基于汉字字段的关系数据库数字水印研究

王 堂[1,2,3] 曹宝香[1] 芦效峰[2,3] 杨义先[2,3] 钮心忻[2,3]
(曲阜师范大学计算机科学学院 曲阜 276826)[1]
(北京邮电大学网络与交换技术国家重点实验室信息安全中心 北京 100876)[2]
(北京邮电大学网络与信息攻防技术教育部重点实验室 北京 100876)[3]

摘 要 提出了一种实用的基于汉字字段的关系数据库数字水印新方案。通过主键、用户密钥集合和水印嵌入间距确定目标属性值。根据定义的规则,计算比较属性值和水印的特征值并将不同位作为水印嵌入位。通过语义分析给出属性域中某个最相关的非上下结构的汉字,嵌入水印的过程就是编辑(插入/删除)该汉字。嵌入的水印具有不可见性,并且不影响数据库的可用性,可实现盲提取。该方案对插入、删除、修改数据库记录以及删除数据库字段等常见数据库更新具有较好的鲁棒性。

关键词 关系数据库,数字水印,汉字字型结构,语义分析,多数投票
中图法分类号 TP309.2 **文献标识码** A

Research on Watermarking Relational Database Based on Character Field

WANG Tang[1,2,3] CAO Bao-xiang[1] LU Xiao-feng[2,3] YANG Yi-xian[2,3] NIU Xin-xin[2,3]
(College of Computer Science, Qufu Normal University, Qufu 276826, China)[1]
(Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China)[2]
(Key Laboratory of Network and Information Attack & Defense Technology of MOE, Beijing University of Posts and Telecommunications, Beijing 100876, China)[3]

Abstract An innovative practical watermarking scheme for relational database based on character field was proposed in this paper. Identifying the target attribute values through the primary key, set of user keys and the span of watermark. Calculating the characteristics of attribute values and the watermark according to the defined rules and choosing those different bits as watermark bits. Semantic analysis gives a character, which is non-down structure and one of the most relevant attributes in the domain, and watermark embedding is the process of editing(insert/delete) the character. The embedded watermark is invisible and does not affect the availability of the database, and enables the blind extraction. The proposed scheme can achieve good excellent robustness to common database update such as insert, delete, modify and delete database fields.

Keywords Relational database, Digital watermarking, Structure of Chinese characters, Semantic analysis, Majority vote

1 引言

随着关系型数据库的广泛应用,数据库中存储的数据量急剧增大。对于那些提供信息服务(如气象信息、医疗信息、人才市场信息、股票交易信息等)的公司,其主要资产便是存储于数据库里的大量数据。如果不法分子从数据库里获取大量数据转卖给他人,这些信息服务公司必然会蒙受巨大的经济损失。在这些大量数据的背后也隐藏着许多重要信息,利用数据挖掘技术可挖掘出有用的商业信息。这些数据面临着被窃取的危险,数据库的版权保护迫在眉睫[1]。

数据库水印是指在不破坏数据库可用性并满足数据库的语义约束关系和结构约束关系的前提下,用信号处理的方法在数据库中嵌入不易察觉且难以去除的标记,以达到保护数据库版权的目的。数据库水印技术并不限制正常的数据存取,只是提供对数据库的所有权证明与内容完整性验证,进而对数据窃取与攻击行为起到电子举证的作用,从而有效地进行版权保护。

目前关系数据库数字水印算法研究基本上都集中于对数值型属性的研究,针对非数值属性的水印算法研究比较少,其中针对汉字字段的研究更少。然而,随着数据库技术在各个

到稿日期:2011-01-20 返修日期:2011-04-15 本文受国家973项目(2007CB311203),国家自然科学基金(60803157,90812001),国家242项目(2009A105),国家标准制定计划(20080200-T-339),国家质检公益性科研专项(10-126),山东省自然科学基金项目(ZR2009GM009)资助。
王 堂(1985-),女,硕士生,主要研究方向为多媒体数字水印等,E-mail:wangtang_2008@163.com;曹宝香(1955-),男,教授,硕士生导师,主要研究方向为企业信息化与系统集成;杨义先(1961-),男,博士,教授,主要研究方向为密码学、计算机网络与信息安全;钮心忻(1963-),女,博士,教授,主要研究方向为数字水印、信息隐藏、隐写分析。

行业的深入应用，汉字字段在数据库中的应用越发普及，譬如企业销售数据库、医药数据库、人才数据库等。因此，如何有效地利用汉字字段使数据库数字水印技术更具实用性，是目前该领域的一个前沿研究热点。

2 问题分析

2.1 关系数据库中数字水印的特点

关系数据库中数字水印和其他（如图像、音频、视频和文本等）数字媒体中数字水印一样，有如下基本特征[2]。

（1）不可见性：不因加了水印而降低关系数据库数据的使用价值，数据库的使用者应该感觉不到水印的存在。

（2）鲁棒性：不因用户对数据库数据的某种改动而导致隐藏的水印信息丢失的能力。这里的改动是指对关系数据库正常的数据更新（如增加、删除元组或者修改元组的属性值等）和各种恶意攻击。如果攻击者修改的关系数据的元组数量足以破坏该关系数据的水印，那么该关系数据也会因数据修改太多而失去本身的使用价值[3]。

（3）检测能力：从可疑的数据库中检查到一定数量的元组含有所做标记的能力，同时降低误判率。这里的误判是指没有加水印的关系数据库却检测到水印，而加了水印的关系数据库却没有检测到水印的存在。

2.2 现有数据库水印设计方案

（1）基于嵌入定位型：R. Agrawal 等[3]所提的方法选择主键作为定位标志嵌入数据库数据，但它无法对无主键数据进行水印嵌入，且其主键也因此易受攻击。R. Sion 等[4]和 Y. Li 等[5]则以数据 MSB(Most Significant Bit)值来构造虚拟主键定位，其缺点在于较难寻找能唯一区分各元组的数值属性，且这样的属性值极有可能在正常使用中发生变化，从而导致提取水印失败；在这种类型的数据库水印算法中，不论所选的是真正的主键还是虚拟的主键，如果只用二者作为定位依据，则在分组嵌入水印时会使嵌入后的载体数据改变过大。

（2）基于嵌入内容型：R. Sion 等[4]和 Y. Li 等[5]在数据库中嵌入带有实际意义的信息；而 R. Agrawal 等[3]的方法则是嵌入数值型属性的 LSB(Least Significant Bit)位，方法是按一定规则将特定位的比特置为 1 或 0。这种算法基于数据的概率特性，虽能验证数据是否包含隐藏的水印信息，但嵌入的水印信息编码没有特定含义，仅与数值本身有关，很难通过其内容信息证明版权的所有者。牛夏牧等在文献[3]的算法上做了一些改进[1]，嵌入了有实际意义的信息，但这种嵌入会对载体数据有较大的影响。而张勇等提出一种新的关系数据库云水印技术，它把图像信息转换为水印云滴后嵌入到数据库数据中。其缺点也是对载体数据的影响较大，且它的抽取结果只能是一个相似度值而无法直接提取版权信息。

（3）基于嵌入方法型：R. Sion 等针对数值型属性嵌入水印的方法[4]，在可用性规则限定下修改数据的分布特性，成组嵌入水印。它对数据的改变虽然在误差允许范围内，但由于改变了数据的分布特性，因此隐蔽性不够好，而且嵌入计算量很大；文献[6]由于在提取水印时需要使用嵌入图（Embedding map），因此它不是一种“全盲”数据库水印算法。文献[7]利用水印将元组中的某个属性用其他值替换，可嵌入任意类型属性，虽没有改变数据的分布特性，但属性值的变化会影响数据正常使用。Y. Li 等通过改变数据各“索引对”间的顺序嵌入水印，不改变数据的物理位置及关系数据值，几乎不影响数据的使用。但索引是关系数据之外的附加信息，若对关系表做重新索引或删除索引，则水印信息完全丢失，鲁棒性不够好；文献[8]利用图像水印技术对数据嵌入水印，它只能用于包含图片属性的特殊关系数据，若将图片属性删去，则水印无法保留。David 提出在保持查询结果不变的前提下，成对改变数值而保持总和不变来嵌入水印，从总体结果看不影响数据的使用，但数据改动可能涉及用户级应用程序，不符合数据应用的需求。

2.3 字段分析

非数值型数据拥有与数值型数据完全不同的特性，在设计针对非数值型数据的水印算法时无法照搬数值型数据的水印算法思路，必须寻找新的方法向非数值数据中嵌入水印信息。非数值型数据的二进制位串表示的是信息编码，改变其任意一位，将直接导致编码的转移，甚至导致数据表达含义的完全改变。因此，找到一种通用的针对非数值型字段的水印算法，是不太可能的，这里只考虑非数值型汉字字段的数字水印方案。

2.4 汉字结构分析

考虑到汉字字段中属性域是汉字的集合，编辑属性值时必须尊重汉字自己的特点。每个汉字都可以看作由部首、偏旁和字按照一定的结构组合而成[9]。汉字集可以分成如下 4 种结构[10]。

1）独体结构型。由单式、连式、交式字根组成的单字，结构紧密，独自成为一体，这样的构型称作独体型。

（1）单式独体型。例如，三、石、鱼、米、山；

（2）连式独体型。例如，天、下、千、少、尺；

（3）交式独体型。例如，夫、丈、事、秉、半、坐。

2）左右结构型。单字内分成左根和右根两半，中间有一定间隙的散式构型，称作左右型。例如，桂、相、够、错、海。

3）上下结构型。单字内分成上根和下根两半，中间有一定间隙的散式构型，称作上下型。例如，苹、学、杂、岩、昌。

4）包围结构型（又称内外结构型）。单字内一个内根被一个外根全部或局部包围的散式构型，称作包围型。

（1）全包围的单字。例如，困、囚、圆、园；

（2）3 个方向包围的单字。例如，区、冈、凶、网、同；

（3）两个方向半包围的单字。例如，这、历、司、延。

2.5 常见数据库攻击

关系数据库数字水印应具有一定的鲁棒性，应该可以防御各种各样的攻击，包括正常的数据更新和恶意的攻击。因为关系数据库数据需要经常维护更新，所以包含在一个关系数据库中的标记不能因为正常的数据更新而在无意中被去除掉，否则其效果就如同恶意攻击一样。即如果数据窃取者不知道关系数据加了水印，那么他在对偷来的关系数据进行正常更新时不能因数据的更新而丢失嵌入在关系数据库里的水印信息。如果数据窃取者知道偷来的关系数据加了水印，他就会试图擦掉水印或者用其他方法声明对关系数据库的所有权[3]。

水印系统应该能保护数据的原始所有权而阻止数据窃取者各种形式的恶意攻击。对水印关系数据库常见的恶意攻击方式有[3,11]：

（1）子集攻击：数据窃取者并不使用水印关系数据库的全

部属性和元组，而仅仅使用其部分属性或元组子集，从而达到既可以使用数据库而又能破坏掉原有水印的目的。

(2)混合和匹配攻击：数据窃取者首先收集多个包含有相似信息的数据库，然后从每个数据库中提取出部分彼此不相关的元组，将它们混合后创建出自己的数据库。

(3)添加攻击：数据窃取者在窃取来的已经加入水印的数据库元组上再简单地加上属于他自己的水印信息，然后声明自己对该数据库的所有权。这种攻击简单易行而且可以有效混淆数据库的真正所有权，从而达到使水印失效的目的。

(4)可逆性攻击：如果数据窃取者在其偷来的数据库内发现了一个虚幻的水印，他就可以采取可逆性攻击，声称自己对关系数据库的所有权。而实际上数据窃取者声称的水印只是偶然出现在数据库中的一些特殊信息，他伪称这些信息是自己的水印信息。

(5)位变相攻击：对数据库中某些属性值中部分位进行简单的值修改或替换，从而试图实现对数据库水印的去除。

(6)零攻击：将数据库中某些属性位上的值统一设置成为零值，从而破坏原水印信息对属性位上值的修改。

(7)循环攻击：数据窃取者通过给属性值进行循环赋值来尝试去除隐藏在属性值中的水印信息。

2.6 关系数据库数字水印特点分析

给关系数据库嵌入数字水印，应考虑以下几点：(1)关系数据库由许多独立的元组组成，几乎没有可以利用的信息冗余。在不影响数据库使用的前提下，如何确定水印嵌入位并且在水印检测提取时能够正确定位，应该找一个稳定的参考点。数据库中相对最稳定的应该是主键值，但是攻击者也会考虑到这一点。这时考虑到加密，应用只有版权所有者知道的密钥加密。本文正是在主键值不变的前提下采用用户提供的密钥、水印嵌入间距等信息展开工作的。(2)数据库中元组之间以及元组的属性值集合之间是无序的，只是在数据属性之间存在某种依赖关系，嵌入的水印如何能抵抗元组乱序攻击，元组的编号肯定不能用于嵌入位的确定。本算法用到了消息认证码，该认证码只与主键值和用户密钥有关。(3)对元组进行小部分去除或用外部元组来代替原数据库元组很难被发觉，也就是如何防御子集攻击。本文采用了水印循环嵌入的方式。攻击者盗用数据库数据的最终目的是使用数据库，水印循环嵌入在版权保护方面很有效。(4)关系数据库内部的数据更新十分频繁，如何动态嵌入水印以及保障嵌入水印的鲁棒性，应该添加数据库嗅探器，实时统计数据库的更新。当更新数据量超过一定的阈值时，启动重新嵌入水印过程。

3 相关术语和约定

定义 1 待嵌入水印信息的关系数据库表：$R(P,A_1,A_2,\cdots,A_m)$。其中，P 是关系表 R 的主键，$t.A_i$ 表示关系表 R 中元组 t 的第 i 个非数值属性列，其中 $1\leqslant i\leqslant m$；且其取值有 $|A_i|$ 种可能：$\{a_{i1},a_{i2},\cdots,a_{i|A_i|}\}$(即属性 A_i 的属性域)，在算法描述中一般用 A_i 表示其属性域，而用 $|A_i|$ 表示其属性域的长度。$|R|=N$ 表示关系表 R 中有 N 个元组。

定义 2(水印信息 wm) 水印信息 wm 表示要嵌入到关系表 R 中的水印信息。$wm.length$ 表示预处理后水印信息 wm 的编码长度，即其二进制表示形式含有 $wm.length$ 个二进制位。

定义 3 单向哈希函数 H。

$h=H(M)$是一个满足如下要求的单向哈希函数：

(1)给定 M，h 易于计算。

(2)给定 h，M 难于计算。

(3)给定 $H(M)$，难以寻找一个 M'，使得 $H(M')=H(M)$。

定义 4 消息认证码 F[12]。

$F(t.P,K)=H(K||H(t.P||K))$

式中，K 是惟有数据库拥有者可知的私有密钥，$||$表示连接操作。$F(t.P,K)$称为元组的消息认证码(MAC)。

定义 5 水印嵌入间距 e。

水印嵌入的间距用 e 表示，即平均每 e 个元组中嵌入一位水印信息。

4 特征值计算

汉字字段属性域可以看成是 4 种汉字结构排列组成的集合。假定某个字段的属性值由 n 个词组组成，第 $i(1\leqslant i\leqslant n)$ 个词组用 C_i 表示，H_{ij} 表示第 i 个词组中第 j 个汉字。用 B 表示各个变量的二进制值[10]。

设定汉字结构与二进制值的对应规则 1：

(1)若 H_{ij} 的结构为独体结构，则 $B(H_{ij})=001$；

(2)若 H_{ij} 的结构为左右结构，则 $B(H_{ij})=010$；

(3)若 H_{ij} 的结构为上下结构，则 $B(H_{ij})=011$；

(4)若 H_{ij} 的结构为包围结构，则 $B(H_{ij})=100$。

$B(C_i)=B(H_{i1})||B(H_{i2})||\cdots||B(H_{ij})\cdots$。其中，$||$表示连接操作；每个词组由一系列的 0，1 组成的二进制代码组成。

设定词组特征值计算规则 2 如下：

用规则 1 表示属性域中的汉字，用 t 表示属性 i 的代码序列特征值，则当 $B(C_i)$中二进制位为“1”的个数为奇数时，$t(i)=1$；当 $B(C_i)$中二进制位为“1”的个数为偶数时，$t(i)=0$。

5 水印方案

5.1 方案描述

本文将对每个汉字字段单独进行水印嵌入，并且用于每个字段的密钥是不一样的，这满足了水印方案的安全性需求。水印方案由 3 个核心部分构成：预处理、水印嵌入、水印检测提取。

5.1.1 预处理过程

预处理过程由两个模块并行处理：字段列属性预处理和水印信息预处理。

字段列属性预处理过程的主要处理步骤如下：

(1)扫描要嵌入水印的整个汉字字段，通过语义分析[13-19]找出属性值的共同点，并用最相关的某个非上下结构的汉字 $comm$ 表示，如地址列一般是区/市/县/镇等；

(2)计算元组的消息认证码，确定水印嵌入间距，选择嵌入水印的目标元组；

(3)对目标元组中的属性重新编号并计算其特征值。

水印信息预处理过程的主要处理步骤如下：

(1)确定要嵌入的水印信息。如果欲嵌入的信息是文本，需要查询 ASCII 表获得水印的 ASCII 值，并将其表示成二进制比特位；若嵌入图片需要记录图片的二值流，嵌入图文时，

将文本信息对应的二进制比特位追加在图片的二进制位后，这部分二值流记为 *wm_content*。在欲嵌入水印信息后再添加 4 个字节，前两个字节用来记录嵌入文本信息的字节数，后两个字节用来表示嵌入水印类型。只嵌入文本信息时用 00000000,00000001 表示，只嵌入图片时用 00000000,00000010表示，同时嵌入图片和文字时用 00000000,00000011 表示。这 4 个字节也当成水印信息嵌入数据库字段，以便水印的检测与盲提取。

(2)在起始位前加上 00000001，结束末加上 00000100，作为起始标志和结束标志。为方便描述，将添加有起始和结束标志的水印信息称为“循环水印块”，记作 *wm_block*。

5.1.2 水印嵌入过程

水印嵌入的主要步骤如下：

(1)按位对比目标元组特征值与 *wm_block*，记录不同的位 *b*，当 *wm_block* 位数不足时，从开头循环使用；

(2)对编号为 *b* 的属性值进行水印嵌入，也就是插入或删除某个独体/左右/包围结构的 *comm* 字。

字段列属性预处理和水印信息预处理过程完成后，执行水印嵌入子模块，完成整个水印嵌入流程，见图 1。

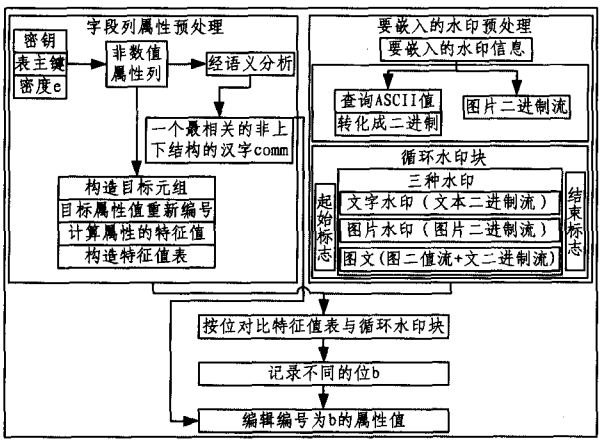

图 1 水印嵌入完整流程

5.1.3 水印检测提取过程

水印检测提取的主要步骤如下(见图 2)：

(1)用只有数据库所有者才知道的密钥 K_1、水印嵌入间距 *e* 和元组主键计算并选出可能嵌入水印的元组 *T*；

(2)计算元组 *T* 中汉字属性的特征值，并且从起始位按照长度 *wm. length* 对特征值分组，将不满足起始和结束标志的分组去掉；

(3)每组特征值都去掉起始标志 00000001 和结束标志 00000100，根据多数投票原则选出出现次数最多的二进制位流 *wm'*；根据 *wm'* 的最后 4 个字节中的前两个(记作 *wm_type*)判断水印类型，后两个字节(记作 *wm_cbytes*)判断嵌入文本的字节数；

(4)若 *wm_type* 为 00000000,00000001，则将选出 *wm_content'* 的二进制位流转化成十进制，然后查询 ASCII 码表，找出对应的信息，该信息就是嵌入的文本水印信息；若 *wm_type* 为 00000000,00000010，表示提取出的水印类型为图片，根据 *wm_content'* 的值重绘图片，该图片就是提取出的水印信息；若 *wm_type* 为 00000000,00000011，表示提取出的水印包括图片和文字，根据 *wm_cbytes* 的值确定文字的字节数，根据该值将图片和文字分开，然后分别如上两种处理恢复出信息。

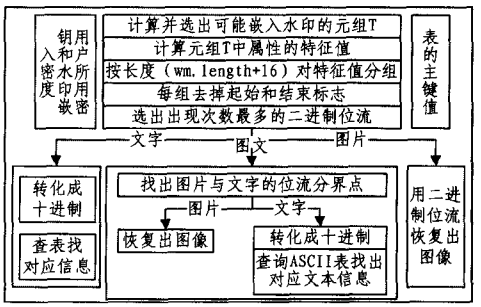

图 2 水印检测提取过程

5.2 水印嵌入实验

本文基于汉字字段的数据库水印嵌入方案中嵌入水印的属性列由用户选定。此处设用户选定的汉字字段集合为$\{A_1, A_2, \cdots, A_m\}$，同时用 *A* 表示该集合，并且关系表 *R* 的主键属性为 *P*，则水印嵌入过程每次从集合 *A* 中选出一个属性列 A_i，然后将该属性列与主键属性列 *P* 组合成一个新的关系表 $R(P, A_i)$，至此完成了目标列选择。此处需要注意的是，该处理过程完全是逻辑意义层面上的，其在物理意义上是虚拟的。也就是说，该操作不引起数据库原关系表的任何变动。

选择水印嵌入元组，依据元组的主键属性值以及某个私有密钥 K_1 确定。如果 t_i 表示某个元组，$t_i.P$ 表示该元组的主键属性值，则水印嵌入元组的选择如式(1)所示：

$$j = F(t_i.P, K_1) \bmod e \tag{1}$$

式中，$F(t_i.P, K_1)$为元组 $t_i.P$ 的消息认证码；参数 *e* 为调节因子，用以控制嵌入水印的元组的比例。式(1)中如果 $j=0$，表示该元组被选定用于水印嵌入。先后选出的用于水印嵌入的元组依次编号为 $1,2,3,\cdots,n,\cdots$。

为了提高水印的鲁棒性，在选出的满足水印嵌入的元组中重复嵌入水印信息。这样，在水印局部被破坏的情况下，仍能正确地提取出水印信息，而且本算法是基于单个属性列的，所以对部分拷贝数据具有抵抗性。

从人才库的人员工作倾向地点字段中选出的需要嵌入水印的元组属性值为：

1 北京朝阳 2 北京海淀 3 山东泰安 4 山东菏泽 5 山东济宁 6 山东聊城 7 河南开封 8 上海 9 北京东城区 10 北京崇文区 11 北京怀柔区 12 山东滨州 13 山东德州 14 湖北宜昌 15 湖北枝江市 16 湖北黄冈 17 湖北荆州 18 湖北汉川 19 辽宁阜新 20 辽宁铁岭 21 辽宁抚顺 22 辽宁鞍山 23 吉林通化 24 吉林长春 25 吉林白山 26 辽宁朝阳 27 山东枣庄 28 山东潍坊 29 山东莱芜 30 山东济南 31 山东临沂 32 江西南昌 33 山西太原 34 山西忻州 35 山西晋中市 36 山西吕梁市 37 山西临汾 38 陕西西安 39 陕西咸阳 40 陕西宝鸡市 41 陕西铜川市 42 陕西渭南 43 陕西延安 44 陕西榆林 45 浙江杭州 46 浙江绍兴 47 浙江丽水 48 浙江金华 …

运用规则 1 和规则 2 计算各属性值的特征值如下(见表 1)：

B(北京朝阳)=B(北)||B(京)||B(朝)||B(阳)=010 011 010 010

$t(1)=1$

B(北京海淀)=010 011 010 010

$t(2)=1$

B(山东泰安)=001 001 011 011

$t(3)=0$

B(山东菏泽)=001 001 011 010

$t(4)=1$

B(山东济南)=001 001 010 001

$t(5)=0$

B(山东聊城)=001 001 010 010

$t(6)=0$

……

表 1　属性值的特征值

元组编号 (i)	特征值 t(i)	元组编号 (i)	特征值 t(i)	元组编号 (i)	特征值 t(i)
1	1	17	0	33	0
2	1	18	0	34	0
3	0	19	0	35	0
4	1	20	1	36	1
5	0	21	1	37	0
6	0	22	1	38	1
7	0	23	1	39	0
8	0	24	0	40	0
9	0	25	1	41	1
10	1	26	1	42	0
11	1	27	1	43	1
12	0	28	0	44	0
13	0	29	0	45	0
14	0	30	0	46	0
15	1	31	0	47	0
16	1	32	1	48	0
…	…	…	…	…	…

欲嵌入“才鼎”二字，它的 ASCII 码对应的二进制串为 0100，1101，0011，1011；0100，1001，0101，1010，0000，0000，0000，0001，0000，0000，0000，0100；在起始位前加入 00000001 和结束末加上 00000100，作为起始标志和结束标志。这样得到 80 位的水印信息。

将水印信息和 $t(i)$ 进行对比如下：

00000001010011010011101101001001010110100000000 0…

11010000011000110001111011100001000101001010000 0…

不同的位有 1、2、4、8、11、13、14、15、19、22、24、25、27、29、34、37、38、39、41、43、…

所以需要对第 1、2、4、8、11、13、14、15、19、22、24、25、27、29、34、37、38、39、41、43、…个元组进行变换，以嵌入水印信息。

例如：在第一个元组“北京朝阳”中添加“区”，变为“北京朝阳区”，属性的特征值由 $t(1)=1$ 变为 $t(1)=0$，由此在该元组中嵌入水印；第 15 个元组“湖北枝江市”，去掉“市”，变为“湖北枝江”，属性的特征值由 $t(15)=1$ 变为 $t(15)=0$；第 14 个元组“湖北宜昌”中添加“市”，变为“湖北宜昌市”，属性的特征值由 $t(14)=0$ 变为 $t(14)=1$。

嵌入水印后的元组属性值分别为：

1 北京朝阳区 2 北京海淀区 3 山东泰安 4 山东菏泽市 5 山东济宁 6 山东聊城 7 河南开封 8 上海市 9 北京东城区 10 北京崇文区 11 北京怀柔 12 山东滨州 13 山东德州市 14 湖北宜昌市 15 湖北枝江 19 辽宁阜新市 20 辽宁铁岭 21 辽宁抚顺 22 辽宁鞍山市 23 吉林通化 24 吉林长春市 25 吉林白山市 26 辽宁朝阳 27 山东枣庄市 28 山东潍坊 29 山东莱芜市 30 山东济南 31 山东临沂 32 江西南昌 33 山西太原 34 山西忻州市 35 山西晋中市 36 山西吕梁市 37 山西临汾市 38 陕西西安市 39 陕西咸阳市 40 陕西宝鸡市 41 陕西铜川 42 陕西渭南 43 陕西延安市 44 陕西榆林 45 浙江杭州 46 浙江绍兴市 47 浙江丽水 48 浙江金华 …

5.3　水印检测提取实验

水印提取时选择的元组由元组的主键属性值以及只有数据库拥有者才知道的水印嵌入时用到的私有密钥 K_1 确定。

$$j=F(t_i.P,K_1) \bmod e$$

式中，$F(t_i.P,K_1)$ 为元组 $ti.P$ 的消息认证码，调节因子 e 也是水印嵌入时选用的。依次选出 $j=0$ 的元组，计算出 $j=0$ 的元组对应的汉字属性的特征值。根据多数投票原则，选出长度最长且出现次数最多的 0、1 串，并通过排查选出以 00000001 开头、以 00000100 结尾的串，然后去掉前 8 个特征值(00000001)和后 8 个特征值(00000100)，最后剩下的就是水印信息的二进制位。对于本文中的实验，最后剩下的 *wm_content* 32 位特征值为 0100 1101 0011 1011 0100 1001 0101 1010，查 ASCII 表得到水印信息为“才鼎”。

结束语　本文关系数据库数字水印方案主要基于汉字字段。嵌入的水印具有不可见性，并且不影响数据库的可用性，可实现盲提取。数据库中的数据操作是一个动态的过程，数据值处在不断更新的情况下。因此，随着元组的添加或删除以及属性值的改变，嵌入的水印也必须能够随之不断地更新，即必须能够为加入或修改的元组重新计算水印的值，这一点本算法还不能实现。下一步应该添加数据库嗅探器，实时统计数据库的更新。当更新数据量超过一定的阈值时，启动重新嵌入水印过程。

参 考 文 献

[1] 牛夏牧，赵亮，黄文军，等. 利用数字水印技术实现数据库的版权保护[J]. 电子学报，2003，31(12)：2050-2053

[2] 张勇，赵东宁，李德毅. 关系数据库数字水印技术[J]. 计算机工程与应用，2003，25：193-195

[3] Agrawal R，Kiernan J. Watermarking Relational Databases[C]// Proc. of the 28th VLDB Conference. HongKong，China，2002：155-166

[4] Sion R，Atallah M，Prabhakar S. On Watermarking Numeric Sets [C]// Proc. of the Workshop on Digital Watermarking. Seoul，Korea，2002：1-15

[5] Li Y，Swarup V，Jajodia S. Constructing a Virtual Primary Key for Fingerprinting Relational Data[C]// Proc. of Digital Rights Management Workshop. Washington，DC，USA，2003：133-141

[6] Sion R，Atallah M，Prabhakar S. Rights Protection for Relational Data[J]. IEEE Journal of Trans. On Knowledge and Data Engineering，2004，16(6)：1509-1525

[7] Sion R，Atallah M，Prabhakar S. Ownership Proofs for Categorical Data [C]// Proc. of the IEEE International Conference on Data Engineering. Boston，2004：584-596

[8] Zhang Z，Jin X，Wang J，et al. Watermarking Relational Database Using Image[C]// Proc. of the Third International Conference on Machines Learning and Cybernetics. Shanghai，2004：1739-17

(下转第 190 页)

的实验如图4所示，与数据集 T40I10D100k 的测试结果一样，Bala_Tree 所需内存还是最少。DSTree 算法利用字母表顺序建树，并且树中保留大量过期垃圾节点，造成空间开销巨大。CPS-tree 和 Bala_Tree 算法都利用了重构技术，然而 Bala_Tree 算法更简洁，所以在任意大小滑动窗口下，无论稀疏或者稠密数据集，算法 Bala_Tree 都显示出空间优越性。

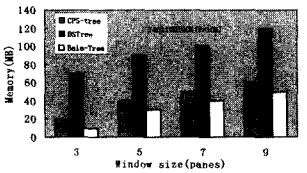

图3 在数据集 T40I10D100k 下比较内存大小

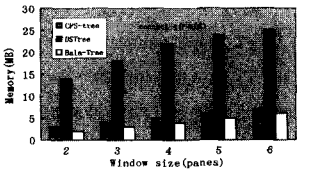

图4 在数据集 Connect-4 下比较内存大小

接下来试验比较 DSTree、CPS-tree 和 Bala_Tree 算法的时间开销。在图5中，对于数据集 T40I10D100K，设定滑动窗口 $w=4$，支持度分别为(25,20,15,10)；在图6中，对于稠密数据集 Connect-4，设定滑动窗口 $w=2$，支持度分别为(99,97,95,93,91)，可以看到无论是在较高支持度还是较低支持度下，Bala_Tree 算法时间开销都小于 DSTree 和 CPS-tree 算法，并且随着支持度降低差距越来越大。虽然 Bala_Tree 算法重构树花费了额外花销，然而实验中发现频繁项集挖掘阶段是最耗费时间的，基于支持度降序排列树利用 FP_growth 算法挖掘频繁项集效率非常高，在重构次数有限的情况下完全可以忽略重构时间花销，所以 Bala_Tree 算法总计时间花销明显小于 DSTree 算法。

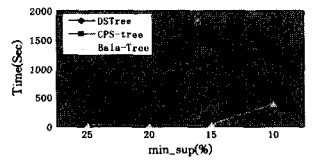

图5 在数据集 T40I10D100k 下比较时间大小

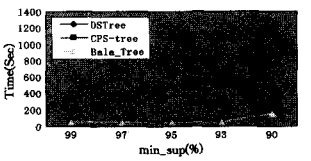

图6 在数据集 Connect-4 下比较时间大小

结束语 提出一种新颖的数据流中挖掘频繁项集的算法 Bala_Tree，Bala_Tree 算法对树中节点设置簇计数器，并且引入树重构思想得到一棵基于支持度降序树，不但节省了内存开销，而且适合利用 FP_growth 方法挖掘频繁项集。Bala_Tree 算法达到了时间和空间相对均衡，优于其他同类算法。

参 考 文 献

[1] 刘旭，毛国君，孙岳，等. 数据流中频繁闭项集的近似挖掘算法[J]. 电子学报，2007，35(5)

[2] Han J，Pei J，Yin Y. Mining Frequent Patterns without Candidate Generation[C]// Proc. ACM-SIGMOD Int'l Conf. Management of Data. 2000

[3] Manku G S，Motwani R. Approximate frequency counts over data streams[C]// Proceedings of the 28th international conference on very large data bases. Hong Kong，2002

[4] Giannella C，Han J，Pei J，et al. Mining frequent patterns in data streams at multiple time granularities[M]. Data Mining：Next Generation Challenges and Future Directions. AAAI/MIT Press，2004

[5] Chi Y，Wang H，Yu P，et al. Moment：maintaining closed frequent itemsets over a stream sliding window[C]// Proceedings of the 4th IEEE international conference on data mining. Brighton，UK，2004：59-66

[6] Leung C K-S，Khan Q I. DSTree：a tree structure for the mining of frequent sets from data streams[C]// Proc. ICDM. 2006：928-932

[7] Khairuzzaman S，Ahmed C F，Jeong B-S，et al. sliding window-based frequent pattern mining over data streams[J]. Information Sciences，2009，179：3843-3865

[8] 李国徽，陈辉. 挖掘数据流任意滑动时间窗口内频繁模式[J]. 软件学报，2009，10(19)：2585-2596

[9] Tanbeer S K，Ahmed C F，Jeong B-S，et al. CP-tree：a tree structure for single-pass frequent pattern mining[C]// Proc. PAKDD. 2008，5012：1022-1027

[10] Koh J-L，Shieh S-F. An efficient approach for maintaining association rules based on adjusting FP-tree structures[C]// Proc. the DASFAA. 2004：417-424

[11] Blake C L，Merz C J. UCI Repository of Machine Learning Databases[D]. University of California-Irvine，Irvine，CA，1998

(上接第166页)

[9] 中华人民共和国教育部. 中华人民共和国国家标准 GB2312-80《信息技术、信息交换、用汉字编码字符集、基本集》[S]. 1980

[10] 左双勇. 基于汉字字型结构的文本数字水印算法[J]. 计算机与现代化，2010，1006-2475，08-0029-04，29-32

[11] Sion R，et al. Watermarking Relational Databases[R]. CERIAS. 2002

[12] 张金永. 基于非数值属性的数据库数字水印算法研究[D]. 兰州大学，2010

[13] Ido D Lillian L，Fernando C N P. Similarity-based Models of Word Co-occurrence Probabilities[J]. Machine Learning，1999，34：43-69

[14] Jiang J，Conrath D. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy[C]// Proceedings of International Conference Research on Computational Linguistics. 1997：19-33

[15] 黄果，周竹荣. 基于领域本体的概念语义相似度计算研究[J]. 计算机工程与设计，2007，28(10)：2460-2463

[16] 吴健，吴朝晖，李莹，等. 基于本体论和词汇语义相似度的 Web 服务发现[J]. 计算机学报，2005，28(4)：595-602

[17] 夏天. 汉语词语语义相似度计算研究[J]. 计算机工程，2007，33(6)：191-194

[18] 陈杰，蒋祖华. 领域本体的概念相似度计算[J]. 计算机工程与应用，2006，42(33)：163-166

[19] 赖院根，等. 概念语义相似度计算与参数估计[J]. 情报杂志，2009，28(8)：148-152