两个窗口中数据流的比较来发现变化较大的元素,这个问题实际上是热门元素发现问题的复杂版本,也可以称为变化热门元素发现问题。当把一个窗口看作空时,变化热门元素发现问题就简化成为热门元素的发现问题。有关数据流的热门元素发现问题,国内外取得了一定的进展。

Fisher 和 Salzberg[1] 提出了 Majority 算法,用来查找频率超过 1/2 的元素。该算法通过对数据集进行单遍访问,就能找到满足条件的元素。Misra 和 Gries[2] 对 Majority 算法进行了扩展,使其适用于更一般的情况,可以用来查找频率超过 $1/k$ 的元素。近来,Demaine 等人[3] 和 Karp 等人[4] 又将这一算法进行了改进,把最坏情况下的处理时间减少到 O(1)。

Manku 和 Motwani[5] 提出了用损耗统计算法来解决热门元素的发现问题,这是一种确定性的算法,它非常适用于分布不对称的数据流。

Golab 等人[6] 提出了一种算法,该算法使用基本窗口模式在滑动窗口中查找热门元素。这一算法利用有限的内存空间对数据流进行单遍访问,每个元素的处理时间恒定。其缺点是在最坏情况下,空间代价偏大。

Arasu 和 Manku[7] 对 ε 近似热门元素统计和分位数问题

*)国家自然科学基金资助项目(60403027)。聂国梁　博士研究生,主要研究数据库技术和流数据算法;卢正鼎　教授,博士生导师,从事分布异构系统和数据库技术研究。

提出了确定性的和随机性的算法。这些算法既适用于固定尺寸的滑动窗口，也适用于变化尺寸的滑动窗口。但它们输出了大量不满足条件的元素，大大降低了程序的实用性。

虽然关于数据流的热门元素发现问题已经被国内外学者深入研究，但是变化热门元素发现问题的研究却不多，只有Cormode和Muthukrishnan[8]对变化热门元素的发现问题提出了解决方案。该方案采用哈希方法，把两个窗口中的数据流元素分别哈希到不同的集合，该集合利用位指针记数。然后合并集合。对合并集合，通过对大于用户指定阈值的位指针的组合，输出元素。集合合并会引入误差，$L_{a,b}$的计算也会引入误差。该方案却忽略了这些误差。

与本文紧密相关的还有F_1[9]的计算问题。

Alon等人[9]最早为单个数据流的$F1$提出了解决方案。Feigenbaum等人[10]把该方案进行了改进，解决了两个数据流A和B之间的$L_{a,b}$的计算问题。Indyk[11]利用稳定分布的特性，解决了两个数据流A和B之间的$L_{a,b}$的计算问题。

本文基于随机子集求和的方法[12]，为单个窗口中的数据流构造了一种新的概要结构，并提出了一个算法，解决了变化热门元素的查询问题。当需要查询两个窗口流之间变化热门元素的时候，合并对应的概要结构，采用二分法来挖掘变化热门元素。一旦为两个窗口的数据流建立ε近似概要结构，就可以查询绝对变化率不小于任意β(只要β大于ε)的变化热门元素。

3 算法

数据流的数据格式在本文中为(u,v)。u用来标识元素，它在窗口中是唯一的；v用来代表u的值。u的值域是$[0,U-1]$，v的值域是$[0,V-1]$(U,V都是大于0的长整数)。为了表述方便，本文假定U是2的整数次幂。如果不是，可以放大U来满足。

首先，描述单个窗口中数据流的概要数据结构的生成过程。

本文在概念上将数据流复制$L+1$次，$L=\log(U)$。这些复制的数据流处于不同的层上，这些层的顺序记为$0,1,\cdots,L$。在每层上，把U的值域均匀分成若干子段。在第I层上，每个子段的大小为$2^{\log|u|-I}$，即每个子段可表示为$[k2^{\log(U)-I},(K+1)2^{\log(U)-I}-1]$，$K$小于$2^I$。因此可见，在第0层上，就一个子段，它就是值域$[0,U-1]$，而在第$\log(U)$层上，每个子段的长度为1，即该层拥有$U$个子段，每个子段仅包含一个整数值。在每层上，本文都定义了若干子段集合，每个子段集合D都是以1/2的概率包含任意的子段。本文为每个子段集合D维持一个统计变量S。如果对每个D包含的子段都记载下来，则需要的空间太大，本文一般采用为D保存一个分段种子，利用种子来判断一个子段是否属于D。

一旦接收到一个新数据，就将其分别插入各层。

第I层对数据的处理框架描述如下：

一个数据(e,V_e)到达

(1)计算出e在该层上隶属的子段d。

(2)对该层上的每一个子段集合D，利用该集合的分段种子，检验d是否包含在其中。

(3)如果不包含，转步骤(5)。

(4)如果包含，则把E加入D对应的统计变量S，加入过程如下：

①为e生成随机变量R_e。

②结合R_e和$K(K<V_e)$，生成独立随机变量$R_{e,k}$，其取值只能为1或者-1，而且二者等概率。

③$S+=\sum_{j=1}^{V_e}R_{e,j}$

(5)继续执行步骤(2)。

由于子段的划分性质，因此可以通过把e右移I位，来判断e在第I层上所隶属的子段。

为每个子段集合D维护5个种子：IntervalSeed，SumSeed_0，SumSeed_1，SumSeed_2，SumSeed_3。通过对IntervalSeed和子段d进行运算来判断d是否属于D。对第j个子段(子段从0开始记数)，得到一个二进制字符串j'，它是数字j左移1位然后加1的二进制表示形式。然后把IntervalSeed和j'对应的位相乘，再相加，得到的和对2取模。如果为1，则第j个分段属于D；如果为0，则表示第j个子段不属于D。例如，IntervalSeed为101101。当j为6时，则j'为1101，通过运算则可判断：第6个子段属于D。而当j为5时，则j'为1011，通过运算可判断第5个子段不属于于D。

利用SumSeed和e生成随机变量R_e，生成公式如下：$R_e=\text{SumSeed}_3*e^3+\text{SumSeed}_2*e^2+\text{SumSeed}_1*e+\text{SumSeed}_0$。

$R_{e,j}$的生成：$R_{e,j}$取值只为-1或者1。$R_{e,j}$由R_e和V_e决定。$R_{e,j}=(-1)^{R_e*j'+f(j)}$。$f(j)$如文[10]定义。不失一般性，假设$j$的二进制表示的最高位为第$M$位(从0开始)，则$f(j)=(j_0\vee j_1)\oplus(j_2\vee j_3)\oplus\cdots\oplus(j_{m-1}\vee j_m)$或者$f(j)=(j_0\vee j_1)\oplus(j_2\vee j_3)\oplus\cdots\oplus(j_{m-2}\vee j_{m-1})\oplus j_m$。$j'$是数字$j$左移1位然后加1的二进制表示形式。$R_e*j'$表示把$R_e$和$j'$对应的位相乘，然后相加得到的和。而在实际中，$\sum_{j=1}^{U_e}R_{e,j}$只需要$O(\log(Ve))$的时间即可计算[10]。

以上描述了单个数据流的概要数据结构的生成方法。当需要发现变化热门元素的时候，首先把两个窗口中数据流的概要结构合并，然后在新的概要结构上采用二分法来寻找满足条件的元素。查询过程Query(low，high，threshold)，low代表查询下界，high代表上界，threshold代表条件阈值。Sum(low，high)估算出$\text{SD}_{m,n}$，具体过程在下面描述。查询过程Query(low，high，threshold)描述如下：

(1)如果Sum(low，high)$<$threshold，则本查询过程结束。

(2)如果low=high，则输出(low，Sum(low，high))，本查询过程结束。

(3)分别执行Query(low，low+(high$-$low)/2，threshold)和Query(low+(high$-$low)/2+1，high，threshold)。

4 算法分析

定理1 本文提出的算法能够以大于$1-\delta$的概率输出绝对变化率不小于θ的元素并估计绝对变化值，还能以大于$1-\delta/\log(U)$的概率保证变化值的误差小于$\varepsilon L_{a,b}$。算法的空间复杂度为$O(\log^2(U)\log(\log(U)/\delta)\log(NM)/\varepsilon^2)$比特($N$是窗口尺寸)。插入每个元素需要的更新为$O(\log^2(U)\log(\log(U)/\delta)\log(M)/\varepsilon^2)$。查询需要的时间复杂度为$O(\log(U)/(\theta-\varepsilon))$。

证明：在第0层上，维护$480\log^2(U)\log(\log(U)/\delta)/\varepsilon^2$个统计变量$S$。把统计变量均匀地划分到$3\log(\log(U)/\delta)$个组，每个组有$160\log^2(U)/\varepsilon^2$个统计变量。在合并后的概要

数据结构的第0层上，对每个组的统计变量进行平方计算，再求得均值。然后从这 $3\log(\log(U)/\delta)$ 个均值中求得中值 Lmedianofavg。利用中值 Lmedianofavg 来表示 $L_{a,b}$。根据切比雪夫不等式和契尔诺夫边界，可以证明：以大于 $1-\delta/\log(U)$ 的概率保证 $|L_{medianofavg}-L_{a,b}|<\varepsilon L_{a,b}$。文[10]给出了详细证明。

Sum(n,m)的求解：对$[n,m]$进行分解，分解到不同层不同子段。使得$[n,m]=d_{j1}\cup\cdots\cup d_{jr}$ $(r\leqslant\log(U))$。对任意子段 d_j，在其对应的层中找到子段集合 D_k，使得 $d_j\in D_k$，然后用 $2SKI^2_{D_k}-L_{medianofavg}$ 表示 SDI_{d_j}。

下面证明 Sum 求解的有效性。

由 $R_{e,k}$ 的定义和独立性可得 $E(SKI^2_D)=SDI_d$。在第 I 层上，用 d_j 表示第 j 个子段。子段集合 D 以 1/2 的概率包含任意的子段 d_j。定义变量 X_j，使得：如果 $d_j\in D$，$X_j=SKI_{d_j}$；否则，$X_j=0$，则 $SKI_D=\sum_i X_i$。

$$E(2SKI^2_D\mid d_k\in D)=E(2(\sum_i X_i)^2\mid d_k\in D)==E(2SKI^2_{d_k})+E(2SKI_{d_k}SKI_{D-d_k})+E(SKI^2_{D-d_k})$$

由 $R_{e,k}$ 的定义和变量之间的独立性决定：

$$E(\sum_{i\neq j}SKI_{d_i}SKI_{d_j})=0$$

所以

$$E(2SKI^2_D\mid d_k\in D)=SDI_{d_k}+L_{a,b}$$

$$E^2(2SKI^2_D\mid d_k\in D)=L^2_{a,b}+2L_{a,b}SDK_{d_k}+SDI^2_{d_k}$$

$$E((2SKI^2_D)2\mid d_k\in D)=E(4X^4_k+16X^3_k\sum_{i\neq k}X_i+24X^2_k(\sum_{i\neq k}X_i)^2+16X_k(\sum_{i\neq k}X_i)^3+4(\sum_{i\neq k}X_i)^4\mid d_k\in D)$$

由于独立变量 $R_{e,k}$ 是 3 相独立，而且以等概率取 -1 或者 1，因此仅 $E(R_{i1,j1}{}^4)$，$E(R^2_{i1,j1}R^2_{i2,j2})$ 和 $E(R_{i,j1}R_{i,j2}R_{i,j3}R_{i,j4})$ 可能不为 0。

所以

$$E((2SKI^2_D)^2\mid d_k\in D)=E(4X^4_k+24X^2_k(\sum_{i\neq k}X_i)^2+4(\sum_{i\neq k}X_i)^4\mid d_k\in D)$$

由随机变量的独立性和文[10]可知：

$$E(4X^4_k\mid d_k\in D)\leqslant 44SDI^2_{d_k}$$

$$E(24X^2_k(\sum_{i\neq k}X_i)^2\mid d_k\in D)=12SDI_{d_k}\sum_{j\neq k}SDI_{d_j}$$

$$E(4(\sum_{i\neq k}X_i)^4\mid d_k\in D)=E(4\sum_{i\neq k}X^4_i+24\sum_{i\neq j\neq k}X^2_iX^2_j)$$

$$E(4\sum_{i\neq k}X^4_i)\leqslant 22\sum_{i\neq k}SDI^2_{d_i},\ E(24\sum_{i\neq j\neq k}X^2_iX^2_j)=6\sum_{i\neq j\neq k}SDI_{d_i}SDI_{d_j}$$

所以

$$Var(2SKI^2_D\mid d_k\in D)\leqslant 44SDI^2_{d_k}+12SDI_{d_k}\sum_{j\neq k}SDI_{d_j}+22\sum_{i\neq k}SDI^2_{d_i}+6\sum_{i\neq j\neq k}SDI_{d_i}SDI_{d_j}-(L^2_{a,b}+2L_{a,b}SDI_{d_k}+SDI^2_{d_k})\leqslant 40L^2_{a,b}$$

由相关性知：

$$Var(\sum_{d_j\in D_k}(2SKI^2_{D_k}-L_{medianofavg}))\leqslant Var(\sum_{d_j\in D_k}2SKI^2_{D_k})+r^2Var(L_{medianofavg})$$

由文[10]可知：

$$r^2Var(L_{medianofavg})\leqslant\varepsilon^2L^2_{a,b}/16$$

对 $640\log(U)/\varepsilon^2$ 个 $2SKI^2_{D_k}$ 取均值 Z_{D_k}，得

$$Var(Z_{D_k})\leqslant\varepsilon^2L^2_{a,b}/16\log(U)$$

则

$$Var(\sum_{d_j\in D_k}(Z_{D_k}-L_{medianofavg}))\leqslant\varepsilon^2I^2_{a,b}/8$$

对 $3\log(\log(U)/\delta)$ 个 Z_{D_k} 取中值 MZ_{D_k}，使得 Sum(n,m) 等于 $\sum_{d_j\in D_k}(MZ_{D_k}-L_{medianofavg})$。

根据切比雪夫不等式和契尔诺夫边界，得 $Pr(|Sum(n,m)-SD_{n,m}|<\varepsilon L_{a,b})>1-\delta/\log(U)$。又因为一个变化热门元素的输出最多需要 $\log(U)$ 次判断，所以算法以大于 $1-\delta$ 的概率输出变化热门元素 e 及其绝对变化值。

第0层，需要维护 $3\log(\log(U)/\delta)$ 个组，每个组拥有 $160\log^2(U)/\varepsilon^2$ 个统计变量。对从1到 $\log(U)$ 的任意一层，需要维护 $3\log(\log(U)/\delta)$ 个组，每个组拥有 $1280\log(U)/\varepsilon^2$ 个统计变量。每个统计变量最多需要 $\log(NM)$ 个比特，N 表示窗口的尺寸。所以，本算法的空间复杂度为 $O(\log^2(U)\log(\log(U)/\delta)\log(NM)/\varepsilon^2)$。

当一个新的元素到达的时候，需要的更新最大为 $O(\log^2(U)\log(\log(U)/\delta)\log(M)/\varepsilon^2)$。

用二分法查找满足条件的元素，所需要的时间复杂度为 $O(\log(U)/(\theta-\varepsilon))$。

5 实验

所有的实验都是在处理器为 1.6GHz Pentium V、操作系统为 Windows Server 的 PC 机上进行的。实验数据来自模拟数据。

本文为了评价算法的有效性，定义了两个简单标准：参考价值和准确度。参考价值是包含在输出中的变化热门元素的数目与输出元素数目的比值。很明显，参考价值是介于 0 和 1 之间的实数。参考价值越接近 1，表明算法越有参考价值。准确度为输出的真实满足条件的元素数目和真实满足条件的元素数目的比值。同样，准确度也是介于 0 和 1 之间的实数。准确度越大，算法越理想。

本文首先通过实验对算法的性质进行了检验。

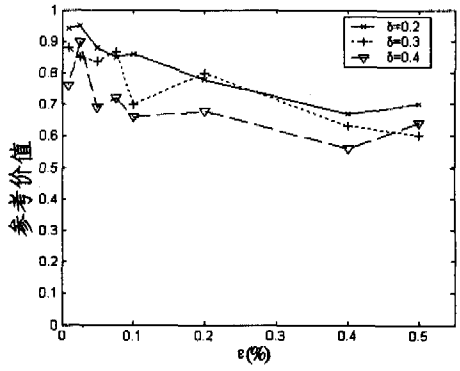

图 1 参考价值与近似参数和概率参数之间的关系

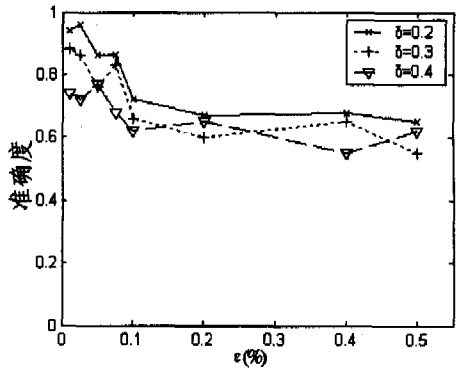

图 2 准确度与近似参数和概率参数之间的关系

图1描述了算法的参考价值性质。随着近似参数的增大,参考价值基本上呈现下降趋势,虽然中间可能有波动,波动是由于算法的不确定性决定的。当近似参数一样的时候,随着概率参数的增大,算法的参考价值会变小。图2展示了算法的准确度性质。同样,准确度随着近似参数的增大而缩小。同样的近似参数,准确度随着概率参数的减小而保持着上扬趋势。由此可见,近似参数和概率参数很大程度上决定了算法的参考价值和准确度。

其次,本文对本文算法和采样算法进行了比较。为了具有可比性,通过调整采样率,使得采样算法利用的空间等于本文算法的利用空间。

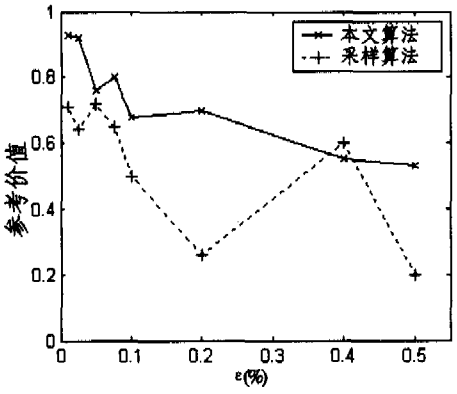

图3 本文算法和采样算法的参考价值比较

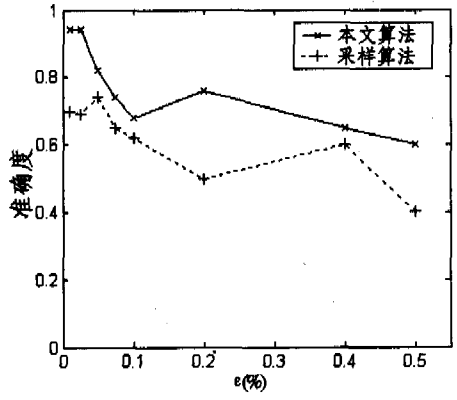

图4 本文算法和采样算法的准确度比较

由图3和图4可见,无论是参考价值和准确度,本文算法都要比采样算法好。随着近似参数的增大,采样算法的采样率降低,因此采样算法的参考价值和准确度都会降低。但是,由于随机性,其呈现较大的波动。虽然在局部采样算法可能比本文算法好,但是总体趋势上,本算法比采样算法更优秀、更实用。

总结 变化热门元素的检测问题,虽然在许多文献里提到,但真正解决此问题的很少。本文构造了一种新型的概要结构,基于此结构提出了一个算法,解决了该问题。算法以一定的概率(可以任意大)输出满足条件的元素,而需要的空间却远远小于数据流的尺寸。一旦为两个窗口的数据流建立ε近似概要数据结构,就可以查询绝对变化率不小于任意β(只要β大于ε)的变化热门元素。但本算法的缺点是可能会输出不属于数据流的元素。这个缺点也是下一步要解决的难点。与本文工作相关的数据流的变化检测模型也是亟需要研究的问题。数据流的研究尚处于初级阶段,还有许多工作需要做。

参考文献

1 Fischer M, Salzberg S. Finding a Majority among n Votes: Solution to Problem 81-5. Journal of Algorithms, 1982, 3(4): 376~379

2 Misra J, Gries D. Finding Repeated Elements. Sci Comput Programming, 1982, 2(2): 143~152

3 Demaine E D, L_opez-Ortiz A, Munro J I. Frequency Estimation of Internet Packet Streams with Limited Space. In: Proc. of the 10th Annual European Symp. on Algorithms, Rome, 2002

4 Karp R M, Shenker S, Papadimitriou C H. A Simple Algorithm for Finding Frequent Elements in Streams and Bags. ACM Trans. on Database Systems, 2003, 28(1): 51~55

5 Manku G S, Motwani R. Approximate Frequency Counts over Data Streams. Ramakrishnan. In: Proc. of the 28th Intl Conf. on Very Large Data Bases, Hong Kong, 2002

6 Golab L, DeHaan D, Demaine E, et al. Identifying Frequent Items in Sliding Windows over On-line Packet Streams. SIGCOMM Internet Measurement Conference, Miami, 2003

7 Arasu A, Manku G S. Approximate Counts and Quantiles over Sliding Window. In: Proc. of ACM Symp. on Principles of Database Systems, Paris, 2004

8 Cormode G, Muthukrishnan S. What's New: Finding Significant Differences in Network Data Streams: [technique report]. www.cs.rutgers.edu/~muthu/: S. Muthukrishnan, 2003

9 Alon N, Matias Y, Szegedy M. The space complexity of approximating the frequency moments. Journal of Computer and System Sciences, 1999, 58(1): 137~147

10 Feigenbaum J, Kannan S, Strauss M, et al. An approximate L1-difference algorithm for massive data streams. In: Proc. of the 40th Annual Symposium on Foundations of Computer Science, New York, 1999

11 Indyk P. Stable distributions, pseudorandom generators, embeddings and data stream computation. In: Proc. of the 41th Symposium on Foundations of Computer Science, Redondo Beach, 2000

12 Gilbert A, Kotidis Y, Muthukrishnan S, et al. How to summarize the universe: Dynamic maintenance of quantiles. In: Proc. of the 28th Intl Conf. on Very Large Data Bases, Hong Kong, 2002

(上接第158页)

结论和未来的工作 本文在空间数据库和数据采用R树索引的环境中解决了约束最接近对查询K-CCPQ。通过变换连接和范围查询的次序,给出了两个两阶段查询处理算法*RJ*和*JR*。同时,在定义的距离函数的基础上给出了裁减、更新和访问顺序规则,设计了基于堆的单阶段SPH算法。最后对一个真实的数据集进行了实验比较和分析,实验结果表明SPH算法具有较好的性能和较大的适用范围。设计约束最接近对查询的代价模型将是下一步的研究工作。

参考文献

1 Corral Y, Manolopoulos Y, Theodoridis, Vassilakopoulos M. Closest Pair Queries in Spatial Databases. In: Proc. of ACM SIGMOD Conf. 2000. 189~200

2 Yang, Lin K I. An Index Structure for Improving Nearest Closest Pairs and Related Join Queries in Spatial Databases. In: Proc. of Intl. Database Engineering and Applications Symposium (IDEAS'02), 2002. 140~149

3 Hjaltason G R, Samet H. Incremental Distance Join Algorithms for Spatial Databases. In: Proc. of ACM SIGMOD Conf. 1998. 237~248

4 Shin H, Moon B, Lee S. Adaptive Multi-Stage Distance Join Processing. In: Proc. of ACM SIGMOD Conf. 2000. 343~354

5 Ferhatosmanoglu H, Stanoi I, Agrawal D, Abbadi A E. Constrained Nearest Neighbor Queries. In: 7th International Symposium on Spatial and Temporal Databases(SSTD '01), 2001. 257~278

6 Shan Jing, Zhang Donghui, Salzberg B. On Spatial-Range Closest-Pair Query. In: Proc. of 8th Symposium on Spatial and Temporal Databases (SSTD'03), 2003. 252~269

7 Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. SIGMOD, 1984

8 Beckmann N, Kriegel H P, Schneider R, Seeger B. The R*-tree: an Efficient and Robust Access Method for Points and Rectangles. SIGMOD, 1990