SPQ:数据流上面向可伸缩模式的查询方法

李菲菲 李红燕 曲 强 苗高杉

(北京大学信息科学技术学院 北京 100871)

(北京大学机器感知与智能教育部重点实验室 北京 100871)

摘 要 数据流的模式查询具有很高的领域价值,它不仅需要较高的抗噪能力和实时性,而且查询目标模式还具有可伸缩性,即由多个子模式复合而成,且某些子模式可重复、缺失或倒置.文中提出一种可伸缩模式的查询(SPQ)方法,允许用户定义目标模式并设置可伸缩条件.然后在查询处理中通过模式匹配生成模式流,进而完成满足可伸缩条件的目标模式查询.在真实数据集上进行的实验从查全率、查准率和处理效率上证明了 SPQ 方法是可行和灵活的.

关键词 数据流;查询;可伸缩模式;目标模式;查询重写;查询处理

中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2010.01481

SPQ: A Scalable Pattern Query Method over Data Streams

LI Fei-Fei LI Hong-Yan QU Qiang MIAO Gao-Shan

(*School of Electronics Engineering and Computer Science, Peking University, Beijing* 100871)

(*Key Laboratory of Machine Perception of Ministry of Education, Peking University, Beijing* 100871)

Abstract Pattern query over data streams possess high domain significance. It requires anti-noise capability and real time processing. Meanwhile, in many cases, the query target pattern is also scalable, which means it is comprised of sub-patterns, with some sub-patterns gained, lost or even inverse. This paper presents a scalable pattern query (SPQ) method. It allows users to define target pattern and set corresponding scalable constraints according to their knowledge and concerns, finally the target pattern can be changed to scalable pattern. In the stage of query evaluation, pattern stream is generated by pattern matching, and then scalable pattern query is carried out on the pattern stream. From the perspective of recall ratio, precision ratio and processing efficiency, the experimental results on real datasets show that SPQ is feasible and flexible.

Keywords data stream; query; scalable pattern; target pattern; query rewrite; query evaluation

1 引 言

数据流的应用愈发广泛和深入,为了在数据流上获取关注的信息,用户可以根据需求设置相应的查询.例如在重症监护病房(Intensive Care Unit, ICU),医护人员往往会设定一段有代表性的数据流序列,并关注其出现的位置或各种变化.在居民用电需求分析中,决策者会查询异常用电趋势的产生并及时做出决策.

收稿日期:2010-06-11.本课题得到国家自然科学基金(60673113,60973002)和国家"八六三"高技术研究发展计划项目基金(2007AA01Z191,2009AA01Z150)资助.**李菲菲**,女,1986 年生,硕士研究生,主要研究方向为数据流管理和挖掘. E-mail: alex6unica@gmail.com. **李红燕**(通信作者),女,1970 年生,教授,主要研究领域为数据管理与挖掘. **曲 强**,男,1985 年生,硕士研究生,主要研究方向为查询处理以及大规模图挖掘. **苗高杉**,男,1985 年生,硕士研究生,主要研究方向为数据流管理与挖掘.

在实际应用中,用户根据需求设置不同的查询,并允许查询内部的某些部分重复或缺失.这给数据流上的查询处理方法带来了很多挑战.为了满足日益复杂的查询需求,数据流上的查询应该具备以下特点:

(1)查询粒度较大,语义丰富.

多数的传统数据库查询技术关注元组(tuples).在数据流领域一些查询也基于元组.这种由数据流上连续数据点组成的模式具有更丰富的语义,其变化可以体现用户关心的变化趋势.如金融数据流中的 Zigzag 模式;又如图 1 所示的医疗数据流监控中的心电图(Electrocardiograph, ECG):一个完整的 ECG 周期由 P 模式、QRS 模式、T 模式、U 模式组成,反映了心脏跳动周期中的不同阶段.如 P 模式因心脏除极而产生,其变化决定了心房肥大的状况.

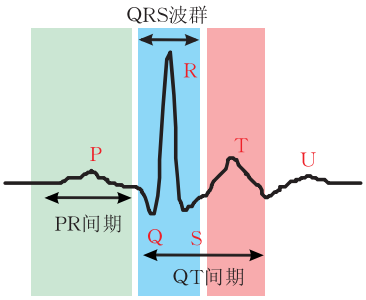
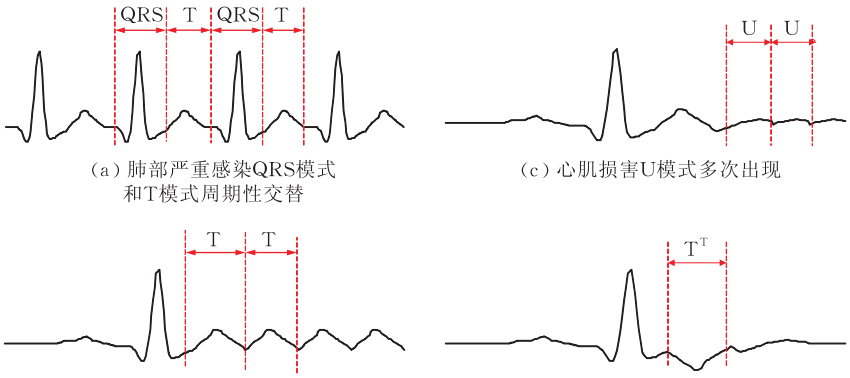


图 1 由多个子模式组成的标准 ECG 模式

(2)查询目标模式具有可伸缩性.

在线的数据流往往会无法预知地重复和缺失某些部分.因此查询目标模式应具有可伸缩性,即是由多个子模式复合而成的复杂模式.此处的可伸缩性指在保持时序关系的前提下,允许内部某些子模式重复、缺失,抑或是波形倒置.

以医疗数据流监测领域为例,如图 2(a)所示,QRS 模式和 T 模式周期性交替可能预示着肺部严重感染的产生.又如 P 模式缺失,T 模式出现多次可能会导致急性肾功能衰竭(图 2(b));心肌损害常由 U 模式多次出现预示(图 2(c));若 T 模式出现波形倒置(shape inversed),可能预示着有甲状腺功能减退性心脏病的危险(图 2(d)).



(a)肺部严重感染QRS模式和T模式周期性交替

(c)心肌损害U模式多次出现

(b)急性肾功能衰竭 P模式缺失,T模式多次出现

(d)甲状腺功能减退性心脏病 T模式倒置

图 2 查询目标模式的可伸缩性表达丰富语义

为了更加灵活地表达目标模式的可伸缩性,需要寻求一种即席查询(ad hoc query)的方法.

(3)抵抗无规律噪声的干扰.

现实环境中往往会有噪声的干扰.如医疗监测中病人的动作会触动传感器产生干扰数据.又如图 3 所示民用用电量曲线,突然的跳闸也是无法预知建模的.在一定程度的噪声下,要求算法可以较好地执行,并返回满足需求的查询结果.

(4)查询处理满足实时性要求.

以心电监护仪为例,其工作频率是 500Hz,每小时产生 1800000 条数据,一天的记录数将达到 43GB/台.巨大的数据量使得只能对数据流进行单遍扫描,需要实时地进行查询处理.

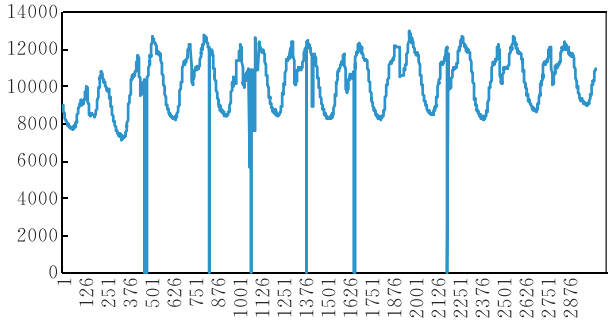


图 3 有噪声干扰的民用用电量曲线

以上是数据流上的查询应具备的一些特点.本文的研究重点是为用户提供数据流上灵活、方便的模式查询处理机制,提出了数据流上可伸缩模式的

查询(SPQ)方法. SPQ 方法允许用户根据查询需求定义目标模式并设置可伸缩条件. 为了避免匹配二义性生成基模式覆盖，并对目标模式进行查询重写形成 SPQ. 进而通过模式匹配形成模式流. 在模式流上的查询处理中，尽可能复用已处理结果来提高查询效率.

本文第 2 节简述相关研究；第 3 节介绍整体架构；第 4 节详述可伸缩模式查询表示；第 5 节描述可伸缩模式查询处理；第 6 节通过实验来分析方法的性能；第 7 节总结全文并展望下一步的方向.

2　相关研究

近年来有关数据流的研究受到了学术界和工业界很多关注. 与本文相关的工作大致可分为以下两类.

2.1　可伸缩模式匹配

一类与本文相关的研究是可伸缩模式匹配. 从技术上讲，大致分为基于时序数据库的模式匹配、基于树的模式匹配和基于图的模式匹配这 3 类.

第 1 种是基于时序数据库的可伸缩模式匹配方法. 代表有 Kahveci 的多长度索引结构 MR 方法[1]和 Kaghazian 等人提出的 RSPS 方法[2]. 前者的方法提出 MR 索引结构存储不同处理长度的数据信息，将查询切分成对应于 MR 的多个子查询并分别执行. 针对每个处理层面维护一个索引，空间复杂度较高，对数据库中的全部数据建立索引，时间复杂度为 $O(nt)$. 将这种方法应用到数据流领域是不现实的. 该方法中的可伸缩是查询长度可变，而非由某些部分重复或缺失导致的查询模式改变，故与本文中的可伸缩模式不同. RSPS 方法支持使用 SQL 表达可伸缩模式，即部分增加或缺失的复杂模式，并受 KMP 算法启发计算 shift(j)和 next(j)来发现元素间的内在联系，减少对相同数据的多次扫描. RSPS 虽然减少了扫描次数，但计算和存储转移信息消耗较大，为 $O(m^2)$. 由于数据流只是一遍流过，故该方法并不能很好地应用于数据流领域.

第 2 种是基于树的可伸缩模式匹配方法. Mei 和 Madden 提出的复合事件处理模型 ZStream[3]是其代表. 它将查询操作封装为事件操作符，支持可伸缩模式的查询，使用基于树的查询计划表示查询模式，利用代价模型评估最优计划. 但 ZStream 以事件为粒度，其处理对象是由数据源直接产生的，不支持其内部细分，因此无法处理事件间的包含关系. 对应到数据流的可伸缩模式处理，则无法解决 SPQ 中目标模式内部各子模式间的相似性包含问题. 这是因为 SPQ 以基模式为粒度，比事件的粒度要细. 基模式不是数据源直接产生的，是由用户划分、后台处理而得的. 另外，ZStream 的树结构存储也带来额外的维护代价.

第 3 种是基于图的可伸缩模式匹配方法. NFA 是很典型的方法. 由于转移状态不确定，匹配不成功时会产生大量回溯，因此传统的 NFA 方法是离线的. 为了提高效率，改进方法[4-7]逐步产生，其中一些还被应用于事件流领域，支持可伸缩模式查询. 改进主要是减少回溯[4-6]和引入并行处理[7]. Agrawal 提出了 NFA^{b}的方法[5]，引入 singleton 和栈进行可伸缩模式匹配. 每次只处理一个执行过程称之为 singleton，此过程一直执行直至 singleton 匹配成功或失败，其间产生的其它匹配分支点均存入栈中. 当匹配失败时，从栈中弹出最近的分支点作为新的 singleton 进行匹配. 此方法也是以直接产生于数据源的事件为粒度，未处理事件间的包含关系. 另外当模式比较复杂时，其分支点会比较多，使得匹配次数增加. 另一类方法是将 NFA 转换为多个较小的 NFA 或者对应的 DFA 后采用并行处理的方法. 但是此方法会随着查询模式的复杂性增加而形成大量的 NFA 或 DFA，其效率很大程度上取决于查询目标模式的复杂程度.

2.2　数据流管理系统(DSMS)中的查询处理

很多查询处理技术被用来实现数据流管理系统(Data Stream Management System，DSMS). 比较著名的如 STREAM 系统[8]、TelegraphCQ[9]、Aurora 系统[10]、Cougar 项目[11]、Hancock 项目[12]等. DSMS 大都着眼于在近似实时的前提下，完成对数据流的预定义查询. 但其查询粒度是元组，不支持模式的整体查询. 正如 Papadimitriou 等人所述，这些系统的共同缺陷在于仅着眼于在数据流上完成传统的查询，没有去寻找数据的特征，更无法完成数据流上的模式查询[13].

3　整体架构

图 4 展示了可伸缩模式查询的整体框架. 主要由查询表示和查询处理两阶段组成. 在查询表示中，用户通过图形用户界面在原始数据流上选择并定义查询目标模式，设置相应的可伸缩条件；根据目标模式，通过基模式生成器生成基模式字典和基模式覆

盖;进而查询重写模块以基模式为粒度,将目标模式转为可伸缩模式.在查询处理中,基于基模式的匹配模块将不断到达的数据点与基模式并行比较,从而生成模式流;把可伸缩模式查询应用于模式流,通过可伸缩模式查询执行器完成查询处理;将结果通过查询结果展示模块以标记数据流的方式返回给用户.

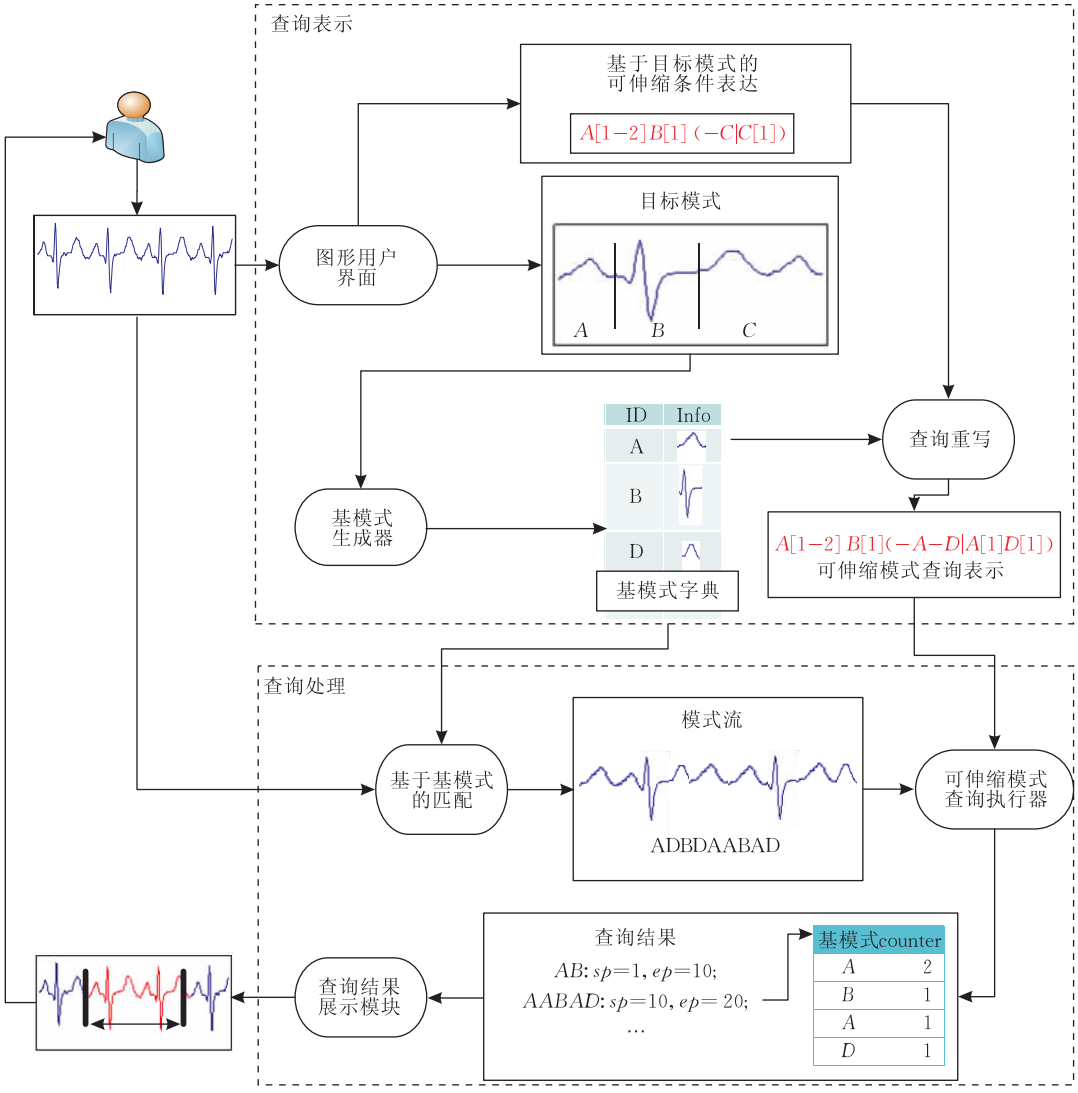


图 4 可伸缩模式查询整体架构

4 可伸缩模式查询表示

用户首先根据查询需求设定模式和相应的可伸缩条件,形成目标模式;为避免子模式间的相互包含关系导致的匹配二义性,引入基模式和基模式覆盖;进而基于基模式对目标模式进行查询重写,生成SPQ.

4.1 目标模式

本文为用户提供一种即席查询的机制.允许他们结合领域知识和关注重点,选择富含语义的子模式并设置相应的可伸缩条件,从而给定查询目标模式.这种方法提高了模式查询的灵活性,更好地考虑了用户的经验和需求.但是用户直接选择的模式可能会出现如图 5 所示的子模式间部分重叠的情况.

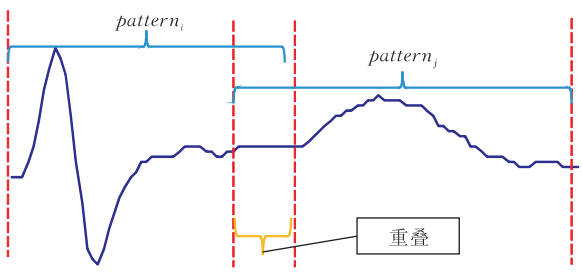


图 5 子模式间的重叠

它会带来二义性，例如用户希望查询子模式 i 出现 3 次，且子模式 j 不出现的情况，那么将无法判断图 5 的重叠部分究竟是重复出现 3 次还是缺失. 因此本文涉及的查询目标模式内部的各子模式间不允许重叠，即它们是不重叠模式，如定义 2 所示.

定义 1.　数据流. 令 S 代表数据流，其中 X_i 表示时间戳为 t_i 时的数据值，$1\leq i\leq n$，则 S 可以如下表示：

$$S=\{(X_1,t_1),(X_2,t_2),\cdots,(X_i,t_i),\cdots,(X_n,t_n)\}.$$

定义 2.　不重叠模式. 令 p_i 表示第 i 个子模式. 不重叠模式具有如下特征：$\forall\, 1\leq i\leq m$，$1\leq h\leq m$，$i\neq h$，子模式 p_i 和 p_h 满足 $t_{ik_i}<t_{h1}$ 或 $t_{hk_h}<t_{i1}$ 条件，其中

$$p_i=\{(X_{i1},t_{i1}),(X_{i2},t_{i2}),\cdots,(X_{ik_i},t_{ik_i})\},$$

$$p_h=\{(X_{h1},t_{h1}),(X_{h2},t_{h2}),\cdots,(X_{hk_h},t_{hk_h})\}.$$

用户设定不重叠模式后，根据查询需求为其设置相应的可伸缩条件(见表 1)，进而形成目标模式，用以灵活地表达丰富的查询语义.

定义 3.　目标模式. 令 tp 表示目标模式，c 表示可伸缩条件，则目标模式可以表示为 $tp=\bigcup_{i=1}^{m}p_i[c_i]$，其中共 m 个不重叠模式，c_i 为 p_i 对应的可伸缩条件.

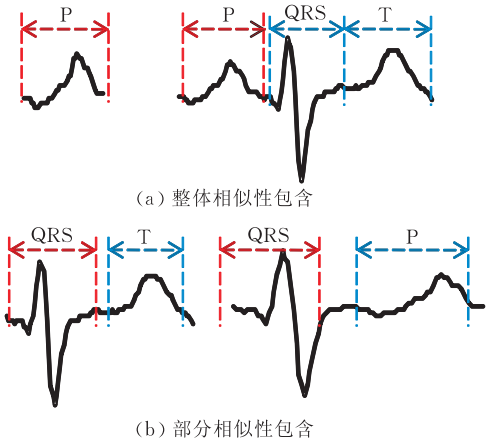
由定义 3 知用户选定连续的区段，继续在其上定义子区段作为不重叠模式，并设置相应可伸缩条件. 可见目标模式是由若干个子模式复合而成的，它允许内部某些子模式重复、缺失或倒置，表达语义更复杂.

表 1　SPQ 方法中的可伸缩条件

类型	符号表示	解释
重复	$[m]$	m 个应被匹配；
	$[m,n]$	m 或 n 个应被匹配；
	$[m-n]$	m 至 n 个被匹配均可；
	$[m-]$	至少 m 个被匹配，m 为下界；
	$[-n]$	至多 n 个被匹配，n 为上界；
缺失	$-P$	P 模式不出现；
倒置	P^{T}	P 模式波形倒置，180°反转；
或	$A\mid B$	A 出现或 B 出现

4.2　基模式覆盖

在用户定义的目标模式中，不重叠模式仅仅保证了物理上各模式间是没有重叠部分的，但并不能保证彼此之间逻辑上可以很好地区分，即并未消除模式间可能潜在的相互包含关系. 不重叠模式间的潜在相似性包含关系如图 6 所示，(a)中相似于左边的模式整体又出现在右边目标模式中，(b)中两个模式都包含了相似的一个组成部分. 随着数据流源源不断地到达，匹配可能会出现二义性. 这主要是由于不重叠模式间存在着相似性包含关系，在匹配时它们都在误差阈值范围之内，导致多个匹配结果的出现，无法正确返回查询结果. 因此本文提出的 SPQ 方法引入了基模式(base pattern)和基模式覆盖(base pattern cover)的概念，如定义 4 和 5 所示，用于保证不重叠模式彼此间不存在包含关系，在逻辑上是彼此可区分的.



(a) 整体相似性包含

(b) 部分相似性包含

图 6　不重叠模式间的相似性包含关系

定义 4.　基模式. 基模式除了具有不重叠模式的特征外，还有如下特征：

$$\forall\, bp_i,bp_h\in \mathrm{Cover}(bp),\ i\neq h,$$

$$bp_i=\{(X_{i1},t_{i1}),\cdots,(X_{ij},t_{ij}),\cdots,(X_{ik_i},t_{ik_i})\},$$

$$bp_h=\{(X_{h1},t_{h1}),\cdots,(X_{hj},t_{hj}),\cdots,(X_{hk_h},t_{hk_h})\},$$

$\exists\, t_{h1}\leq t_{hp}\leq t_{hk_h}-(t_{ik_i}-t_{i1})+1$(假定 $t_{hk_h}>t_{ik_i}$)，对于 $subp=\{(X_{hp},t_{hp}),\cdots,(X_{hp+k_h-1},t_{hp+k_h-1})\}$，使得 $similarity(subp,bp_i)<\varepsilon$，其中 ε 是相似度衡量误差阈值.

由定义 4 可知基模式本质上仍是不重叠模式，但基模式彼此间可相互区分，不存在包含关系.

定义 5.　基模式覆盖. 存在时间戳从 t_{w_1} 到 t_{w_2} 跨度的目标模式，则对于 r 个基模式，基模式覆盖具有如下特征：

$$\bigcup_{q\in\{1,2,\cdots,r\}}bp_q\approx\bigcup_{i=w_1}^{i=w_2}\{(X_i,t_i)\}.$$

由定义 5 可知，所有的基模式按目标模式中时序顺序首尾相连能够近似重构回目标模式起止时间戳所跨越的范围.

生成基模式及其覆盖算法的主要思想是将不重叠模式两两比较. 若存在包含关系，则将包含部分拆分出来. 由于新拆分出来的子模式会影响当前所有模式的比较，本文对模式按大小进行排序，如算法 1 所示，每次判断相似性包含关系时，都用较小的探针模式(probe pattern)去比较或拆分较大的目标模式

(target pattern),以保证任意两个基模式间的可区分性;同时维护当前比较记录来提高效率.若两模式尚未比较则调用算法 2 来判断是否存在包含关系,若存在则拆分成可相互区分的基模式,其间优化新产生基模式的插入位置,用来减少比较次数和保证准确性.

算法 1. Base patterns generation(基模式生成).

输入: tp, error bound Eb_p

输出: Dictionary(bp), Cover(bp)

中间变量: probe index i, target index j, Set(bp);

1. Initialize Set(bp) with tp, and order Set(bp) in ascending sequence;
2. for $i<$Set(bp).size do //compare with other bp
3. $j=i+1$;
4. for $j<$Set(bp).size do
5. if bp_i and bp_j have already been compared
6. j++, move to next target bp_j of Set(bp);
7. else bp_i and bp_j haven't been compared
8. $i,j\leftarrow$Pattern split(bp_i,bp_j,Set(bp),Eb_p);
9. if i does not change then i++;
10. generate Dictionary(bp) and Cover(bp) due to Set(bp).

算法 2. Pattern split(bp_i, bp_j, Set(bp), Eb_p)(模式拆分).

输入: bp_i, bp_j, Set(bp), Eb_p

输出: Set(bp), probe index i, target index j

中间变量: *insertIndex*, *compareResult*;

1. *compareResult*$\leftarrow$Similarity Compare(bp_i,bp_j,Eb_p), and keep the compared record;
2. if *compareResult* indicates bp_i isn't similar to bp_j
3. j++; //move to next bp(base pattern)
4. else split bp_j with *compareResult*,
5. replace bp_j by bp_{new} in ascending order;
6. record all the insert indexes;
7. *insertIndex*$\leftarrow$*find* minimum insert index;
8. if *insertIndex*$>i$ then j++;
9. else if *insertIndex*$<i$ then //disturb record
10. i=*insertIndex*, break to start a new pass;

算法 1 调用算法 2 作为关键步骤,考虑目标基模式含 k 个点,探针基模式含 h 个点,则算法 2 的关键方法 Similarity Compare 决定了其时间复杂度为 $O(h(k-h))$.若共有 m 个不重叠模式,此处假设平均每次比较后进行一次拆分,增加一个新的模式.则算法 1 的时间复杂度为 $O(mh(m-1)(k-h))$.实际应用中一般不会达到上述假设情况,通常拆分的次数并不太多,而且算法 1 和 2 对算法的优化也提高了效率.此处复杂度虽然不一定能满足线性要求,但是保证了生成基模式的准确性;并且此过程只是根据目标模式执行一次即可.所以为了后续查询处理的精度、生成准确的基模式字典,以一定时间换精确度是可以接受的.

4.3 基于基模式的查询重写

产生基模式后,可以对表达查询需求的目标模式进行查询重写.这种重写是基于基模式粒度的,不但表达灵活,而且在后续查询中也可避免匹配二义性的问题.通过对目标模式进行基于基模式的查询重写,可以形成可伸缩模式查询(SPQ).

定义 6. 令 sp 表示可伸缩模式,mc 表示可伸缩条件,共 r 个基模式,则可伸缩模式表示如下:

$$sp=\sum_{i\in\{k|k=1,2,\cdots,r\}} bp_i[mc_i],$$

其中 bp_i 为基模式,mc_i 为其对应的可伸缩条件.

可伸缩模式查询是对目标模式的查询重写.它基于基模式的粒度.SPQ 同样允许内部的某些基模式重复、缺失或倒置,处理粒度相对更细一些.而这种查询重写首先要将可伸缩条件从目标模式中的不重叠模式映射到可伸缩模式中的基模式上.由同一个不重叠模式拆分出的基模式要共享相同的可伸缩条件,且这种查询重写的过程对用户是透明的.具体如算法 3 所示.例如目标模式为 $A[1-2]B[1](-C|C[1])$,且 C 可以拆分为 A 和 D,所以基模式为 A、B、D,则 SPQ 可以表示为 $A[1-2]B[1](-A-D|A[1]D[1])$.

算法 3. SPQ generation(可伸缩模式查询生成).

输入: Dictionary(bp), Cover(bp), tp;

输出: scalable pattern sp;

1. for every bp in Cover(bp)
2. if it corresponds to p in tp
3. bp.*constraints*$\leftarrow p$.*constraints*;
4. update sp with bp and scalable constaints;

若目标模式中有 m 个模式,通过算法 1 和 2 生成 r 个基模式,一般情况下 r 稍大于 m.算法 3 要浏览一遍 Cover(bp),则其时间复杂度为 $O(kr)$,其中 k 为常数.

5 可伸缩模式查询处理

可伸缩模式查询处理可以根据已生成的基模式字典在数据流上自动执行.首先对依次到达的数据点并行判断连续点组成的轨迹是否满足任一基模式,从而完成基于基模式的匹配,形成模式流;进而对模式流中的基模式按时序顺序出现的情况进行计数,尽可能复用已处理的结果,实现基于模式流的查询.

5.1　基于基模式的匹配

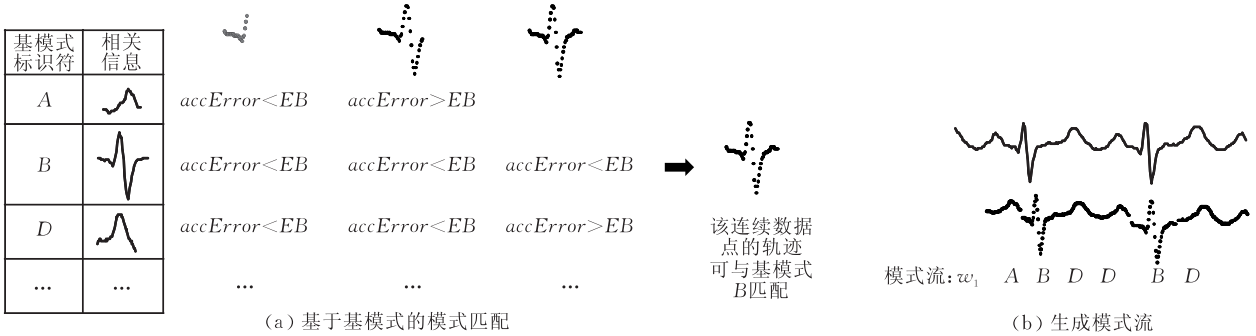
在数据流应用领域，数据点源源不断地到来，连续点的轨迹可以显示出这些点的大致走向与趋势. 受此启发，每到达一个数据点，SPQ 方法就将其与基模式字典中的所有基模式同步比较，即并行地判断点的轨迹与哪个基模式相似，可以匹配成功.

在并行匹配过程中，随着数据点不断地到达，连续点的轨迹与基模式并行比较会形成多个分支，分支停止情况大致有以下 3 种关系：

(1) 与某一基模式的比较尚未结束，但是累积误差已经超过误差阈值，及时停止该分支比较；如图 7(a)所示，其中 EB 为误差阈值. 最开始与所有基模式的比较都在误差范围内，随着数据点不断地到达，点的轨迹与基模式 A 相似度超过误差阈值，停止与 A 的比较，其它分支继续并行比较；

(2) 与某个基模式比较结束，且累积误差满足阈值要求，连续点的轨迹很快收敛到该基模式的波形，说明匹配成功，停止其它分支比较：如图 7(a)所示，与基模式 B 的比较已完成，且误差未超过阈值，则认为当前比较的连续点的轨迹与基模式 B 匹配，对新到达的连续数据点重复执行上述过程；

(3) 全部基模式的匹配都未完成，但所有比较分支的误差均已超过误差阈值，此时认为该连续点的轨迹与任意基模式均不匹配或者可能是噪声数据.



(a) 基于基模式的模式匹配　　(b) 生成模式流

图 7　基于基模式的模式匹配与模式流生成

在基于基模式的匹配中，匹配成功的点的轨迹就可以用与其相似的基模式来替换，使用与其对应的唯一标识符记录；若匹配不成功，则将该连续点的轨迹直接记录下来，并记作 w_i，其中可能包含噪声数据，它不会被任何基模式匹配，因此具有一定的抗噪声能力. 如算法 4 所示生成相应的模式流，以提高后续查询的效率. 此模式流由大量的基模式标识符以及一些未匹配点的轨迹组成，将这些点的轨迹完整地记录下来，并保存在 w_i 中. 如图 7(b)所示，对应于当前数据流的模式流是 $w_1ABDDBD$，且 w_1 中同时记录了此匹配不成功的连续点的轨迹信息. 在查询不匹配时，在模式流上进行回溯的效率要高于在数以百计的数据点上进行回溯.

算法 4.　Base pattern stream generation(模式流生成).

输入：S：a data stream，EB：error threshold，
　　Dictionary(bp)：the set of all base patterns

输出：bps：base pattern stream

中间变量：$errorArray[]$：distance errors for each bp，
　　$len[]$：numbers of compared points for bp；
　　$isRecognized$：recognize as a bp or not

```
1. initialize the interior variables;
2. for every new arriving data point d_i;
3.    for j<Dictinary(bp).size;
4.       errorArray[j]+=isBelongToBP(d_i, bp_j);
                        len[j]++;
5.       if len[j]<bp_j.size and errorArray[j]>EB
           then
6.          stop comparison for bp_j;
7.       else if len[j]==bp_j.size and
                errorArray[j]<=EB
8.          add bp_j to bps;
            //maintain base pattern stream
9.          isRecognized←true;
10. if isRecognized==false then
11.    join these data points after the bps as w_i;
                  //none match, maybe contain noise data
```

在算法 4 中，将数据流的大小设定为$|S|$，计算当前点是否落在某基模式上的开销为 $O(1)$. 算法 4 的主要操作为计算累积误差并判断分支是否停止的操作，而对于多个并行比较的分支，只参考处理时间最长的即可，因此其时间复杂度为 $O(|S|)$.

5.2　基于模式流的查询实现

在模式流上，SPQ 方法对按时序顺序排列的基

模式的出现情况进行计数. 如 4.3 节中提到的可伸缩模式查询 $A[1-2]B[1](-A-D|A[1]D[1])$，设置 *counter* 数组用来记录基模式 A、B、A、D 的匹配次数. 进而根据其出现次数与 SPQ 的约束限制判断是否能够匹配上. 若不匹配时，对第一个模式的出现次数进行判断，若其在 SPQ 第一个基模式的可伸缩条件的限制范围内，且与该模式相似，则可复用部分已处理结果；否则在以基模式和 w_i 为粒度的模式流上继续后续匹配.

可伸缩模式查询的整体处理如算法 5 所示：首先调用算法 1 Base patterns generation 生成基模式和基模式覆盖，进而调用算法 3 SPQ generation 对目标模式进行基于基模式的查询重写形成 SPQ，完成可伸缩模式查询表示；随后在数据流上执行可伸缩模式查询处理，调用算法 4 Base pattern stream generation 将连续点的轨迹与所有基模式并行地比较，判断其是否与某一基模式相似，生成模式流，继而对按时序排列的基模式的出现情况进行计数，完成后续 SPQ 匹配.

算法 5. Scalable Pattern Query over Data Stream (数据流上的可伸缩模式查询).

输入：S: a data stream, EB: error threshold,
tp: target pattern, Eb_p: matching error bound
输出：MR: matching result with graphical information
中间变量：Dictionary(bp): the set of all base patterns,
Cover(bp): the base pattern cover,
SPQ: Scalable Pattern Query,
bps: base pattern stream,
counter: the appearance times,
pointer: denote the location of backtrack;
//Scalable Pattern Query Expression is focused below
1. base patterns generation(tp,Eb_p),
return Dictionary(bp) and Cover(bp);
//generate base patterns
2. SPQ generation with constraints mapping to Cover(bp);
//Scalable Pattern Query Evaluation is focused below
3. initialize the other interior variables;
4. base pattern stream generation to generate bps;
5. for every bp in bps, compare it within SPQ
6. when the type of the current bp is change, calculate the counter for the prior bp;
7. if it meets the type and counter is within the SPQ
8. locate pointer to the first bp of the latest same type in bps, update MR, continues;
9. else backtrack bps at pointer;
10. return MR;

对于模式查询而言，主要时间消耗在生成基模式及基模式覆盖上，开销为 $O(mh(m-1)(k-h))$，正如 4.2 节中提到的，此过程只执行一次，以一定的时间换取了后续匹配的精度. 在查询处理时，若共回溯 $N_{\text{backtrack}}$ 次，则时间复杂度为 $O(c|S|)$，其中 $c=1+(N_{\text{backtrack}}/|S|)$.

6 实验结果与分析

基于上述思想，本文实现了数据流上的可伸缩模式查询，设计了相关实验对比不同方法在查全率、查准率和处理效率上的区别. 具体实验环境如下：OS 是 Microsoft Windows XP Professional Edition Service Pack 3，CPU 为 Intel Core 2，内存 1GHz，主频 1.66GHz，内存 1GB. 算法开发环境是 MyEclipse 7.0 Enterprise Workbench，JDK 1.5.0；算法开发语言是 Java.

在真实的医疗数据流和用电数据集上进行实验.

(1) 医疗数据流. 从 Portland State University 的 Biomedical Signal Processing Laboratory 下载了超过 2 千万个数据点的儿科重症监护数据，是对一位儿童 6 项生理体征进行连续 6h 的同步测量得到的，频率为 500Hz. 其中包括心电图数据(ECG)、动脉血压(ABP)、中心静脉血压(CVP)、颅内压(ICP)、呼吸机数据(RESP)和动脉血氧饱和度(PLETH). 其中 ECG 的模式种类多且复杂，CVP 和 ABP 的模式种类少而且变化简单.

(2) 民用用电量数据. 数据来源于福建某电力公司，设立多个观察点，数据点超过 12 万个. 每个数据都有相应的时间戳和对应数值. 该数据集特点是包含较多噪声数据，如图 3 所示.

根据第 2 节的相关研究，SPQ 方法在查询表示上优于 DSMS 中基于元组的查询，表达灵活、语义丰富，而后者不支持对模式的查询处理. 基于时序数据库的模式匹配方法，其往往对数据进行多遍扫描，很难满足数据流的实时处理要求. 基于树的 ZStream 方法是以事件为粒度的，它是由数据源直接产生的，不支持在其内部细分，因此无法处理事件间的包含关系，在匹配中可能造成二义性. 基于图的 NFA 传统方法是离线的，近些年基于 NFA 的方法逐步向在线处理方向发展. 综上所述，本文选择一种典型的基于 NFA 的方法，即 NFA^b 的方法与 SPQ 方法进行比较. 实验主要从查全率、查准率和处理效率这 3 个指标来衡量.

6.1　查全率

查全率表示为 $R_{\text{recall}}=N_{\text{real}}/N_{\text{due}}$，用于衡量算法查询出的结果个数是否达到预期目标. 它是查询出的相关结果数与系统预期的相关结果数之比. 其中 N_{real} 为查询出的相关结果数，N_{due} 为预期的相关数目. N_{due} 采用人工标识的方式，由专业用户根据经验给定，从而形成统一的衡量标准.

图 8 展示了 SPQ 和 NFA^b 两种方法应用于医疗数据流查询时的查全率. 对应于 6 个数据集的误差阈值分别为 0.07、0.55、0.02、0.03、0.6、0.6. 从图 8 中可以看出，当模式较为复杂时，在查全率方面 SPQ 方法优于 NFA^b 方法. 由于 ECG 数据点间隔密集、模式复杂、多样性较高，导致模式间的组合多，拆分出的基模式也就更多，从而使得基于基模式操作粒度的 SPQ 方法能较准确地发现符合要求的结果；而 NFA^b 基于事件的粒度（对应于本文中不重叠模式的粒度），由于未拆分包含关系，在匹配中可能出现二义性的情况，实现时任意选择一个匹配结果以供后续处理. 所以 SPQ 方法的查全率优于 NFA^b 方法，最主要的原因是前者考虑了模式间相互包含的关系，并使用更细的基模式粒度进行查询. 类似的，RESP、ICP、PLETH 等数据集模式波形都相对较复杂，因此两种方法对比稍明显，而 CVP、ABP 模式多样性低，故两种方法对比相差不大，且其查全率相对于其它数据集要高些.

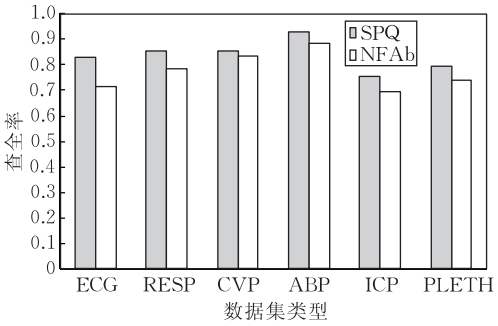


图 8　查全率对比

6.2　查准率

查准率表示为 $R_{\text{precision}}=N_{\text{real}}/N_{\text{out}}$，用于衡量查询结果的正确性. 它是查询出的相关结果数与查询出的总数之比. 其中 N_{out} 为查询出的总数，N_{real} 仍为查询出的相关结果数. 查准率 $R_{\text{precision}}$ 和查全率 R_{recall} 综合起来衡量查询结果的完备性.

图 9 展示了 SPQ 和 NFA^b 两种方法应用于医疗数据流查询的查准率. 该实验的误差阈值取值与查全率实验相同. 从图 9 中可以看出，当模式较为复杂时，SPQ 方法在查准率上优于 NFA^b 方法. 主要原因与查全率实验相似，仍是因为 SPQ 方法考虑了模式间相互包含的关系，并使用更细的基模式粒度进行查询，从而保证了查询的精准度. 但由图 9 可以看出，当模式较为复杂时，其查准率相对更高一些，比如 ECG、RESP、PLETH 等；而对于 CVP 等模式变化较少的数据集其查准率则相对低一些. 这主要是因为某一误差阈值设定后，当模式较复杂时，相互之间差别也较大，累计误差随之增加，可以较好地区分查询结果，因此查询出的结果多为吻合度较高的；而模式简单、缺乏变化的，其累计误差增加缓慢，不容易超过误差阈值，所以查询结果中会存在一些并非预期的结果，它们因未超过误差阈值被认作匹配成功，进而被包含进来. 因此对于查准率而言，SPQ 方法在模式较为复杂的数据上效果更加明显.

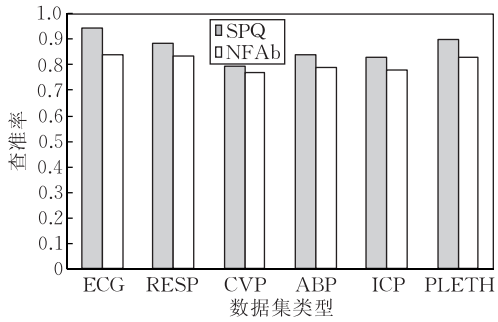


图 9　查准率对比

6.3　处理效率

处理效率表示为

$$E_{\text{process}}=\left(\sum_{i=1}^{k} response\_time_i\right)\Big/k.$$

E_{process} 用于衡量算法执行效率. 此处以返回单个查询结果的响应时间（response time）为核心，计算所有响应时间的平均值来衡量查询算法的处理效率.

如图 10 所示，SPQ 方法在处理效率上优于 NFA^b 方法，且模式越复杂优势越明显. 这主要是因为 SPQ 是基于基模式粒度的，匹配过程中，在模式流上为 SPQ 中每个基模式的连续出现次数计数，以反映其重复或缺失的情况. 当不匹配时，对第一个模式的出现次数进行判断，若其在 SPQ 第一个基模式可伸缩条件的限制范围内，且与该模式相似，则可复用部分已处理结果. 而 NFA^b 这种减少回溯的方法，当处理的模式较复杂时，每个 singleton 中可能会形成多个分支点，将其均存入栈中，当发生不匹配时逐个弹出分支点，按其位置依次匹配下一种查询的可

能性.当目标模式较复杂时,会花费较多的时间去尝试各种可能.由于 ECG 数据点间隔密集、模式复杂、多样性较高,形成的基模式较多,查询相对其它来说较复杂,因此返回单个查询结果的时间消耗也较长.

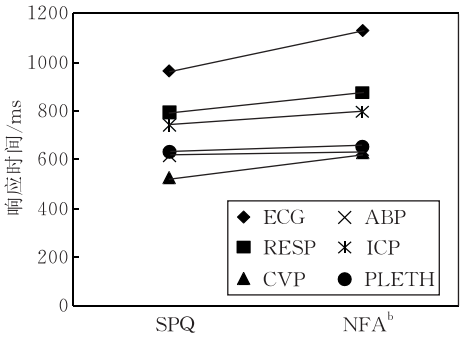


图 10 处理效率对比

6.4 误差阈值设置

图 11 显示了匹配误差阈值的设置对 SPQ 方法的影响,该实验在民用用电数据集上进行.由图 11 的结果显示可以看出,方法较好地适应了有噪声的环境.随着误差阈值的增大,查全率(图 11 菱形表示)会有较明显的增高,而到达某一值后,后续增长很平缓,这主要是因为阈值的增大在初期可以使得很多预期结果被查询出来,而随着阈值的增大,大多数的预期结果均被查询出来,所以此时再增长阈值,其查全率增长并不明显.而查准率(图 11 方块表示)则是会随着阈值的增大而逐步减小,到达某一值后减小趋势开始明显,如图中的 0.08 之后,这主要是因为随着阈值的增大,查询出的结果数增多,一些并不十分相似的也被包含进来,从而导致查准率的降低.因此为了得到全面的查询结果,SPQ 方法应从查全率和查准率两方面综合考虑来设定阈值,如民用用电数据集用例中较合理的阈值为 0.08 左右.

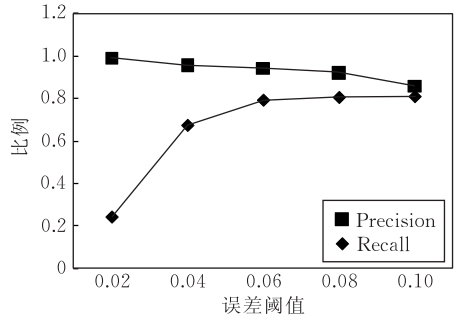


图 11 误差阈值设置

7 总结与展望

为了在数据流上支持以模式为粒度的查询,并允许目标模式具有一定的可伸缩性,本文提出了一种数据流上面向可伸缩模式的查询(SPQ)方法.SPQ 方法提出了以基模式为粒度的查询重写机制,进而在数据流上通过基模式匹配形成了模式流.在模式流上的查询处理中,尽可能复用已处理结果来提高查询效率.在真实数据集上的实验结果证明 SPQ 方法在查全率、查准率和处理效率方面优于 NFA^b 的方法,尤其当目标模式较为复杂时效果更加明显.

今后的工作主要是扩展更多约束条件以支持更丰富的语义;考虑查询结果的共享等.

致 谢 感谢为本文的成稿提供有建设性意见的各位朋友,是你们的灵感启发和无私的帮助使得作者更加完善了本文的核心思想和整体方法.同时也要感谢匿名评委,他们对本文的评审意见和建议对本文的改进大有裨益!

参 考 文 献

[1] Kahveci Tamer, Singh Ambuj K. Variable length queries for time series data//Proceedings of the 17th International Conference on Data Engineering. Heidelberg Germany, 2001: 273-282

[2] Kaghazian Leila, McLeod Dennis, Sadri Reza. Scalable complex pattern search in sequential data//Proceedings of the 17th ACM Conference on Information and Knowledge Management. Napa valley California, 2008: 1467-1468

[3] Mei Yuan, Madden Samuel. ZStream: A cost-based query processor for adaptively detecting composite events//Proceedings of the 35th SIGMOD international conference on Management of data. Providence, USA, 2009: 193-206

[4] Wu Huanmei, Salzberg Betty, Sharp Gregory C, Jiang Steve B, Shirato Hiroki, Kaeli David. Subsequence matching on structured time series data//Proceedings of the 2005 ACM SIGMOD international conference on Management of data. Baltimore, Maryland, 2005: 682-693

[5] Agrawal Jagrati, Diao Yanlei, Gyllstrom Daniel, Immerman Neil. Efficient pattern matching over event streams//Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. Vancouver Canada, 2008: 147-160

[6] Wu Eugene, Diao Yanlei, Rizvi Shariq. High-performance complex event processing over streams//Proceedings of the 2006 ACM SIGMOD International Conference on Management of data. Chicago, USA, 2006: 407-418

[7] Brenna Lars, Gehrke Johannes, Hong Mingsheng, Johansen Dag. Distributed event stream processing with non-deterministic automata//Proceedings of the 3rd ACM International

Conference on Distributed Event-Based Systems. Nashville Tennessee, 2009: 1-12

[8] Badu Shivnath, Widom Jennifer. Continuous queries over data streams. ACM SIGMOD Record, 2001, 30(3): 109-120

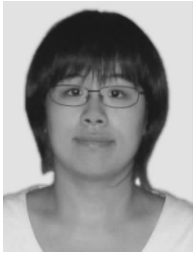
[9] Chandrasekaran Sirish, Cooper Owen, Deshpande Amol, Franklin M J, Hellerstein J M, Hong Wei, Krishnamurthy S, Madden S R, Reiss F, Shah M A. TelegraphCQ: Continuous dataflow processing for an uncertain world//Proceedings of the 2003 ACM SIGMOD International Conference on Management of data. San Diego California, 2003: 668-668

[10] Abadi D J, Carney Don, Cetintemel U, Cherniack M, Convey C, Lee Sangdon, Stonebraker M, Tatbul N, Zdonik Stan. Aurora: A new model and architecture for data stream management. The International Journal on Very Large Data Bases, 2003, 12(2): 120-139

[11] Yao Yong, Gehrke Johannes. The cougar approach to in-network query processing in sensor networks. ACM SIGMOD Record, 2002, 31(3): 9-18

[12] Cortes Corinna, Fisher Kathleen, Pregibon Daryl, Rogers Anne. Hancock: A language for extracting signatures from data streams//Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Boston, USA, 2000: 9-17

[13] Papadimitriou Spiros, Sun Jimeng, Faloutsos Christos. Streaming pattern discovery in multiple time-series//Proceedings of the 31th International Conference on Very Large Data Bases. Trondheim Norway, 2005: 697-708



LI Fei-Fei, born in 1986, M. S. candidate. Her research interests include data stream management and mining.

LI Hong-Yan, born in 1970, professor. Her research interests include data management and data mining.

QU Qiang, born in 1985, M. S. candidate. His main research interests include query processing in database community and large-scale graph mining.

MIAO Gao-Shan, born in 1985, M. S. candidate. His research interests include data stream management and mining.

Background

Nowadays query process over evolving data streams is a rich and rapidly growing research domain. It has received considerable attentions from the industrial and academic community, such as biomedical-based monitoring, especially in the environment of Intensive Care Unit (ICU).

Since patterns usually indicate abundant semantics, pattern query over data streams possesses high domain significance. In many cases, the query target pattern is scalable, which means it is comprised of sub-patterns, with some sub-patterns gained, lost or even inversed. Meanwhile, querying over data streams also requires anti-noise capability and the evaluation should be real-time. However, traditional query methods on Data Stream Management Systems (DSMS) are based on the granularity of tuples, not semantic patterns. Furthermore, some methods for pattern matching, especially the ones used in complex event process systems (CEP), are not applicable, particularly when the query target pattern is scalable.

This paper presents an scalable pattern query (SPQ) method. It allows users to define target pattern and set corresponding constraints according to their knowledge and concerns. And then the target pattern can be changed to scalable pattern. In the stage of query evaluation, pattern stream is generated by pattern matching, and then scalable pattern query is carried out on the pattern stream. From the perspective of recall ratio, precision ratio and processing efficiency, the experimental results on real datasets reveals that SPQ is feasible and flexible.

This research is supported by the National Natural Science Foundation of China (NSFC) under grant No. 60673113 and No. 60973002. Those researches aim to improve the data stream management and some related issues, developing basic issues and the key technologies. This paper is part work of them, and it mainly focused on the query process over data streams.

The research team has published papers in top-level conferences of database. One paper named "Effective Variation Management in Pseudo Periodical Stream" has been published by ACM SIGMOD 2007. It mainly focused on the management of abnormal pattern over pseudo data streams. The work in this paper was on the basis of the previous work, and placed more emphasis on the scalable pattern query. The authors enabled some sub-patterns of the target pattern can gain, lost or even inverse, which provided the users with a feasible and flexible mechanism to query more complex patterns over evolving data streams. As the typical data stream scenario illustrated, it can be an assistant method in ICU, which can provide the doctors with some additional therapy information.