第36卷 第22期 计 算 机 工 程 2010年11月
Vol.36 No.22 Computer Engineering November 2010
·软件技术与数据库· 文章编号：1000—3428(2010)22—0043—03 文献标识码：A 中图分类号：TP301.6

基于流立方体的数据流频繁模式挖掘算法

袁正午，程宇翔，梁均军，李 林
(重庆邮电大学计算机科学与技术学院，重庆 400065)

摘 要：针对关系型数据流，提出一种基于流立方体框架的频繁模式挖掘算法。通过数据流的不断到达动态地创建流立方体来保存近期数据流信息，当用户提出查询请求时在以创建的流立方体基础上进行频繁模式的挖掘计算，返回相应的查询结果，可以快速地挖掘数据流各维之间存在的所有频繁模式。通过分析和实验表明该算法有较好的性能。

关键词：频繁模式；流数据；流立方体；关系型数据流

Mining Algorithm for Frequent Pattern in Data Stream Based on Stream-cube

YUAN Zheng-wu, CHENG Yu-xiang, LIANG Jun-jun, LI Lin
(College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China)

【Abstract】 This paper proposes a new method based on the stream-cube architecture, which is used to deal with relational data stream model. In this algorithm, a dynamical process creates stream-cube to save recent data stream, which continuously arrives with time sequence. Based on the architecture with an analyst's query, this algorithm can find all of frequent patterns in data stream fast among all dimensions. At last, the analysis and experiments show that this method has good performance.

【Key words】 frequent pattern; data stream; stream-cube; relational data stream

1 概述

近年来，数据流已成为国内外数据库领域热点，具有无限性和流动性等特点。针对数据流的算法必须考虑以下条件[1]：(1)所有数据元素最多只能访问一次；(2)新产生的数据必须能够尽快地处理；(3)有限的存储空间；(4)算法必须具有快速响应的能力。而传统的频繁模式挖掘算法难以满足以上条件，因此，基于数据流的频繁模式挖掘技术面临十分艰巨的挑战。

挖掘数据流上的频繁模式能够为数据流应用提供重要的决策依据。考虑到数据流的特点，这些频繁模式会随着数据流的不断产生而发生变化。而在许多数据流的应用中，用户往往更加关注数据流上最近数据所包含的模式信息，而并非整个数据流上的模式信息。因此，挖掘数据流中最近时间的频繁模式便成为研究的重点。在过去的几年中，数据流上的频繁模式挖掘工作取得了许多研究成果。文献[1]在 FP-stream 算法基础上提出了另一种更加适合于长模式的数据流频繁模式挖掘算法 FP-DS。文献[2]根据 Moment 算法提出一种基于频繁闭项集挖掘的增量式维护算法，通过滑动窗口增量更新数据流中的事务，采取一种高效的项的位序列表示方法降低窗口滑动的时间和空间复杂度，应用压缩的模式树进行频繁闭项集检查，确保挖掘结果的准确性。文献[3]提出一种挖掘频繁闭合项集的算法。该算法将数据流分段，用 DSFCI_ tree 动态存储潜在频繁闭合项集，对每一批到来的数据流，建立局部 DSFCI_tree，进而对全局 DSFCI_tree 进行更新并剪枝，从而有效地挖掘整个数据流中的频繁闭合模式。

上述各种算法针对的数据流类型都是序列数据流模型，数据流中的每个元素之间不存在任何关系，两两之间相互独立。而在很多数据流的应用中，比如金融分析、网络监控、电信数据管理、Web 应用、传感器网络等方面，每个数据项都是关系型元组，通常被称为关系数据流模型。本文正是将关系数据流作为研究对象提出一种基于流立方体[4]挖掘频繁模式的方法。该方法根据数据流中不断产生的数据项，动态建立流立方体结构，并借助流立方体结构进行挖掘。通过实验分析和对比，证明本文提出的方法在处理大量数据时具有很好的性能，适合于挖掘关系型数据流上的频繁模式。

2 流立方体的相关概念

2.1 流立方体体系结构

文献[4]提出了流立方体的概念和体系结构，其目的是用于对大量、无限、动态的数据流进行有效的在线分析处理。数据流中所包含的数据存在多维多层次的特点，在设计流立方体时为了满足这些特点，在其体系结构中包含 3 个核心概念：(1)倾斜时间框架；(2)关键层；(3)通用路径。

倾斜时间框架将时间分割成不同的粒度等级，越近的时间拥有越细分的粒度，而过去越久的时间则会被聚集到一个比较粗的粒度等级中。这种划分也符合了现实需求，因为对于数据流来说，人们更关心当前的状态，并且随着时间的流逝，数据的影响力也会逐渐变弱。利用这种时间分割更能有

基金项目：国家“863”计划基金资助项目“国家项目综合 EM 方法与混合模型的空间统计挖掘技术研究”(2007AA12Z226)
作者简介：袁正午(1968－)，男，教授，主研方向：空间定位技术，空间信息集成；程宇翔、梁均军、李 林，硕士研究生
收稿日期：2010-06-24 **E-mail：**yuanzw@cqupt.edu.cn

效地在有限的存储设备中保持更多的数据流信息。

数据流特有的性质决定原始数据流中所包含信息多属于低粒度级，而这些信息并非是应用真正需要的；因此，在流数据立方体中包含 2 个关键层：最小兴趣层和观察层。在最小兴趣层，把原始数据通过概念分层原理聚集到一个感兴趣的最低层次；观测层包含分析人员感兴趣的最高层次，应用概念分层的原理得到最小兴趣层数据的聚集。流立方体为了存储空间和计算时间进行折中，采用了部分物化的方式计算和存储立方体，通用路径用来说明需要物化立方体的哪些单元格。

2.2 基于 H-tree 实现的流立方体

流立方体的实现是基于一种 H-tree[5]的数据结构。它是一种基于内存的层次树结构，能保证在有限内存中有效地进行多维多层次的聚集计算，并且在创建 H-tree 时，需要对数据集扫描一次。这种特点也决定了 H-tree 能很好地适应对数据流的处理。在流立方体中，H-tree 为适应流立方体的体系结构与文献[5]有所不同。

定义 1(H-tree 在流立方体中的结构) H-tree 是由两部分组成的，一部分是 H-tree 树结构，另一部分是 Header 表结构。在树结构中，每个结点的结构为(*Parent, Label, Children, Time-tilt*)。其中，*Parent* 表示结点的父节点；*Label* 为该结点的标识；*Children* 中保存该结点的孩子结点；*Time-tilt* 保存时间倾斜中不同粒度的数据。在 Header 表中，每个表项的结构为(*Label, Sidelink*)。其中，*Label* 代表该项所对应的结点的标识；*Sidelink* 按时间顺序保存 *Label* 所标识的结点。

在流立方体中，充分利用了倾斜时间窗口的概念，保存不同时间粒度下的模式信息，从而使得对数据流的处理更具动态性和时间效率。H-tree 通过其特有的结构，可以很好地在有限的内存中保存和处理大量的数据流信息，一次扫描就能够充分保证数据的完整性，在很大程度上提高了对数据流进行的挖掘结果的正确性，使得结果更加可靠。

3 基于流立方体的数据流频繁模式挖掘方法

3.1 关系数据流的定义

通过对流立方体概念和结构的分析，证实基于流立方体的数据流频繁模式挖掘与过去的挖掘方法有所不同。过去数据流频繁模式的挖掘针对的是任意长度的序列型数据；而在流立方体中每条记录拥有相同的维信息，属于关系型数据。

对于任意长度的频繁模式算法一般都基于以下定义：设 $I=(i_1,i_2,\cdots,i_n)$是一个数据项集，D 是一个事务集，每条事务 T 对应于一个数据项子集，即 T_I。对于数据项集 X，当且仅当 X_I，称事务 T 包含 X。项目集中项目的个数称为项目集的维数或长度，若项集长度为 k，称为 k-项目集。项集 X 在 D 上的支持度(*support*)是包含 X 的事务在 D 中所占的百分比。若 *support*(X)不小于用户给定的最小支持度，则称项集 X 为频繁项集。而本文中所研究的数据流有别于以上假设，而基于以下定义。

定义 2(关系数据流) 设 $R=\{r_1,r_2,\cdots,r_n\cdots\}$是由一个数据流组成的无限数据表，$r_i$ 表示数据表中的一条记录，每条记录 r_i 满足 S(T,D_1,D_2,$\cdots$,D_n,M)属性域。其中，T 表示时间戳属性；$D_1,D_2,\cdots,D_n$ 是 n 个不同的关系型数据维，即 n 个不同的属性列；M 是一个度量值。

定义 3(数据流频繁模式) 设 $D_i=\{di_1,di_2,\cdots,di_m\}$是一个有限集合$||D_i||=m$，在 R 中存在对应于 $D_1,D_2,\cdots,D_n$ 的属性组合 $P(D'_1,D'_2,\cdots,D'_n)$，$||Set||=n$ 且 $D'_i\{di_1,di_2,\cdots,di_m\}\{*\}$，*代表取改属性的任意值。将 *Set* 在 R 中出现的次数记为 *Count*(*Set*), 用户提出的最小支持度为 *Minsupport*，如果 *Count*(P)大于 *Minsupport*，则称 R 中存在频繁模式 P。

定义 4(数据流频繁模式集) 在数据流 R 中，包含多个不同的频繁模式 $P_1,P_2\cdots$，这些频繁模式的集合 $Set=\{P_1,P_2,\cdots\}$，称为数据流频繁模式集。

3.2 数据流频繁模式挖掘

本节提出一种基于流立方体的数据流频繁模式挖掘方法。该方法通过流立方体来保存近期数据流信息，在一定时间间隔(由最大时间倾斜时间决定)或者在用户查询请求到达时进行频繁模式的挖掘计算，返回相应的查询结果。整个算法实现包含两部分，具体算法描述如下：

算法 1 流立方体频繁模式算法

输入 流立方体(*SC*)，查询请求(*Q*)

输出 数据流频繁模式集(*FPSet*)

(1)解析查询请求 *Q*，得到查询的时间范围 *T* 和最小支持度 *Minisupport*；

(2)根据 *SC* 获取 header 表 *HTable* 以及数据维数 *DCount*；

(3)设频繁模式的结果为 *Rusult*={*,*,…,*,*}，||*Result*||=*DCount*；

(4)将频繁模式集 *FPSet* 初始化设置为空；

(5)调用算法 *ComputeFP*(*DCount, HTable, Result, FPSet, T, Minisupport*)；

(6)返回频繁模式集 *FPSet*。

算法 2 频繁模式计算算法

输入 数据维数(*DCount*)，Header 表(*HTable*)，频繁模式结果(*Result*)，频繁模式集(*FPSet*)，查询时间范围(*T*)，最小支持度(*Minisupport*)

输出 数据流频繁模式集(*FPSet*)

(1)遍历所有 *DCount* 个维，并设当前计算的属性值为 *d*，对属于该维的所有数据进行遍历，根据 *T* 计算在此时间段内 *d* 出现的总次数 *Sum*；

(2)如果 *Sum*>*Minisupport*，*Results*[*i*]=*d*，并将结果添加到 *Set* 中，如果没有遍历结束所有维，那么计算当前数据的父节点 Header 表 *PHTable*，调用函数 *ComputeFP*(*i*-1, *PHTable, Result, Set, T, Minisupport*)；

(3)如果 *Sum*<*Minisupport* 但是没有遍历结束所有维，则 *Result*[*i*] = *，并调用函数 *ComputeFP*(*i*-1, *PHTable, Result, Set, T, Minisupport*)。

4 实验与性能分析

本文采用的数据流是以 IBM 模拟数据生成器产生的数据为基础得到的，实验在 CPU 为 Intel Pentium 4 2.60 GHz，内存大小为 2.0 GB，操作系统为 Windows XP SP3 的 PC 上进行。所有算法使用 Sun Micorsystems 发布的 Java 1.5.16 实现。通过性能测试，并与经典算法 FP-stream 比较本文提出算法具有良好的稳定性和挖掘速度。

4.1 算法性能分析

实验采用 3 个不同的数据集分别对挖掘运行时间和内存占用 2 个方面进行测试，3 个数据集的元组长度分别为 5、6 和 7。元组数据量从 200 KB 到 2 MB 递增，每次的增量为 200 KB，在对比不同长度的元组时所使用的支持度为 0.005。

从图 1 和图 2 中可以看出，随着元组长度的增加，内存占用量和运行时间都逐渐增加。但是在长度一定时，内存占用量和运行时间都逐渐趋于稳定，并不会随着数据量的增加

而不断增加，也就是说与数据集的大小无关。因此，算法能够很好地应用于数据流的计算中。

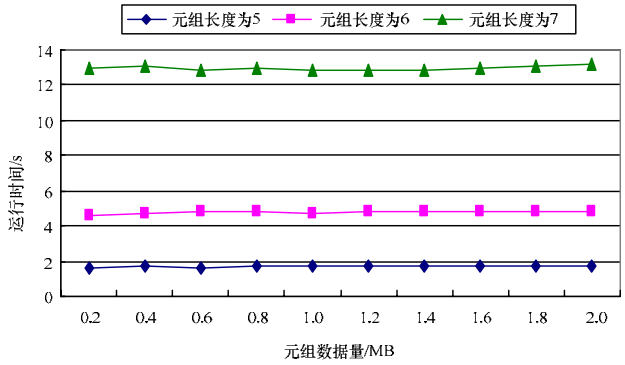

图 1　数据处理运行时间

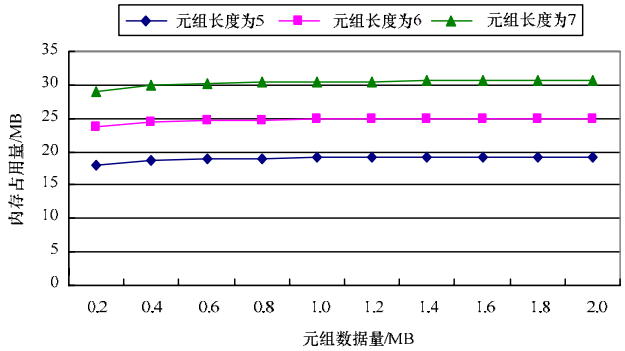

图 2　数据处理占用内存量

图 3 中使用长度为 7 的数据作为测试数据。可以看出随着支持度的降低，运行时间也逐渐平稳地增加，没有出现比较明显的波动和陡增的情况，这也充分体现出了本文提出的算法具有较好的稳定性。

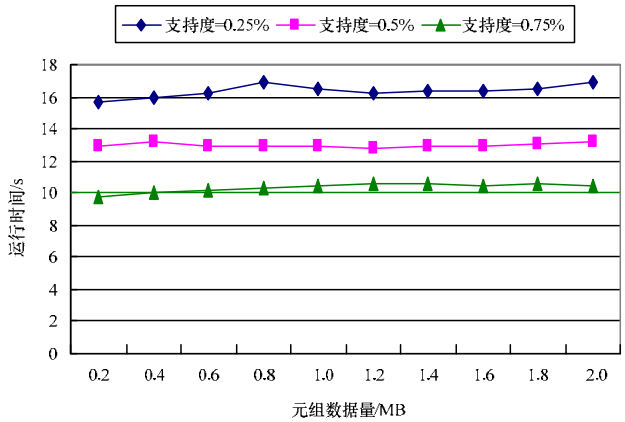

图 3　不同支持度下的运行时间

4.2　实验比对分析

FP-stream 算法是一种典型的数据流频繁模式挖掘方法。采用数据长度为 7、支持度为 0.005 的数据将其与本文中提出的算法进行运行时间和内存占用量两方面的对比分析，如图 4 和图 5 所示。

根据图 4 可以得到，在元组数小于 0.8 MB 时，FP-stream 的运行时间要优于本文中提出的算法。而由于本文提出算法的稳定性要优于 FP-stream，运行时间并不随着元组数的大小而发生变化，因此在处理大于 0.8 MB 的数据时性能逐渐优于 FP-stream 算法。根据图 5 可以看出，2 个算法在内存占用量上都比较平稳，没有出现较大波动。由于本文算法是基于流立方体结构，其中 H-tree 和 Header 表结构占据了一定的存空间，因此整体内存占用较高，不过仍处于可接受范围内。

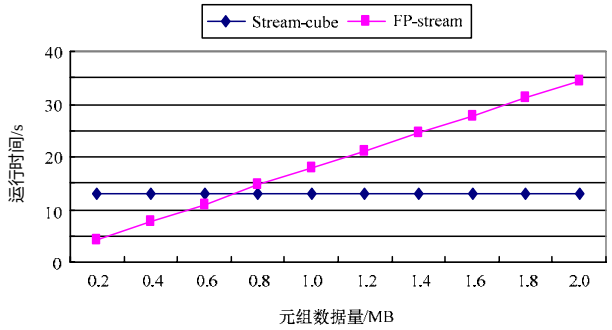

图 4　运行时间的比较

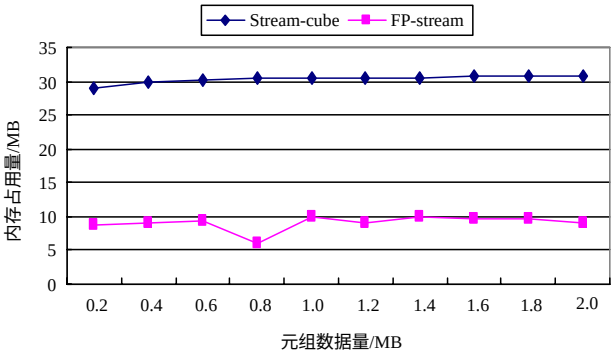

图 5　内存占用的比较

5　结束语

本文基于流立方体，针对关系型数据流频繁模式挖掘方法，提出一种基于流立方体框架的频繁模式挖掘算法。通过性能分析，可以看出，本文提出的算法具有良好的稳定性，在与经典的 FP-stream 方法的对比中，在大数据量的计算上具有明显的优势。由于使用了流立方体的结构，使得在内存占用量上比 FP-stream 明显增加，但是也处于可接受范围内。并且通过实际验证，该算法可以准确地获得与时间倾斜粒度对应的所有频繁模式。

参考文献

[1] 刘学军, 徐宏炳, 董逸生. 挖掘数据流中的频繁模式[J]. 计算机研究与发展, 2005, 42(12): 2192-2198.

[2] 李　俊, 杨天奇. 基于滑动窗口的数据流频繁闭项集挖掘[J]. 计算机工程, 2009, 35(13): 37-39.

[3] 程转流, 胡学刚. 数据流中频繁闭合模式的挖掘[J]. 计算机工程, 2008, 34(16): 50-52.

[4] Han Jiawei, Chen Yixin, Dong Guozhu, et al. Stream Cube: An Architecture for Multi-dimensional Analysis of Data Streams[J]. Distributed and Parallel Databases, 2005, 18(2): 173-197.

[5] Han Jiawei, Pei Jian, Dong Guozhu, et al. Efficient Computation of Iceberg Cubes with Complex Measures[C]//Proc. of ACM-SIGMOD International Conference on Management of Data. Santa Barbara, USA: [s. n.], 2001.

编辑　顾逸斐