基于BIBD的数据库压缩水印技术

栗新宇　王以刚
（东华大学计算机科学与技术学院　上海 201620）

摘　要　根据关系数据库的特殊性，结合组合设计的区组设计，本文将BIBD(平衡不完全组设计理论)引入关系数据库数字水印算法。该算法具有隐蔽性好，最大限度降低了对原有数据库的修改，水印恢复不需要原始数据库，且具有很好的灵活性等特点。

关键词　数据库水印，区组设计，压缩水印

Compressed Watermarking for Database Based on BIBD

LI Xin-Yu　WANG Yi-Gang[1]
(College of Computer Science and Technology, Donghua University, Shanghai 201620)

Abstract　According to the specialties of the relational database, this paper proposes a digital watermarking algorithm based on BIBD(Balanced Incomplete Block Design). This algorithm has the characteristics of good hiding and flexible, moreover it can largely reduce the modifications to the database and it doesn't need the original databases in the recover progress.

Keywords　Database watermark, Block design, Compressed watermark

1　引言

数字水印技术是20世纪90年代兴起的一门崭新的技术，它通过在数字产品中嵌入可感知或不可感知的信息来确定数字产品的所有权或检验数字内容的原始性[1]。随着数据库技术的不断发展及数据库管理系统的广泛应用，数据库也面临着版权保护的问题。但目前国内外关于数据库水印的研究，多集中于数值型数据，而在非数值型数据方面的研究还非常有限[2]。只针对数值型数据的字段不仅不符合现实的要求，更重要的是，方便了破坏者对水印算法进行攻击。本文结合二维空间标识法，将组合设计中的区组设计引入数据库水印中，通过压缩水印信息、用户自定义等方法，实现了在关系数据库中，灵活地对应多种类型数据的水印。

2　数据库水印技术的研究

数据库水印是指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，在不破坏数据库内容和可用性的前提下，达到保护数据库安全的目的[3]。

就目前的相关文献资料[4,5]，水印嵌入的过程主要是解决三方面的问题：嵌入什么样的水印信息、嵌入到哪里(哪个元组下的哪个字段)以及如何嵌入。为此，本文提出的新方法如下。

首先通过压缩算法，由含有版权信息的字符串或图片信息得到待嵌入的水印信息。对于数据库元组的选定，本文使用的是一维转换为二维的元组标识算法，并以此将数据库中的元组进行分组、排序，得到元组的排列方式。在每一个元组分组的内部，运用BIBD算法对数据库中的元组进行筛选，最终得到待嵌入水印信息的元组。对于特定元组字段的选择以及水印位的替换，本文都采用用户自定义的方法，由用户指定相应字段数据以及可以容忍的最大修改量，继而将水印信息嵌入到数据库中。

3　平衡不完全组设计理论

首先引入区组设计中的平衡不完全组设计，简称BIBD[6,7]，建立基于组合设计的区组设计模型。令 X 是 v 个元素的任意集合，而令 B 为 X 的 k ($2<k<v$)个元素子集的集合 $B_1, B_2, \cdots, B_N$，这些子集叫做区组。设 M 是元素数，N 是区组数，k 是在每个区组中元素的个数，r 是包含每个元素的区组的个数。如果 X 的每一对元素恰好同时出现在 λ 个区组中，则 B 就是 X 的一个平衡区组设计。如果 $k<v$，且 B 是平衡的，则有一个平衡不完全组设计BIBD。

每一对元素恰好同时出现在 λ 个区组中，意味着没有两对元素同时出现在 λ 个区组中，同时也保证了元素的利用率和均衡性，换言之，当区组数一定时，可以得到最小的元素集合。正是因为BIBD具有了这样的性质，我们将BIBD引入到了数据库水印技术中。此时，元素即为数据库中的元组，区组即为元组集合，则在水印数据量一定的情况下，可以最大限度地降低修改原有数据库的元组数，减少了对原有数据库数据的修改。

引理　在BIBD中每个元素含于 $r=\frac{\lambda(M-1)}{k-1}$ 个区组中。　(1)

推论　在BIBD中，有

$$bk=Mr \qquad (2)$$

证明过程详见参考文[6]。由于BIBD算法要求每对元素至少可以同时出现在1个区组之中，即 $\lambda_{min}=1$，因此本文所使用的数据库，要求每一个元组中至少有两个以上的字段可以进行少量的修改。

栗新宇　硕士研究生；王以刚　教授，主要研究方向：计算机网络、信息安全。

4 基于 BIBD 的数据库水印算法

4.1 用户自定义

数据库的拥有者在嵌入水印信息之前，需要指定以下 4 方面内容：

(1)某一个或几个字段，其值对于整个数据库而言是极为重要的，这些数据的丢失或是篡改将对数据库的正常使用造成很大的损害。称这样的字段为 MSA(Most Significant Attributes)。

(2)某两个或两个以上字段，其值是可以做微小改变的，如一些数值型或字符型。称这样的字段为 LSA(Least Significant Attributes)。

如果数据库中的每个元组有 2 个字段可以被修改，即可以嵌入 2 个字符，则每个元组便可以被重复 2 次，以此类推，不同的字符可以嵌入到相同的 k，$k-1$，…，$k-n$，…$(k-n>0)$个元组中。通过用户定义，便可以得到 BIBD 算法中的 k 值。

(3)若 LSA 是数值型，则指定可修改的范围；若是字符型，则指定可以被替换的字符。

(4)指定常数 C，相当于用户密钥，用于水印信息的压缩。

4.2 数据库元组的标识

对关系数据库而言，其标识算法的输入是其元组的内容，输出是元组标记 ID[8]。目前主要的关系数据库水印算法采用的元组标识都是完全依赖于元组主键的[9]。而本文采用将一维空间映射到二维空间的方法，通过读取每一元组 MSA 的内容比特流，每读取一个比特，就将 x 坐标加 1，如果这个比特是 1 就将 y 坐标加 1，如果是 0 就将 y 坐标减 1，最终将终点坐标作为元组的标记 ID，这样便得到了每一元组所对应的坐标。

在这其中必然会存在具有相同坐标的元组，故而把元组按照相同坐标个数多少排序，重复的越多越排在前面，相同次数的，再按照主键排序。这样就排列出了待嵌入水印的元组的次序。

数据库的标识算法如下所示。

```
(1) 建立结构体 Coordinates,用于记录各元组的坐标值;
struct Cordinates{
    int X;    int Y;
} Coordinates[n];
(2) for(int k=0; k<n; k++)
(3) byte[] bsStr = MSA[k].getBytes();  // bsStr []记录的是MSA 的比特流
(4) for(int i=0; i<bsStr.length; i++){
(5) String byteStr = Integer.toBinaryString(bsStr[i]);
(6) Coordinates.X += byteStr.length();
(7) for(int j=0; j<byteStr.length(); j++){
(8) if (byteStr.charAt(j)=='1') Coordinates.Y++;
(9) else     Coordinates.Y-;
```

4.3 水印压缩算法

通常来说，不论是版权字符串还是含有版权信息的图片，转换为数字串后数值量会非常大。如果直接将如此庞大的数字串嵌入到原来的数据库当中，这本身就是对原有数据库极大的伤害，势必将造成数据库中很多原有数据的改变。因此本文采取了水印压缩算法，在保证水印信息不丢失的情况下，很简单地就减少了需嵌入的水印量。同时，也在一定程度上达到加密的效果。

4.3.1 版权字符串

对于含有版权信息的字符串(假设其长度为 N)，首先将每个字符转换为 ASCII 码，组成数组 copyright[]，因为常用字符的 ASCII 码最大不超过 128，所以用户自定义常数 C 可以取任意大于 128 的整数，系统也会根据水印字符串，自动提示用户压缩率最好的 C。最后将用户指定的 C 对于 copyright[i]的余数 Items[i]作为最终的待嵌入版权字符串水印信息。

压缩字符串的算法如下所示。

```
(1) for(int C=129;C<129*2;C++){
(2) for(int i=0; i<copyright.length; i++)
(3) Item[i] = C%copyright[i];
(4) csStr[i] = Integer.toBinaryString(Item[i]);  //csStr[i]记录压缩后的二进制代码
(5) countNum[i] += csStr[i].length();     //countNum[i]记录压缩后的二进制代码的长度
(6) bestC = findMin(conutNum);          //找到使压缩后的字符串长度最短的 C
(7) 输出 bestC,提示用户
```

4.3.2 版权图片

对于大小为 $M*N$ 的含有版权信息的图片来说，首先，将图片中的第 i 行第 j 列的像素值依次保存到数组 Num[i][j]($i\subset[0,M]$，$j\subset[0,N]$)中，然后对 Num[i][j]进行压缩。因为像素值最大不超过 255，所以用户自定义常数 C 可以取任意大于 255 的整数，系统也会根据水印图片，自动提示用户压缩率最好的 C，最后将用户指定的 C 对于 Num[i][j]的余数 Items[i]作为最终的待嵌入水印信息。

压缩图片与压缩字符串类似，只是 $C=256$ 开始循环，并且将(3)改成 Item[i] = C%Num[i][j]；

无论是字符串还是图片信息，系统都将记录下最终需嵌入的水印信息长度 q。

4.4 水印嵌入算法

综上所述，本文所提出的基于 BIBD 的数据库压缩水印算法的具体实施步骤如下：

(1)根据数据库标识算法，对元组进行标识，得到 ID 坐标

(2)将元组按照 ID 坐标进行排序，坐标相同的元组组成元组集合 S_i

在 S_i 内部，再按照主键排列相应次序，并对每一个 S_i 集合统计所包含元组个数 M_i。

(3)根据 BIBD 公式(1)、(2)及 M_i，计算对于集合 S_i 最多能嵌入的水印字符数 b_i

根据公式(1)，$r=\dfrac{\lambda(M-1)}{k-1}$，及公式(2)，$bk=Mr$，则 $b=\dfrac{\lambda M(M-1)}{k(k-1)}$。又因为 λ 表示的是每一对元组恰好同时嵌入相同的水印字符的个数，为使水印信息重复嵌入的次数尽量少，这里取 $\lambda=1$，因此，

$$b_i=\frac{M_i(M_i-1)}{k(k-1)} \tag{3}$$

这样对于任何一个元组集合 S_i 都可以得到相应的 b_i。

(4)由需嵌入的水印量 q，向所有 $b_i\geqslant q$ 的 S_i 集合中依次嵌入水印信息

水印的嵌入过程如图 1 所示。

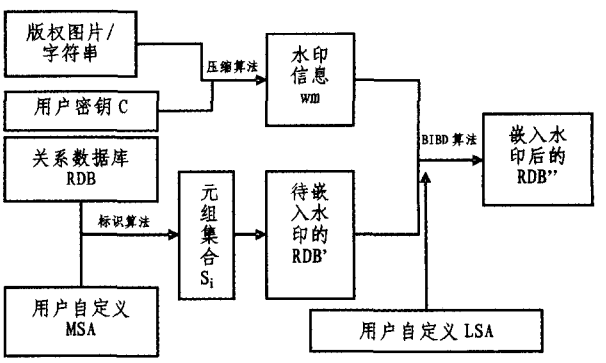

图 1 水印的嵌入过程

(下转第 102 页)

```
〈element name="resourceSize" type="xsd:int"/〉
〈/complexType〉
〈complexType name="SearchResourceReqStruct"〉
〈element name="token" type="xsd:string"/〉
〈element name="resourceName" type="xsd:string"/〉
〈/complexType〉
〈/types〉
```

4.4 网格管理器的连接

下面通过在 WSDL 文件中创建的 Web 服务存根定义到网格管理器的连接：

CSCLGridAppPortTypeStub stub = new CSCLGridAppPortTypeStub(null, "http://localhost:8080/axis2/services/CSCLGridApp")。

结束语 本文提出了利用网格技术来解决协作学习系统中的资源共享和管理问题，搭建基于网格的协作学习系统平台。将网格计算技术和 CSCL 有效结合，能摆脱各类学习系统的运行环境、开发语言等技术方面的束缚，建立稳定、安全的交流平台，真正达到协作学习教学资源高效管理和充分利用的目的。尽管我们的工作还是很初步的，但通过这次实验，证明了该方案是可行的、正确的，为今后工作打下了良好的基础。进一步的工作包括将该平台由局域网推广到 Internet，并在全国范围内建立一个基于网格的协作学习平台。

参 考 文 献

1 黄荣怀. CSCL 的理论与方法[J]. 电化教育研究，1999(6)：25～30

2 赵建华，李克东. Web 环境下协作学习系统开发的现状及趋势[J]. 电化教育研究，2004(2)：28～34

3 都志辉，陈渝，刘鹏. 网格计算[M]. 北京：清华大学出版社，2002

4 都志辉. 网格计算：宏伟蓝图？海市蜃楼？[EB/OL]. http://industry.ccidnet.com/art/9/20030123/37462_1.html，2006-7-10

5 Brown M. Build agrid using Web services standards[EB/OL]. http://www-128.ibm.com/developerworks/edu /gr-dw-gr-moviel-i.html，2006-7-15

（上接第 98 页）

4.5 水印恢复算法

水印信息的提取过程几乎是嵌入的逆过程，只是在检测到水印位后，为避免因为内容在使用过程中被更新，而造成计算出的元组二维坐标出错，在提取水印时，采用绝大多数表决法 MVM(Majority Voting Method)，以得到原始的水印信息。

水印的提取过程如图 2 所示。

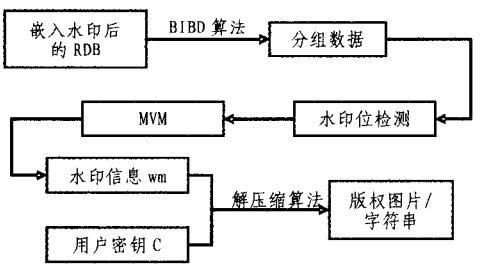

图 2 水印的提取过程

5 仿真实验

实验中使用到的关系数据库共有记录 7600 条。其中可做一定修改的字段为 2 个，即 $k=2$。MSA 字段为地区分店名称。嵌入的水印信息为"DongHua Authority@2007"，原始长度为 149，此时，系统提示最佳压缩率 $C=234$，就按此时的 C 进行压缩，得到 $q=97$。实验时，测得数据库中共有 3 组元组集合可以嵌入水印信息。

对嵌入水印信息后的数据库进行 10 次攻击，随机修改、增加数据库中不同的字段值，而后对水印进行鲁棒性测试，仿真结果如下。

表 1 子集修改攻击

修改比例	68%	66%	64%	62%	≤60%
准确恢复	5 次	7 次	8 次	10 次	10 次

表 2 子集增加攻击

修改比例	110%	104%	100%	96%	≤95%
准确恢复	5 次	7 次	9 次	10 次	10 次

由表 1 和表 2 不难看出，对于子集修改或增加攻击时，攻击的比例越大，对水印的破坏程度就越大，就越不容易提取出正确的水印信息，因而准确恢复的比例就越低。

因为本文提出的嵌入水印算法是根据元组的 MSA 得到的 ID 值以及该元组对应的主键进行的分组排序，与元组本身的顺序无关，所以对于元组重新排序并不会影响水印信息的提取。

总结 本文首次将组合数学中区组设计原理引入数据库水印，并提出了压缩数据库水印技术。在嵌入含有版权信息水印的同时，最小化对原有数据库的修改。通过用户自定义的方式，可以针对不同的数据库中字段的重要性做相应的调整，更加灵活地嵌入水印信息，并且使水印的嵌入过程具有更强的隐蔽性。虽然利用本算法要求原有数据库中至少有 2 个以上的字段是可以进行少量修改的，但在对于大规模的数据库而言，这也是可以接受的。

参 考 文 献

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

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

3 朱勤，于守健，乐嘉锦. 数据库水印研究与进展. 计算机工程与应用，2006(29)：198～201

4 Agrawal R, Kieman J. Watermarking Relational Databases. In: Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002. 155～166

5 张勇，赵东宁，李德毅. 水印关系数据库. 解放军理工大学学报(自然科学版)，2003，4(5)：1～4

6 Brualdi R A. Introductory Combinatorics, 3E. 机械工业出版社，2006

7 Rosen K H. Discrete Mathematics and Its Applications. 4E. 机械工业出版社，2002

8 孙松，孙淑玲，邵佩英. 多级安全数据库中标记生成规则的一致性. 计算机工程与应用，2001(16)：123～127

9 Atallah M S R, Prabhakar S. Rights protection for relational data. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data San Diego, Califonia: ACM SIGMOD, 2003. 98～109