数据流层次窗口模型及聚集查询算法*)

刘青宝　金　燕　侯东风　张维明

(国防科学技术大学信息系统与管理学院　长沙 410073)

摘　要　本文提出了一种多层次时间窗口模型，支持在不同时段对数据流进行不同粒度的建模，并给出了多粒度聚集树结构及其数据流聚集查询算法，从而有效地解决了在有限时空条件下的数据流聚集查询问题。

关键词　数据流，层次窗口模型，在线聚集，近似查询

Data Stream Hierarchical Windows Model and Aggregate Query Algorithm

LIU Qing-Bao　JING Yan　HOU Dong-Feng　ZHANG Wei-Ming

(College of Information System and Management, National University of Defense Technology, Changsha 410073)

Abstract　A multi-hierarchical time windows model is proposed, which can describe the data stream with multiple granularities at different time. In order to query the aggregations of data stream effectively under the condition of limited space-time expense, a multi-granularity aggregate tree data structure is designed, and the query algorithm based on it is given.

Keywords　Data stream, Hierarchical windows model, Online aggregate, Approximate query

1　引言

随着信息处理技术的发展，数据流的应用领域越来越广泛。典型的应用领域包括金融证券信息分析、交通运输监控、计算机网络安全、通信数据分析、个性化 Web 服务等等。特别是在军事应用方面，随着传感器技术的快速发展及其与计算机网络技术的有机结合，各种传感数据流已源源不断地汇聚到军事情报中心，迫切需要一种高效的手段从中获得及时的军事态势信息。这些应用需求给传统的数据管理和分析技术提出了严峻挑战。

目前数据流管理与分析处理技术已成为热点研究问题之一。Datar M 在文[1]中采用滑动窗口的概念对数据流进行了有效的建模，并展示了如何在滑动窗口上维持简单的数据统计。由于滑窗模型的简单性和直观性，基于滑窗模型的数据流处理技术很快得到了广泛研究。Babcock B 将链式采样方法和优先权采样方法应用到滑窗模型中[2]，并在文[3]中提出了在滑窗模型上维持方差和 k 平均聚类的方法。Zhang 等人[4,5]从传统的时空数据库技术角度出发研究了基于滑窗模型的数据流多粒度聚集问题[5]，提出对历史久远的数据采用粗粒度的存储方式，对于相对较近的数据采用细粒度的存储方式，但它所采用的 SB-tree 结构，存在部分空闲结点，空间利用率不高[6]。

本文在继承传统滑窗模型优点的基础上，提出一种多层次时间窗口模型，支持在不同时段对数据流进行不同粒度的建模，并设计出基于该模型进行快速聚集与近似查询的多粒度聚集树结构。同时，给出了基于多粒度聚集树结构的近似聚集查询算法，从而有效地解决了在有限时空条件下的数据流聚集查询问题。

2　多层次时间窗口模型

定义 1　设给定时间域 T 和关系 $TG: T\rightarrow 2^T$，$T=\bigcup_{1\leqslant i\leqslant n} g_i, n\in N$，其中 N 表示自然数集，则称每一个 g_i 为一个时间粒子，映射 TG 是时间域的一种粒度。在不引起混淆的情况下，可以直接用 $G=\{g_i \mid 1\leqslant i\leqslant n\}$ 表示一种粒度。

定义 2　把时间窗口分成 m 个层次，令顶层窗口的长度为最长的时间段 T_m，时间区间为 $[t_{now}-T_m, t_{now}]$。最底层窗口的长度是为最短的时间段 T_l，区间为 $[t_{now}-T_l, t_{now}]$，其他窗口分别为 $[t_{now}-T_i, t_{now}]$，其中 $i=2,3,\cdots,m-1$，我们把这样划分的每个层次的窗口依次表示为 $W_1, W_2, \cdots, W_m$，每个层次的窗口所对应的时间粒度分别为 $G_1, G_2, \cdots, G_m$，则称三元组序列 $\xi=\{(W_i, T_i, G_i)\}(i=1,2,\cdots,m)$ 为多层次时间窗口模型。其示意图如图 1 所示。

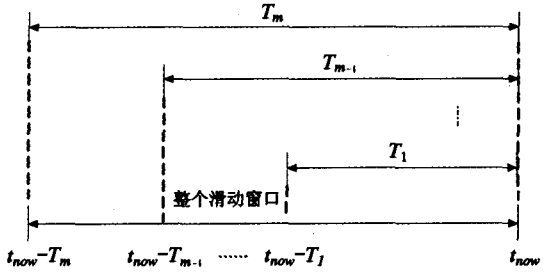

图 1　多层次时间窗口模型示意图

如果把各个层次分开考察，每个层次的窗口都是在对应时间粒度上的滑动窗口。所不同的是多窗口之间存在按层次的时间粒度聚集关系。第一个窗口 W_1 是包含数据信息最详细的窗口，后面的窗口都是在前一个窗口的基础上进行时间粒度聚集而形成的，从而达到对数据流的多时间粒度建模。

多层次时间窗口模型与多维数据模型中的维层次结构相一致，基于该模型的聚集操作与多维数据模型中的上钻操作(Rollup)也相吻合，这后续的数据流联机分析处理研究提供了可能。但两者之间仍然有很大的差别：(1)在数据仓库中，虽然也存在粗粒度的综合数据粒子，但所有的细节数据粒子

*)资助项目：国家自然科学基金(60172012)。刘青宝　副教授，主要研究方向为数据仓库技术和数据挖掘；金　燕　硕士研究生，主要研究方向为联机分析与数据挖掘；侯东风　博士研究生，研究方向为数据流管理与挖掘；张维明　博士生导师，教授，主要研究方向为军事信息系统，信息综合处理与辅助决策。

都完整地保存着，能精确地回答各种粒度的数据查询，而多层次时间窗口模型所保存的数据粒子数目有限，没保存整个时间域的所有粒子，只保存近期的有限数目的数据粒子；(2)数据仓库中的聚集计算一般是由用户的上钻操作引起的，而在多层次时间窗口模型中，数据粒子由细变粗是随数据流的数据时间戳由新变旧而在线聚集的。

多层次时间窗口模型主要有下面几个关键因素：

(1)关于各个层次的窗口大小：要求高层窗口的时间跨度要长于低层窗口的时间跨度。为了便于后面进行基于多层次窗口的时间聚集与查询操作，这里规定各层窗口的大小为(α^l+1)个粒子的容量($\alpha>2, l>1$)；

(2)关于多时间粒度关系定义：各层次的粒度之间得选择的合适，如果两个相邻层次的粒度之间粗细粒度差别较大则会造成一种"断层"的现象。为此，规定下层的窗口容量至少保证可以聚集出一个上层的数据粒子，从而使得数据流可以平滑按层次聚集，不至于造成系统的"抖动"；

(3)关于最低层窗口的基本窗口宽度：在最低层滑窗中，采用文[7]中定义的基本窗口作为最低层滑窗更新处理的缓冲区。基本窗口大小一般设置范围为：大于最低层窗口的一个时间粒子，小于第二层窗口的一个时间粒子。

3 数据流多粒度聚集树结构

在数据流聚集计算的数据结构研究方面，树结构是最典型的一种支持快速聚集与在线维护的数据结构[4~8]。这里提出的数据流多粒度聚集树结构 DSMGA-Tree(Data Stream Multi- Granularity Aggregate Tree)，是基于多层次时间窗口模型的一棵非完全树，树的每个层次对应一个层次的滑动窗口，而且各层节点的最大扇出数由相应层次窗口的时间粒度决定。

设多层次时间窗口模型为三元组序列 $\xi=\{(W_i, T_i, G_i)\}$ $(i=1,2,\cdots,m)$，其中时间窗口分成 m 个层次，则对应的数据流多粒度聚集树结构 DSMGA-Tree 描述如下：

(1)树的非叶结点

非叶结点为树的第 $i(1<i\leqslant m)$ 层结点，其信息结构为 $\{V_{agg}, [t_s, t_e), [ptr_1, ptr_2, \cdots, ptr_k], lptr\}$，其中 V_{agg} 为结点的聚集值集合，可以记为 $V_{agg}=\{V_{SUM}, V_{COUNT}, V_{MAX}, V_{MIN}, \cdots\}$；$[t_s, t_e)$ 为结点的时间区间，t_s 为起始时间，t_e 为结束时间，$I=t_e-t_s$ 是由第 i 层时间窗口的粒度 G_i 决定。若粒度 G_i 为分钟，则 $I=t_e-t_s$ 为 1 分钟；$[ptr_1, ptr_2, \cdots, ptr_k]$ 为指向子结点的指针数组，k 结点的扇出度，它由第 i 层时间窗口和第 $i-1$ 层时间窗口之间的粒度聚集关系决定，如 1 个第 i 层的时间粒子由 4 个第 $i-1$ 层的时间粒子聚集而成，则 k 为 4；$lptr$ 为指向其左兄弟结点的指针，若左兄弟不存在或已流出第 i 层时间窗口，则 $lptr$ 为空指针 $Null$。

(2)树的叶结点

树的叶结点为最底层的结点，其信息结构为 $\{V_{agg}, [t_s, t_e), lptr\}$。

(3)尾指针组

尾指针组 $\{tail_i\}$ 为指向每层中具有最新结束时间的结点。

(4)各层结点数目

树的第 $i(1\leqslant i\leqslant m)$ 层结点数目等于第 i 层窗口的时间粒子数目。

数据流多粒度聚集树结构 DSMGA-Tree 具有以下性质：

(1)若非叶结点 $Node_i$ 的子结点指针 $\{ptr_i\}(1\leqslant i\leqslant k)$ 非空，则它们所指向的子结点序列具有时间可加性，而且它们的时间区间求和等于结点 $Node_i$ 的时间区间 $Node_i[t_s, t_e)$。

(2)非叶结点 $Node_i$ 的子结点指针，在以下两种情况可以为空：一是当其所指向的子结点流出下层窗口时，则子结点指针被清空；二是子结点还没有生成，随着数据流的不断推进，将会指向新产生的子结点。

(3)DSMGA-Tree 是一棵非平衡的树，新的叶结点总是从最右端的尾指针处插入。

4 在线聚集与近似查询算法

在很多实际应用中，快速给出近似的、大体的查询结果是数据流处理的核心工作。

4.1 数据流在线聚集算法

多层次窗口模型的数据聚集过程是与数据流的在线更新过程同步进行的，其具体过程描述如下：

(1)窗口 W_1 的更新

W_1 包含的是最近 T_1 时间段内的数据，对应的是基粒度 G_1，也是数据流分析中最详细的数据信息，例如，网络监控系统中每秒钟产生 200 条记录，而监控数据流分析应用中最详细粒度为秒，则需对每秒钟产生的 200 条记录进行粒度化聚集计算，形成以秒为时间粒度的基本数据粒子。设 W_1 的基本窗口序列为 $S_1[0], S_1[1], \cdots, S_1[k_1-1]$，并约定 $S_1[k_1]$ 为即将到达的基本窗口，$S_1[0]$ 为即将过时的基本窗口，则当新的数据到达时，在 $S_1[k_1]$ 中缓存，进行粒度化聚集计算，当 $S_1[k_1]$ 充满时，加入到 W_1 中。若 W_1 窗口已满，则把最老的基本窗口 $S_1[0]$ 移出 W_1 窗口。

(2)窗口 $W_{i+1}(i=1,3,\cdots,m-1)$ 的更新

在窗口 W_i 中，按照上层窗口 W_{i+1} 的粒度对窗口 W_i 中的数据粒子进行分组。当每组的第一个粒子到达时，创建新数据粒子 $newNode_{i+1}$，并把 $newNode_i$ 插入上层窗口 W_{i+1} 中。依此类推，直到窗口 W_m。

(3)聚集信息的生成

在数据流处理的过程中，涉及到聚集信息的生成。而且聚集信息与所维持的语义是紧密相关的，为了说明简便，这里仅以平均值为例进行介绍。

不失一般性，假设窗口 W_i 中的数据粒子为 $b_1, b_2, \cdots$，并假设由粒子 b_i 到 $b_j(1\leqslant i<j\leqslant k)$ 聚集成 W_{i+1} 中的数据粒子 z。首先取 z 所包含的第一个数据粒子 b_i 的时间戳，作为 z 的时间戳。其次计算元组的个数(*COUNT*)和累计求和(*SUM*)，由于聚集运算 *COUNT* 和 *SUM* 都具有可加性，因此可以将数据粒子 b_i 到 $b_j(1\leqslant i<j\leqslant k)$ 的相应聚集值进行累加性操作，从而得到数据粒子 z 的聚集值 *COUNT* 和 *SUM*，这就可以计算出数据粒子 z 的平均值。

下面给出基于多粒度聚集树的数据流在线聚集算法 OLAgg(On Line Aggregate)的伪代码描述：

```
OLAgg(Node newNode_i, Pointer tail_i)
//tail_i 为 DSMGA-Tree 树的第 i 层尾结点
//newNode_i 待插入到第 i 层的新结点
BEGIN
 IF(第 i 层窗口已满) THEN
  清除第 i 层窗口中最老的结点;
 END IF;
 newNode_i.lptr=tail_i.lptr;
 tail_i=newNode_i;
 //插入到第 i 层窗口中
 IF(newNode_i 是第 i+1 层结点的首子结点) THEN
  创建第 i+1 层的新结点 newNode_{i+1};
  newNode_{i+1}[t_s,t_e)=newNode_i.[t_s,t_e);
  newNode_{i+1}.ptr_1=tail_i;
  newNode_{i+1}.V_agg=newNode_i.V_agg;
  OLAgg(newNode_{i+1},tail_{i+1});
  //采用递归方式,进行第 i+1 层新结点的插入
ELSE IF (是第 i+1 层尾结点的第 j 子结点) THEN
  //1<j
  Node_{i+1}=*tail_{i+1};
  Node_{i+1}.ptr_j=tail_i;
  Node_{i+1}.V_agg=Node_{i+1}.V_agg⊕newNode_i.V_agg;
```

$Node_{i+1}.t_e = newNode_i.t_e$
END IF;
END OLAgg.

4.2 数据流近似聚集查询算法

在聚集查询中通常会涉及到对数据流的原始历史数据，但因空间有限，所采用的多层次窗口模型只保存了历史数据的概化信息，从而给聚集查询带来了一定的非精确性。通过分析可知，产生误差的主要原因为：对距离当前时间比较久远的时段，只保留了较高层次的粗粒度聚集信息，若想查询细粒度的结果，则需要进行估计。在估计的过程中，采用的是局部区域内均匀分布的假设，即区域内的每个个体对总体的贡献是相同的。这种假设在数据流局部特性比较稳定的情况下，所产生的误差相对较小。

不失一般性，这里以 SUM 聚集查询为例，以伪代码描述方式，介绍基于多粒度聚集树的数据流近似聚集查询算法 AAQuery(Approximate Aggregate Query)如下：

```
AAQuery(Pointer root, Scope Tq, Value aggv)
//root 为 DSMGA-Tree 树的惟一根结点，
//Tq=[tqs,tqe],为查询时间段，
//aggv 用于返回查询结果，初始为空值。
BEGIN
 Node=*root;  //设 Node 为根结点
 IF (Tq=Node.[ts,te)) THEN
   aggv=aggv+Node.Vagg;
   RETURN;  //返回精确的聚集值
 ELSE IF (Tq∩Node.[ts,te)=∅) THEN
  RETURN;
 END IF;
 IF(Node 无子结点)THEN
 //Node 是叶结点或无子结点的非叶结点
   aggv=aggv+(Node.vagg×Tq∩Node.[ts,te) / Node.[ts,te));
   RETURN; //返回近似的聚集值
 END IF;
 IF(Node 的第一个子结点指针为空)THEN
   α=(Tq∩(Node.[ts,te)−⋃(ptrj≠Null) Node.ptrj→[ts,te)) / Node.[ts,te));
   aggv=aggv+(Node.Vagg×α);
  END IF;
 FOR EACH {Node.ptri|ptri≠Null,1≤i≤k} DO
  AAQuery (Node.ptri,Tq,aggv);
  //对每个非空子结点进行递归查询
 END FOR;
 RETURN;
END SAAQuery.
```

由于 DSMGA-Tree 树是全时间(从数据流的起始时刻到当前时刻)的多粒度聚集树，所以查询时间区间是非受限的。但因为对较久远的数据只保留了很高层次的粗粒度聚集信息，当查询时间区间包含较久远的时段时，查询结果较为粗糙。

5 算法分析

这里对数据流处理的三个关键指标进行分析：存储空间、处理时间和查询结果精度。

5.1 存储空间

设数据流原始数据的基本粒子数为 N，多粒度聚集树 DSMGA-Tree 中上层结点最少由 α 个下层结点聚集而成，则树的高度小于 $\log_\alpha N$。若进一步假设每层最多保存 α^l+1 个结点，则整个树的结点数小于 $(\alpha^l+1)\times\log_\alpha N$。

例 1 在持续 100 年的数据流中，设 $\alpha=2, l=2$，且最小粒度为秒，则所需的空间为：$(2^2+1)\times\log_2(100*365*24*60*60)\approx160$ 个结点的存储空间，是远远少于最底层窗口流出的 $100*365*24*60*60=3153600000$ 个过期数据结点。可以看出，多粒度聚集树 DSMGA-Tree 的存储空间非常小，很适合在内存中进行存储和计算。

5.2 处理时间

数据流处理时间分为两部分：在线聚集时间和聚集查询时间。

(1)在线聚集时间

由数据流在线聚集算法 OLAgg(On Line Aggregate)的描述可知，当一个新基本数据粒子到来时，最极端的情况是需在树的各个层次上插入新结点，而由小节 5.1 可知树的高度小于 $\log_\alpha N$，所以在线聚集时间的复杂度是 $O(\log_\alpha N)$。

(2)聚集查询时间

数据流近似聚集查询算法采用递归方式检索与查询区间有交的结点及其子树，在查询区间较小时，检索的结点数较少，一般无需遍历整个树，速度很快。最极端的情况是遍历整个树，由于树的结点数小于 $(\alpha^l+1)\times\log_\alpha N$，所以时间复杂度为 $O(\log_\alpha N)$。

5.3 查询结果精度

由于数据流规模大、速率快，在有限存储空间和一遍式扫描的时间要求下，难以得到准确的聚集查询结果。为了估计查询结果的近似程度，先给出如下定理：

定理 1 设多粒度聚集树 DSMGA-Tree 的上层结点都由 α 个下层结点聚集而成，每层最多保存 α^l+1 个结点，并假设用户指定的查询窗口为 $W[t_c-h,t_c]$，其中，t_c 为当前时刻，h 为用户查询窗口宽度，若 t_s 为刚好在 t_c-h 之前的某结点的起始时刻，则有 $(t_c-t_s)\leqslant(1+1/\alpha^{l-1})\times h$。

证明：设 i 满足下列条件：

$(\alpha^i>h/\alpha^{l-1})\wedge(\alpha^{i-1}<h/\alpha^{l-1})$

由于在树 DSMGA-Tree 的第 $i-1$ 层具有 α^l+1 个结点，所以总是可以找到一个在 t_c-h 之前的结点，设 t_s 为刚好在 t_c-h 之前的那个结点的起始时刻，即 $(t_c-h)-t_s\leqslant\alpha^{i-1}$。有 $(t_c-t_s)\leqslant(h+\alpha^{i-1})<(1+1/\alpha^{l-1})\times h$，定理得证。

由上述定理可知，当给定用户查询窗口 $W[t_c-h,t_c]$ 时，总能在误差不超过 $(1+1/\alpha^{l-1})\times h$ 范围内找到聚集运算的时间起始点。

小结 本文提出了一种多层次时间窗口的数据流建模方法，设计了支持在线聚集与近似查询的多粒度聚集树结构，同时，给出了基于该树结构的聚集与查询算法，从而有效地解决了在有限时空条件下的数据流聚集查询问题。但是目前的研究仍存在很多需要改进的地方，首先，在模型中没有对全局的粒度关系进行严格的规范；其次，该模型缺乏对流出窗口的历史数据粒子的考虑，难以支持具有分组约束的复杂聚集查询。这些都将是我们下一步要重点研究的内容。

参 考 文 献

1 Datar M, Gionis A, Indyk P, Motwani R. Maintaining stream statistics over sliding windows. In: Proc. of the 2002 Annual ACM-SIAM Symp. on Discrete Algorithms, 2002. 635～644
2 Babcock B, Datar M, Motwani R. Sampling from a moving window over streaming data. In: Proc. of the 2002 Annual ACM-SIAM Symp. on Discrete Algorithms, 2002. 633～634
3 Babcock B, Datar M, Motwani R, O'Callaghan L. Maintaining variance and k-Medians over data stream windows. In: Neven F, ed. Proc. of the 22nd ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems. San Diego: ACM Press, 2003. 234～243
4 Zhang D, Gunopulos D, Tsotras VJ, Seeger B. Temporal aggregation over data streams using multiple granularities. In: Jensen CS, Jeffery KG, eds. Proc. of the 8th Int'l Conf. on Extending Database Technology. LNCS, 2002. 646～663
5 Yang J, Widom J. Incremental Computation and Maintenance of Temporal Aggregates. Proc. of ICDE, 2001
6 张冬冬，李建中，王伟平，郭龙江. 数据流历史数据的存储与聚集查询处理算法. 软件学报，2005,16(12)
7 Zhu Y, Shasha D. StatStream: Statistical monitoring of thousands of data streams in real time. In: Proceedings of the 28th International Conference on Very Large Data Bases, 2002. 358～369
8 Ahmet. Bulut. SWAT: Hierarchical Stream Summarization in Large Networks. Proc. of ICDE, 2003
9 刘青宝，戴超凡，邓苏，张维明. 基于网格的数据流聚类算法. 计算机科学(已录用)