计算机取证过程中基于约束的数据质量问题研究

刘 琴
(华东政法大学信息科学技术系 上海 200042)

摘 要 为了发现线索,提高数据质量,提出了一种应用于计算机取证领域的基于约束的数据修复算法。首先,利用等价类,针对不同的约束对数据进行初始化;然后,对初始化阶段发现的有问题的数据进行修正,修正值依据约束类型的不同而取不同的值;最后,根据函数依赖集合和其他约束集,对经过修复的单元格集合重新生成问题单元格集合,如果依然存在问题单元格集合,则继续修复,直到不存在问题单元格为止。实验数据证明了所提方法的有效性和高效性。

关键词 计算机取证,数据修复,函数依赖,算法

中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.04.028

Study on Data Quality Based on Constraint in Computer Forensics

LIU Qin
(Department of Information Science and Technology, East China University of Political Science and Law, Shanghai 200042, China)

Abstract To find clues and improve the quality of data, this paper proposed a data repairing algorithm based on constraint in the field of computer forensics. First, using the equivalence class, the algorithm initializes the data according to different constraints. Second, it repaires the problem data found in the initialization phase and gets different values according to the type of constraint. Third, the repaired collections are regenerated according to function dependence set and other constraint sets. If the problem data still exist, the algorithm continues to repair until there is no problem so far. Experiment proves the validity and high efficiency of the proposed method.

Keywords Computer forensics, Data repair, Functional dependency, Algorithm

1 引言

在计算机取证过程中,集成来自不同数据源的数据(主要来自不同的数据库管理系统)以便更好地发现线索,已经成为比较普遍的侦查手段。数据源的异构性,导致集成后的数据之间存在冲突,使得对现实世界的同一对象有不同的数据描述。这些不一致、冲突甚至错误的数据,会造成决策失误和经济损失。因此,关注数据质量并且能在知晓质量不高的情况下通过执行操作来提高数据的质量,显得非常有意义。

对于数据质量,主要有两个方面的基本评估指标[1-2]:1)从用户的角度来看,数据是可以信赖的,其中包括完整性、一致性、唯一性、精确性等;2)从用户的角度看,数据是可以使用的,包括稳定性、时间性等。本文主要考虑数据质量中的数据一致性问题。一致性是指同一实体的同一属性值在不同的系统或数据集成中的表达一致。

目前,已经有很多研究机构提出了基于函数依赖的数据一致性处理方法。函数依赖已被广泛用于数据库的规范化设计中,已成为一种重要的描述数据一致性的方法。假设 R 为一关系,X 和 Y 分别为关系 R 上的属性集,函数依赖 $X \rightarrow Y$ 表示对于任意一个可能的关系实例 r,都不可能存在两个元组在 X 上的属性值相等而在 Y 上的属性值不等。在一个关系 R 上可以定义一个函数依赖集合,如果关系 R 能满足该集合中的所有函数依赖,那么就可以认为该关系数据质量达到要求;否则认为该关系数据存在质量问题,需要进行改进。改进的方法主要有两种,一是直接删除不符合要求的记录,这种方法比较简单,但同时也会丢失很多信息;另一种方法是修改属性值,使其满足预置的一致性要求,这种方法复杂,但能最大限度地保留原始信息,也是目前及本文的研究重点。目前的研究[2-4]大都针对函数依赖集合进行数据修复;函数依赖关系属于模式层的约束,较少有研究考虑到实例层的约束,只有文献[5]考虑了类似于外键关系的包含关系 IND(包含关系还是侧重于模式层的约束),文献[5-7]考虑了硬约束、等值约束等关系。这些工作中所考虑的一致性约束条件都较局限,没有考虑领域数据的特点,且无法进行简单的扩展。本文考虑了领域数据的特点,提出了若干种除函数约束以外的实例层约束,并提供了相应的算法来提升数据质量。本文的主要贡献有两方面:1)除了考虑函数依赖外,还考虑了其他约束,尤其考虑了多表之间的关联约束;2)将数据质量的研究成果运用到计算机取证领域,并考虑了领域特征,算法中除了给出数据修补方案外,还给出了可作为线索使用的错误集。

到稿日期:2017-03-29 返修日期:2017-06-16 本文受基于蜜场技术的电子数据取证相关法律问题研究(13CFX054)资助。
刘 琴(1975—),女,硕士,副教授,主要研究方向为计算机取证、Web 数据库等,E-mail:liuqin@ecupl.edu.cn(通信作者)。

2 相关工作

提高数据一致性的研究大致可以分为两类:基于统计的方法和基于数据依赖的方法。本文研究基于数据依赖的方法。根据语义处理层次的不同,数据依赖方法又可以分为模式层依赖和实例层约束两种不同的方法。

模式层依赖主要依据数据模式的依赖(如关系模式的函数依赖)来进行数据一致性的处理。文献[3]给出了一个近似算法,对违背函数依赖而导致数据不一致的数据进行最小代价的修改,最小代价是个 NP 难问题。该算法引入一个常数因子,使得对于固定的函数依赖集和任意一个不一致的数据库,修复距离(通过修改值的数量和修改元组的权值来度量)都能保持在常数因子范围内。文献[2]的目标是通过最少的更新操作来修复数据,优化目标是得到代价最小策略,其通过引入属性等价类的方法提出了贪心算法来进行处理。该算法主要考虑了函数依赖关系,还引入了类似外键关系的包含关系 IND。文献[4]不仅提出了数据修复算法来适应函数依赖,还提出了约束修复算法,通过增加属性到约束中来修复数据库的不一致性。两种算法都主要考虑了函数依赖。

实例层约束主要根据数据实例必须满足的约束(如硬约束,即指定单元格的值)来进行数据一致性的处理,这部分研究相对较少。文献[7]提出了基数集合最小化策略,在基数最小化策略的"最少修改"和集合最小化策略的"必要修改"两点中取得了平衡;除了函数依赖外,还研究了硬约束。文献[5]提出了一种数据修复算法,该算法采用了增量式的修复策略,同时考虑了函数依赖以及其他约束(硬约束、数量约束、等值约束、非等值约束等)来达到数据一致性的要求。这类算法在一定程度上提高了数据质量,但约束的种类不多,且都未考虑具体领域数据的特点,很难直接用于计算机取证。鉴于此,本文扩充了其他约束并进行了数据修复。

3 相关概念

3.1 函数依赖

函数依赖在关系数据理论中是非常重要且基础的一个概念,在关系模式的设计过程中起到了重要作用。

定义 1(函数依赖,Functional Dependency,FD) 设 $R(U)$是属性集 U 上的关系模式,X 和 Y 是 U 的子集,若对于 $R(U)$的任意一个可能的关系 r,其中不可能存在两个元组 t_1 和 t_2 在 X 上的属性值相等,即 $t_1[X]=t_2[X]$,而在 Y 上的属性值不等,即 $t_1[Y]\neq t_2[Y]$,则称 Y 函数依赖于 X,记作 $X\rightarrow Y$。

3.2 多重约束

函数依赖是描述数据质量的重要方式,但对于很多实例层的约束无法描述。因此,针对取证过程中经常以时间作为主序排查线索的特征,本文定义了其他一些约束。

定义 2(硬约束,Hard Constraint,$HC(C,v)$) 硬约束指定某些元组集合 C 的属性值为 v。例如,通过与银行对账,指定日期入账 1 万元,可以表示为 $HC(t_1[\text{收入}],1)$,$t_1[\text{收入}]$表示某元组在收入属性上的属性值。

定义 3(数量约束,Quantity Constraint,$QC(C,v,q)$) 数据约束指定某些属性 v 列的集合 C 的数量上限为 q。例如,指定达金公司转入的账款一共只有 3 笔,可以表示为 $QC(C[\text{收入来源}],\text{达金},3)$,$C[\text{收入来源}]$表示数据库中"收入来源"这一属性列中的所有值。

定义 4(值约束,Value Constraint,$VC(C)$) 值约束指定某几个单元格的值的集合 C 存在大小关系,包括相等、不等、大于、小于等。例如,指定某两笔资金的来源应该不同,可以表示为 $VC(\{t_1[\text{收入来源}]\neq t_2[\text{收入来源}]\})$;两笔资金的入账时间有先后,可以表示为 $VC(\{t_1[\text{入账时间}]<t_2[\text{入账时间}]\})$。

定义 5(关联约束,Interconnection Constraint,$IC(C1,C2)$) 关联约束也可称为线索约束、证据链约束,是指两个或者多个数据表属性值之间依据某条线索而存在的约束。例如,公司 A 在指定日期将一笔账款转入到 B 公司,那么 B 公司在入账记录中就应该有相应的记录,且记录的"金额"属性值应该相等,"日期"值的差值应该在一周以内,可以表示为 $IC((t_1[\text{汇款金额}]=t_2[\text{入账金额}])$ and $(t_2[\text{入账时间}]-t_1[\text{汇出时间}]<=7\text{ 天}))$(其中 t_1 和 t_2 分别来自不同的关系表)。

3.3 数据库修复及线索发现

给定函数依赖集 Σ 和其他约束集 Ω,只要数据库中的某个实例 I 违背了约束集中的任何一个约束条件,就说明 I 相对于(Σ,Ω)存在不一致。在数据质量研究领域,很多算法被研究用于产生一个不一致实例 I 的修复 I',该修复能同时满足所有约束。本文把这些算法用于计算机取证领域,有两个主要的用途:1)修复被破坏的数据;2)不一致的数据可能正是需要的破案线索,因此在数据库修复过程中并不直接用修复值覆盖原值,而是提供一组或者多组修复建议以及不一致的提示。

根据修复目标的不同,修复算法可以分为集合最小化修复(set-minimal repair)、基数最小化修复(cardinality-minimal repair)和基数集合最小化修复(cardinality-minimal repair)。集合最小化修复是指不存在一个比 I'更小的修复 I'',即 $\Delta(I,I'')\subset\Delta(I,I')$,$\Delta(I,I')$表示在修复过程中,$I'$中值发生变更的所有单元格的集合,且每一个单元格的修改 I'和 I''均一致。

基数最小化修复要求不存在另外一个修复 I''使得修改的单元格的数量比 I'小,即$|\Delta(I,I'')|<|\Delta(I,I')|$。基数集合最小化修复要求不存在另外一个修复 I''使得 $\Delta(I,I'')\subset\Delta(I,I')$。本文考虑到计算机取证过程中不仅需要修复数据,还需要发现线索,因此采用约束最小的基数最小化修复要求。

4 算法

4.1 等价类

等价类(equivalence class)是由文献[2]提出的一个概念,被很多研究所使用,本文给出的算法也沿用了这一关键概念。等价类是指关系表中同一列中值相同的单元格构成的集合。通过等价类,我们可以发现数据是否存在异常。

图 1 是一个等价类的例子,图 1(a)是初始的数据库实例,有 3 个属性(A,B,C),包含 4 个元组,假设有两条函数依赖$\{A\rightarrow B,B\rightarrow C\}$。根据定义可以把数据库实例划分为 8 个等价类(以虚线表示),如图 1(b)所示。根据函数依赖关系 $B\rightarrow C$,可以发现 C 中的等价类$\{2,2\}$和$\{4\}$需要合并,也可以推断出数据存在问题,可能存在破案线索。

	A	B	C
t_1	5	6	2
t_2	3	6	2
t_3	3	6	4
t_4	2	2	1

(a)初始实例

	A	B	C
t_1	5	6	2
t_2	3	6	2
t_3	3	6	4
t_4	2	2	1

(b)等价类集合

	A	B	C
t_1	5	6	2
t_2	3	6	2
t_3	3	6	4
t_4	2	2	1

(c)修正后的集合

图 1　等价类示例

Fig. 1　Examples of equivalence classes

4.2　算法框架

本文的算法不仅要考虑传统的函数依赖,还需要考虑 3.2 节中的多重约束,主要的目的是不仅要修复在数据集成过程或者证据搜集过程中的错误数据,还在于发现异常,找到破案线索。传统的算法设计除了函数依赖外,对其他约束考虑得较少,同时也未考虑数据质量研究在计算机取证领域的应用。为此,我们提出了一种新的算法框架,如算法 1 所示。首先,根据函数依赖集和其他约束集生成等价类集、问题单元格集合、修复单元格集合(包含修复后的值和未确定的变量值),如果不存在问题单元格集合,则程序返回;如果存在问题单元格集合,则程序继续。接着,对初始化阶段不能确定的变量值进行修复,对于修复后的单元格集合,对照函数依赖集合和其他约束集,重新生成问题单元格集合,如果依然存在问题单元格集合,则继续修复,直到不存在问题单元格为止(假设给出的各个约束之间不存在矛盾,算法收敛)。

算法 1　总体框架 main

输入:数据库实例 I,函数依赖集∑和其他约束集 Ω

输出:问题单元格集合 QC,修复后的单元格集合 RC

```
1. 初始化 RC,QC,生成等价类集 E;//初始化阶段,详见 4.3 节
2. 如果 QC≠∅,则 Q=True,否则 Q=False 并返回;
3. QC1=QC;
4. while(Q==True)
5. { while each c∈QC1
6.     { Repair(&RC,∑,Ω);//修复阶段,详见 4.4 节
7.     RC=RC+{c};
8.     }
9.   经过修复后,重新生成 QC1,并生成等价类集 E1;//与初始化阶段类似
10. 如果 QC1=∅,则 Q=False;
11. }
12. return
```

4.3　初始化阶段

算法根据函数依赖和其他约束条件,通过建立等价类集合的方法找到问题单元格。对于不同的约束,处理的方法不完全一样。

1)函数依赖。根据函数依赖描述的属性间的关系,两两合并相关等价类,合并后的等价类中如果发现数值不等,则将单元格加入 *QC*,并统一设置值为两个等价类中值的数量较多的那个等价类所包含的值。

2)硬约束。对于每一个硬约束,检查指定单元格的值,若满足约束,则建立一个等价类;若不满足约束,则在建立等价类的同时,将单元格加入 *QC*,并将修改后的单元格加入 *RC*。

3)数量约束。检查指定属性列中每个等价类的数量,若违反约束,则将单元格加入 *QC*,并将生成时间最晚的几个(超数量约束)单元格中的值设为变量。

4)值约束。检查指定属性列中每个等价类的大小关系,如违反约束,将单元格加入 *QC*,并将两数中的较大值设为变量。

5)关联约束。检查关联表 1 中指定属性列的等价类和关联表 2 中的指定属性列的等价类之间是否满足约束条件,如果违反约束,则将单元格加入 *QC*,并把关联表 2 中的指定属性列的值设为变量。

具体思想如算法 2 所示。

算法 2　初始化 Initialize

输入:数据库实例 I,函数依赖集∑和其他约束集 Ω

输出:问题单元格 QC,修复单元格 RC,等价类集合 E

```
1. E 初始化为空;
2. while each FD∈∑,F:X→Y//处理函数依赖
3.   { 创建 X,Y 的等价类;
4.     两两合并等价类;
5.     如果 c1≠c2,QC={c1,c2}+QC 加入 QC,并统一设置值为两个等价类中值的数量比较多的那个等价类所包含的值,RC={修正后的 c1,c2}+RC;}
6. while HC(C,v)∈Ω//处理硬约束
7.   { 如果 C 中的值不等于 v,则 QC={C}+QC;
8.     C 中所有单元格的值设置为 v,RC={C}+RC;
9.     创建 C 的等价类;}
10. while QC(C,v,q)∈Ω//处理数量约束
11.   { 创建等价类;
12.     如果 v 所在类中单元格的数量大于 q,QC={C}+QC,并将最后几个单元格中的值设为变量 u1,RC={C}+RC;}
13. while VC(C1,C2)∈Ω//处理值约束
14.   { 创建等价类;
15.     如果 C1,C2 中等价类不满足值大小约束,QC={C1,C2}+QC,并将较大值设为变量 u2,RC={C1,C2}+RC;}
16. while IC(C1,C2)∈Ω//处理关联约束
17.   { 创建等价类;
18.     如果来自两个不同表中的等价类不满足约束,QC={C1,C2}+QC,并将 C2 中单元格的值设为变量 u3,RC={C1,C2}+RC;}
19. return(QC,RC,E).
```

4.4　修复阶段

初始化阶段生成了一些变量。关于变量值,针对不同情况采取不同的策略:因数量约束产生的变量,随机设置为在其属性列中没有出现过的一个值;因值约束产生的变量,取满足值约束条件的所属属性列中已出现的一个值;因关联约束产生的变量,在满足约束的条件下,先选择在其属性列中已出现的一个值,如果没有找到这样的值,再随机生成一个没有出现过的值。具体思想如算法 3 所示。

算法 3　Repair(&*RC*,*Σ*,*Ω*)

输入:修复单元格 RC,函数依赖集∑和其他约束集 Ω

输出:修复单元格 RC

```
1. while each 变量 u1 in RC
2.   {u1 随机取在其属性列中没有出现过的一个值;}
3. while each 变量 u2 in RC
4.   {u2 取满足 VC 条件的所属属性列中已出现的一个值;}
5. while each 变量 u3 in RC
6.   { u3 取满足 VC 条件的所属属性列中已出现的一个值或随机生成一个没有出现过的一个值;}
```

4.5 性能分析

本文利用一个真实数据库来进行性能分析。该数据库一次经济纠纷，把供货商和客户的往来账目进行了数据集成，通过寻求集成后的数据表中的异常来找到破案线索，并给出了可能的修复方案。分析前，先对数据库进行一些简化处理；舍弃无关属性列；所有属性均是整数型类型；所有关系模式属性列和元组的数目均相同，一共包含 3 个关系模式，即包含 3 类货品的往来账目，每个关系模式各含 5 个属性，即商品编号(不同类型的货品的编号规则不同)、数量、金额、税率、日期(与基准日期的差值)，元组数大致各为 1000 条。根据上文的描述，可以看出算法的性能主要取决于两个方面：1)约束的数目；2)等价类的个数。表 1 列出了在等价类个数一致的情况下，修改约束的数目会导致须更新的单元格的个数和运行时间发生改变。可以看出，当约束的数目达到 20 以后，须更新的单元格的个数几乎没有变化。表 2 列出了约束数目不变时，更改数据内容，等价类的个数以及运行时间随之发生改变的情况。从实验结果可以看出，约束越多，数据分布越均匀(导致等价类个数增多)，运行时间大多会相应提高，但也有一些特例。例如在表 2 中，1678 个等价类的运行时间比 2546 个等价类的运行时间长，这说明除了上述因素外，运行时间还与变量的取值、约束的内容等因素有关。

表 1 算法运行情况 1

Table 1 Algorithmic operation 1

约束的数目	须更新单元格的个数	运行时间/ms
2	1012	124
4	1022	125
6	2010	220
8	2127	221
10	2145	210
12	2239	264
14	2540	303
16	2654	313
18	3011	356
20	4116	388
22	4116	388
24	4112	385
26	4115	387

表 2 算法运行情况 2

Table 2 Algorithmic operation 2

等价类的个数	运行时间/ms
1238	85
1502	102
1678	112
2546	109
2899	124
3456	215
3510	224
4521	321
4884	345
5010	456

结束语 数据质量一直是数据管理领域的热门话题，本文通过对数据质量的考量来发现数据线索，并根据需要进行数据修复。将数据修复的算法应用于数据取证领域属于首次，以往在发现数据中的异常数据(如经济纠纷中诉讼双方各自提供的电子账目往往存在大量的分歧)的过程中，办案人员的个人经验非常重要，而且需要办案人员花费大量的时间去整理和调查。而本文所提算法可以很好地帮助办案人员，在设定相关约束后，能比较快速地找到异常数据，从而揭开真相，找到提供虚假信息的一方；亦可通过数据修复，还原数据的本来面目(是一组符合约束的正确数据，是数据本来面目的一种可能和假设)；此外，修复的过程中，本文不仅考虑了函数依赖，还考虑了其他外部约束条件，包括硬约束、数量约束、值约束、关联约束等。课题组还将在如何丰富外部约束[8]、增加约束预处理(当约束之间产生矛盾)、提高算法效率等方面进行进一步的探索。

参 考 文 献

[1] PIPINO L, YANG W, RICHARD Y. Data quality assessment [J]. Communication of the ACM, 2002, 45(4): 211-218.

[2] LIANG J S, LI T Y, WANG H X, et al. Research on Data Quality Assessment Algorithm Based on Constraint [J]. Science Technology and Engineering, 2012, 3(12): 551-554. (in Chinese)
梁吉胜，李天阳，王惠霞，等. 基于约束的数据质量评估算法研究[J]. 科学技术与工程，2012，3(12)：551-554.

[3] KOLAHI S, LAKSHMANAN L V S. On approximating optimum repairs for functional dependency violations[C] // Proc of ICDT. Petersburg, Russia: ACM Press, 2009.

[4] CHIANG F, MILLER R J. A unified model for data and constraint repair [C] // Proceedings of the ICDE. Iscataway, NJ: IEEE Computer Society, 2011.

[5] BOHANNON P, FLASTER M, FAN W F, et al. A cost-based model and effective heuristic for repairing constraints by value modicfication[C] // Proceedings of the SIGMOD. New York: ACM Press, 2005: 143-154.

[6] JIN C Q, LIU H P, ZHOU A Y. Functional Dependency and Conditional Constraint Based Data Repari[J]. Journal of Software, 2016, 27(7): 1671-1684. (in Chinese)
金澈清，刘辉平，周傲英. 基于函数依赖与条件约束的数据修复方法[J]. 软件学报，2016，27(7)：1671-1684.

[7] BEKALES G, ILYAS I F, GOLAB L. Sampling the repairs of functional dependency violations under hard constraints[C] // Proceedings of the VLDB. Singapore: VLDB Endowment, 2010: 197-207.

[8] BESKALES G, ILYAS I F, GOLAB L, et al. Sampling from repairs of conditional functional dependency viloations[J]. VLDB Journal, 2014, 23(1): 103-128.