一种基于CFDs规则的修复序列快速判定方法

王 欢[1] 张云峰[2] 张 艳[2]
(北华航天工业学院科学技术处 河北 廊坊 065000)[1]
(北华航天工业学院计算机与遥感信息技术学院 河北 廊坊 065000)[2]

摘 要 数据一致性是大数据质量管理研究的一个重要内容。条件函数依赖(CFDs)是维护数据一致性的有效技术手段。然而,在修复过程中选择不同的CFDs修复顺序,会影响修复的准确性和效率。因此,如何选取一个正确且合理的修复顺序对数据修复至关重要。针对该问题,提出一种基于CFDs规则的快速判定修复序列的计算方法。首先,设计了一种数据修复框架。然后,利用CFDs之间的关联关系,提出了修复序列图的概念,以用于CFDs修复顺序的计算。一方面,可以避免某些错误的或者不必要的数据修复,提高修复的准确性。另一方面,使用规则来判定修复顺序比使用实际数据进行判定更为快速。此外,在判定修复序列的过程中,对修复死锁进行了检测,保证了修复过程的可终止性。最后,通过在真实数据集上与现有方法进行对比实验,证明了所提方法具有更高的准确性和运行效率。

关键词 数据一致性,条件函数依赖,修复序列
中图法分类号 TP311.13 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.03.051

Rapid Decision Method for Repairing Sequence Based on CFDs

WANG Huan[1] ZHANG Yun-feng[2] ZHANG Yan[2]
(Department of Science and Technology, North China Institute of Aerospace Engineering, Langfang, Hebei 065000, China)[1]
(School of Computer and Remote Sensing Information Technology, North China Institute of Aerospace Engineering, Langfang, Hebei 065000, China)[2]

Abstract Data consistency is one central issue of big data quality management research. Conditional functional dependencies (CFDs) are effective techniques for maintaining data consistency. In practice, different repairing sequences may affect precision and efficiency of data repairing. It is critical to select an appropriate repairing sequence. To solve the problem, based on CFDs, this paper presented a rapid decision method for repairing sequence. Firstly, a framework is designed for consistency repairing. Then, by analyzing the association between constraints, the concept of repairing sequence graph is presented to determine repairing sequence on CFDs. It contributes to avoiding some incorrect and unnecessary repairs, which can improve the accuracy of repairing. Meanwhile, repairing sequence with rules runs faster than that with real data. Furthermore, in the process of repairing sequence decision, repairing-deadlock detection is implemented to ensure the termination of repairing. Finally, compared with the existing method, this solution is more accurate and efficient evidenced by the empirical evaluation on two real-life datasets.

Keywords Data consistency, Conditional functional dependencies (CFDs), Repairing sequence

1 引言

数据一致性是大数据质量管理研究的重要事务[1]。因不一致数据引发的数据质量问题每年都会给社会造成巨大的经济损失。德国数据分析机构的调查显示:美国每年因为劣质数据造成的损失高达6000亿美元[2]。对于大数据而言,数据一致性管理将变得更加复杂。因此,从数据一致性着手修复错误数据,提升大数据质量,显得非常重要。

条件函数依赖(Conditional Functional Dependencies, CFDs)是进行数据一致性管理的有效技术手段。Bohannon等人[3]提出了CFDs的概念,其可以对满足一定条件的数据进行一致性检测,进而有效避免语义歧义的产生。针对使用CFDs检测得到的冲突数据,因为修复序列的不同,可能得到不同的修复结果。有的修复序列可能会得到错误的修复结果,影响数据的整体质量;有的修复序列可能会产生一些冗余的修复过程,影响修复的效率。因此,如何选择一个合理的修

到稿日期:2016-12-03 返修日期:2017-04-22 本文受河北省自然科学基金(F2014409008),河北省科技计划项目(17210336),廊坊市科技计划项目(2017011042)资助。

王 欢(1985-),男,硕士,讲师,主要研究方向为数据质量、大数据分析,E-mail:clement82wh@163.com(通信作者);张云峰(1982-),男,硕士,讲师,主要研究方向为数据质量评估、数据挖掘;张 艳(1979-),女,博士,副教授,主要研究方向为大数据、数据一致性修复与规则挖掘。

复序列，使得既能得到正确的修复结果，又能快速地进行修复，成为了一个非常值得研究的问题。

目前，在数据修复中，相关学者对如何计算得到一个合理的修复顺序问题进行了研究。文献[1]提出了一种修复框架，该框架在修复过程中采取一种半人工的方式选择修复的顺序；文献[4]使用一种“修复标识”的结构来记录已经修复过的错误元组，由此确定修复的顺序。这种人工方式会增加计算成本，修复过的元组仍可能存在错误。另外，规则本身是存在顺序关系的，一些规则可能对其他规则产生影响，例如规则 $\phi: X \rightarrow Y$ 和规则 $\phi': Y \rightarrow A$。对于如何选择 ϕ 和 ϕ' 的修复顺序，本文提出一种基于 CFDs 的规则修复序列的判定方法，该方法根据得到的规则序列来指导修复过程。例 1 说明了修复序列对修复结果的影响以及在修复过程中可能出现的问题。

例 1 表 1 列出了 1994 年美国人口普查个人信息的关系实例 I，其关系模式 R 包含姓名(NM)、年份(Year)、婚姻状况(Sta)、性别(Gen)、家庭关系(Rel) 5 个属性，t_{id} 表示元组的序号。在关系实例 I 中，黑体表示与实际情况相冲突的错误数据，括号内为相应的真实值。

表 1 1994 年美国人口普查个人信息的关系实例 I

Table 1 Example I of personal information of American census in 1994

t_{id}	NM	Year	Sta	Gen	Rel
t_1	Joe	1971	single	male	unmarried
t_2	Joe	1972	**single (married)**	male	husband
t_3	Joe	1972	married	male	husband
t_4	Joe	1973	divorce	male	unmarried
t_5	Lee	1972	**single (married)**	male	husband
t_6	Ada	1972	married	female	wife

图 1 给出了关系模式 R 上的 CFDs 规则 $\phi_1 - \phi_3$。ϕ_1：对于 Joe 和 Ada，他们的年份可以唯一决定他们的婚姻状况；ϕ_2：对于一个已婚的人，他的性别可以唯一决定他的婚姻关系；ϕ_3：如果一个人的家庭关系是“husband”或者“wife”，那么他的婚姻状况将是唯一确定的。

ϕ_1: NM, Year → Sta

$(tp\{tp_0(\text{“Joe”}, \_ \| \_), tp_1(\text{“Lee”}, \_ \| \_), tp_2(\text{“Ada”}, \_ \| \_)\})$

ϕ_2: Sta, Gen → Rel $(tp\{tp_0(\text{“married”}, \_ \| \_)\})$

ϕ_3: Rel → Sta $(tp\{tp_0(\text{“husband”} \| \_), tp_1(\text{“wife”} \| \_)\})$

图 1 关系模式 R 上的 CFDs

Fig. 1 CFDs on relational schema R

使用 $\phi_1 - \phi_3$ 对关系实例 I 中的数据进行检测和修复。(t_2, t_3) 是 ϕ_1 检测得到的冲突元组，因为 $t_2[\text{NM}] = t_3[\text{NM}] = \text{“Joe”}$，$t_2[\text{Year}] = t_3[\text{Year}] = \text{“1972”}$，$t_2[\text{Sta}] = \text{Single} \neq t_3[\text{Sta}] = \text{married}$，即 Joe 在 1972 年的婚姻状况里出现了未婚和已婚两种不一致的情况，所以 t_2 和 t_3 是相互冲突的。同理，可以分别用 ϕ_2 和 ϕ_3 检测到 (t_1, t_2) 以及 (t_2, t_3, t_5) 之间的不一致。知识溯源[5]是一种用来确定数据真实值的方法，但是其计算过程非常复杂，并且需要大量的相关数据提供支持。因此，本文使用文献[6]提出的“最小代价”方法进行修复，从概率层面找到最贴近真实值的修复值。

然而，不同的修复顺序可能导致不同的修复结果。对于使用 $\phi_1 - \phi_3$ 检测到的数据错误，如果按照 $\phi_1 \rightarrow \phi_2 \rightarrow \phi_3$ 的顺序进行修复，则首先将 ϕ_1 的冲突数据 $t_2[\text{Sta}]$ 改为“married”，修改之后的结果同时消除了 ϕ_2 的错误，然后再将 ϕ_3 的冲突数据 $t_5[\text{Sta}]$ 改为“married”，修复过程结束；如果按照 $\phi_3 \rightarrow \phi_1 \rightarrow \phi_2$ 的顺序进行修复，则首先将 ϕ_3 的冲突数据 $t_3[\text{Sta}]$ 改为“single”(因为修改 t_3 需要改变一条记录，修改 t_2 和 t_5 需要改变两条记录)，进而修复 ϕ_1 和 ϕ_2 的冲突数据，这样就会得到一个错误的修复结果，影响数据的整体质量；若按照 $\phi_2 \rightarrow \phi_1 \rightarrow \phi_3$ 的顺序，则首先将 ϕ_2 的冲突数据 $t_2[\text{Rel}]$ 改为“unmarried”，然后将 ϕ_1 的冲突数据 $t_2[\text{Sta}]$ 改为“married”，这时会由 ϕ_2 检测得到 (t_2, t_3) 之间的不一致，继而将 $t_2[\text{Rel}]$ 改回“married”，这样就造成了一种修复冗余。这些修复会影响修复结果的准确性和效率。

另外，对于 $t_2[\text{Sta}]$ 中存在的不一致，如果 ϕ_1 将其改为“married”，而 ϕ_3 将其改为“single”，则两种修改结果不能达到一致，会形成一种循环修复的状态而无法终止，从而造成修复死锁。在实际修复的过程中，可以利用规则之间的关联关系来进行修复序列的判定，进而避免修复死锁。

使用规则约束进行数据一致性清洗是当前学术界的一个热门研究话题。文献[7-8]提出了一些使用 CFDs 进行一致性修复的方法，但是目前仍然缺少修复序列的相关判定方法。本文在研究修复序列判定问题的过程中遇到了以下难题：

1)如何计算合理的数据修复序列。不同的修复序列可能会产生不同的修复结果，一次修复可能会对下一次修复产生影响。在这种情况下，得到一个合理的修复序列将变得更加困难。

2)如何避免修复死锁的发生。修复死锁会影响清洗过程的可执行性，而处理修复死锁的关键在于查找死锁发生的位置。

本文通过分析规则和规则的相互关系，提出了一种修复序列的判定方法，具体贡献如下：

1)设计了一种判定修复顺序的迭代方法，通过迭代地进行检测和修复，可以使修复结果和检测更好地结合在一起。

2)提出了一种修复序列图的概念，以把检测到冲突的规则联系在一起，并通过修复序列图计算修复的顺序，避免错误修复和冗余修复。

3)提出了一种检测修复死锁的方法，该方法可以检测出死锁发生的具体位置，并将可能发生死锁的数据放在一起处理，避免修复死锁的发生。

4)在 HOSP 和 Adults 两组真实数据集上进行了实验，结果说明了本文方法的实用性和高效性。

本文第 2 节介绍了相关工作；第 3 节给出了修复序列判定的相关概念；第 4 节详细阐述了提出的修复序列判定的具体方法；第 5 节分析讨论了实验结果；最后总结全文。

2 相关工作

数据质量管理是一个经典问题，已有众多相关研究成果，但仍有广阔的探索空间。本文研究的内容亦属于此领域，并且与本研究密切相关的工作主要包含以下两个方面：

1)数据一致性清洗。数据一致性清洗是指对数据源中的

不一致数据进行检测和修复。规则约束可以在语义的层次上对数据进行分析,在进行数据一致性维护时更加方便、快捷。其中,条件函数依赖 CFDs[3] 是经过理论和实际证明的有效的规则约束。文献[8-10]研究了如何使用规则约束进行一致性修复的相关问题。

2)数据修复框架。数据修复框架描述了数据修复的执行过程,通过执行不同的修复策略可以达到相应的修复目标。文献[1]提出了一种自动清洗框架,该框架可以接受人工参与。文献[11]设计了一种商品信息的数据清洗系统 NADEEF,该系统使用约束规则进行数据清洗,操作灵活且易于扩展。文献[4]提出了一种使用 Fixing Rules 进行数据修复的方法,该方法可以针对数据的某些具体错误进行修复。

此外,针对数据修复,文献[7]提出了一种判定数据修复和规则修复的方法,可以用来判定错误是由数据产生还是由规则产生。本文具体研究了使用 CFDs 进行数据清洗时修复顺序的判定问题,同时避免了修复死锁问题的发生。

3　相关概念

本节针对 CFDs 规则修复序列的判定问题,首先对 CFDs 和修复代价问题进行介绍,然后给出修复序列判定问题的相关概念。

本文主要研究的是关系模型中的数据一致性问题,具体的关系模式为 $R=(t_{id},A_1,\cdots,A_n)$。其中,$t_{id}$ 表示记录编号;$A_1,\cdots,A_n$ 表示 R 上的属性;$dom(A)$ 表示属性 A 的值域;R 上的所有属性集合记作 $attr(R)$;N 表示集合中属性的个数;I 是 R 上的关系实例,n 表示 I 中的元组个数。

3.1　条件函数依赖 CFDs

在函数依赖(Functional Dependencies,FDs)的基础上,文献[3]提出了条件函数依赖 CFDs 的概念。关系模式 R 上的一条 CFD 可以表示为:

$$\phi:X\rightarrow A,tp \tag{1}$$

其中,X 是属性集合,$X\subset attr(R)$,称作规则左部,记作 $LHS(\phi)$,单属性 $A\in attr(R)$,称作规则右部,记作 $RHS(\phi)$,并且 $X\cap A=\emptyset$;$X\rightarrow A$ 是一个标准的函数依赖;tp 是条件模板,规定了 $X\rightarrow A$ 的具体适用情况,tp 中的条件使用具体的属性值来表示,并且使用“‖”将 X 和 A 分开。图 1 给出的就是 CFDs 的具体实例。

CFDs 可以用来检测特定条件下元组之间的一致性关系。对于一个实例 I,t_i 和 t_j 是 I 中的两个元组。一条 CFD ϕ 关于 I 是成立的,当且仅当 $t_i[X]=t_j[X]\in tp$,有 $t_i[A]=t_j[A]$,并且记作 $I\models\phi$,否则,记作 $I\not\models\phi$。Σ 是条件函数依赖的规则集合,对于 $\forall\phi\in\Sigma$,都有 $I\models\phi$,则称 Σ 关于 I 成立,记作 $I\models\Sigma$。

3.2　修复代价模型

修复代价是数据质量清洗过程中的一条基本衡量标准。本文使用文献[6]提出的修复代价计算方法,具体形式如下:

$$cost(t,t')=\sum_{A\in attr(R)}dis(t[A],t'[A]) \tag{2}$$

其中,t' 是 t 的修复目标值;$dis(t[A],t'[A])$ 是一个距离函数,描述了 $t[A]$ 和 $t'[A]$ 的相似程度。以例 1 中的修复过程为例,将 t_2[Sta]改为“married”的代价 $cost(t_2)=1$。

对于实例 I,I' 是 I 经过修改后的目标实例,I 关于 I' 的修改代价为:

$$cost(I,I')=\sum_{t\in I,t'\in I'}cost(t,t') \tag{3}$$

本文将使用文献[6]提出的“最小代价”的修复方法对数据进行修复。

3.3　问题定义

规则之间是存在顺序关系的,本文将使用满足 Armstrong 公理[1] 的规则集合来确保规则中不存在冗余的情况,因此本文只关注如何挑选规则的修复顺序。

最小代价修复是目前最广泛使用的数据修复方法,因此本文研究最小代价修复问题。在使用文献[6]所提方法的条件下,将如何判定一个合理的修复序列使数据修复的整体代价最小的问题简称为修复序列判定问题。该问题的描述如下:给定关系模式 R 上的一个实例 I,以及条件函数依赖集合 Σ,找到一个修复序列 $\vec{v}$,并且 I' 是经过 $\vec{v}$ 修复得到的实例,使 $I'\models\Sigma$ 并且 $cost(I,I')$ 最小,即不存在经过另外一个修复序列 $\vec{v}'$ 得到的实例 I'',使得 $cost(I,I'')<cost(I,I')$。

4　修复序列的判定

本节首先提出一种用于判定修复序列的修复框架,然后提出修复序列图的概念,并使用修复序列图计算修复序列,同时针对修复过程中可能出现的死锁问题进行检测。

4.1　一致性修复框架

为了得到一个合理的修复序列,本文提出一种一致性修复框架,如算法 1 所示。

算法 1　一致性修复框架

输入:关系实例 I,CFDs 集合 Σ

输出:修复序列 $\vec{v}$

```
1.  v⃗ = Ø
2.  WHILE TRUE DO
3.    Σ_V = detect(I, Σ);
4.    IF Σ_V ≠ Ø THEN
5.      Σ_R = seqDec(Σ_V, I);
6.      Σ_S = dlDet(Σ_R, Σ);
7.      Σ_R = Σ_R − Σ_S;
8.    END IF
9.    I = repair(I, Σ_R, Σ_S);
10.   v⃗.add(Σ_R, Σ_S);
11.  ELSE
12.    RETURN v⃗;
13.  END ELSE
14. END WHILE
```

其中,第 3 行的 detect(I,Σ)表示一致性检测方法,返回的是被检测出错误的 CFDs 规则集合 Σ_V。第 5 行的 seqDec(Σ_V,I)用来计算需要优先修复的规则集合 Σ_R。第 6 行的 dlDet(Σ_R,Σ)用来进行死锁检测,返回可能发生死锁的规则集合 Σ_S。这两种方法的具体实现将在 4.2 节和 4.3 节中详细进行介绍。第 9 行的 repair(I,Σ_R,Σ_S)分别对优先修复错误和死锁修复进行处理。第 10 行使用 add()方法将 Σ_R 和 Σ_S 添加到规则序列中。

一次修复过程可能会对后续的检测和修复产生影响,选

择不同的修复策略会造成不同的修复结果，这给整个修复过程增加了许多未知因素，因而无法通过一次检测和修复来实现一致性清洗。算法 1 这种迭代的方法可以计算出当前修复的结果，为下一次检测和修复提供依据。

4.2 修复序列图

为了计算一次迭代中需要优先修复的规则集合，本文提出一种修复序列图的结构。

定义 1(修复序列图) 对于给定的规则集合 Σ，定义其修复序列图为 $G(V,E)$，其中 $V=\Sigma$。对于 $\forall \phi_1,\phi_2 \in \Sigma$，如果 $RHS(\phi_1)\subset LHS(\phi_2)$，则 ϕ_1 和 ϕ_2 之间存在一条由 ϕ_1 指向 ϕ_2 的边 e_{ij}，那么 $E=\{e_{ij}\mid i,j\in[1,|\Sigma|]\}$。

对于例 1 中检测出的错误的 CFDs 规则集合 $\Sigma_V=\{\phi_1,\phi_2,\phi_3\}$，$\Sigma_V$ 的修复序列图如图 2 左部所示。

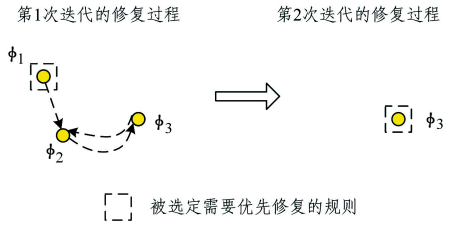


图 2 修复过程示意图

Fig. 2 Schematic diagram of repairing process

从图 2 左部可以看出，ϕ_1 的修复过程会对 ϕ_2 产生影响，ϕ_2 应该在 ϕ_1 修复之后再执行。因此，在一次迭代中，需要找到优先修复的规则，对其检测发现的错误进行处理。

入度为 0 的规则不受其他规则的影响，可以作为优先修复的对象。如果序列图中不存在入度为 0 的点，则序列图中只包含回路。此外，文献[6]证明了最小代价修复问题是一个 NP 完全问题。本文提出了一种启发式方法，该方法选取当前修复代价最小的规则作为需要优先修复的规则，如算法 2 所示。

算法 2 修复序列判定方法

输入：关系实例 I，检测出的错误的 CFDs 集合 Σ_V

输出：需要优先修复的 CFDs 集合 Σ_R

```
1. Σ_R=∅,cost=∞,G_s=graGen(Σ_V);
2. Σ_R=findZeroDeg(G_s);
3. IF Σ_R=∅ THEN
4.   FOR EACH ϕ in Σ_V DO
5.     IF cost_ϕ(I,I')<cost THEN
6.       cost=cost_ϕ(I,I'),Σ_R=ϕ;
7.     END IF
8.   END FOR
9. END IF
10. RETURN Σ_R;
```

例 2 针对例 1 中出现的错误，图 2 描述了使用修复序列图进行修复的过程。第 1 次迭代得到的检测出的错误的 CFDs 集合为 $\Sigma_V=\{\phi_1,\phi_2,\phi_3\}$，选取 ϕ_1 作为优先修复的规则（ϕ_1 的入度为 0），并且将 ϕ_1 的冲突数据 t_2[Sta]改为“married”。第 2 次迭代得到的检测出的错误的 CFDs 集合为 $\Sigma_V=\{\phi_2\}$，选取 ϕ_3 作为需要修复的规则，并将 ϕ_3 的冲突数据 t_5[Sta]改为“married”，修复过程结束，并且得到最终的修复顺序为 $\phi_1\rightarrow\phi_3$。

相关计算复杂度为：graGen(Σ_V)的计算代价为 $O(|\Sigma_V|^2)$，findZeroDeg(G_s)计算代价为 $O(|\Sigma_V|)$，$cost_\phi(I,I')$的计算代价的上限为 $O(|I|)$，实际的修复代价远小于 $|I|$，进而回路规则中的计算代价为 $O(|\Sigma_V|\times|I|)$。因此，修复序列判定方法的计算复杂度为 $O(|\Sigma_V|\times|I|)$。

4.3 修复死锁的检测

一个错误数据可能受到两条不同规则的作用，从而得到不同的修复结果，并且这个结果始终不能同时满足两条规则的要求，造成一种循环修复的状态，这种状态就是修复死锁。

定义 2(修复死锁) 对于条件函数依赖 $\phi:X\rightarrow A,tp$ 和 $\phi':X'\rightarrow A,tp'$，$t$ 是 ϕ 和 ϕ'检测得到的错误元组，即 t 同时违反了 ϕ 和 ϕ'。t'和 t''分别是 ϕ 和 ϕ'作用下 t 的修复目标值，如果 $t'\neq t''$，那么称 ϕ 和 ϕ'是修复死锁的。

修复死锁因循环修复而无法终止，从而导致整个修复过程无法正常进行。为了避免修复死锁的发生，本文提出一种死锁检测算法，如算法 3 所示。

算法 3 修复死锁检测算法

输入：需要修复的 CFDs 集合 Σ_R，CFDs 集合 Σ

输出：可能发生死锁的 CFDs 集合 Σ_S

```
1.  Σ_S=∅,Σ_R'=Σ_R;
2.  FOR EACH ϕ_i in Σ_R' DO
3.    DL=∅;
4.    FOR EACH ϕ_j in Σ_R' DO
5.      IF RHS(ϕ_i)=RHS(ϕ_j) && ϕ_i≠ϕ_j THEN
6.        DL=DL∪ϕ_i∪ϕ_j,Σ_R'=Σ_R'-ϕ_j;
7.      END IF
8.    END FOR
9.    Σ_S.add(DL);
10. END FOR
11. RETURN Σ_S;
```

相关计算复杂度如下：修复死锁检测方法的计算代价为 $\Theta(|\Sigma_R|\times|\Sigma|)$，并且 $|\Sigma_R|$的上界为 $|\Sigma|$。因此，死锁检测方法的计算复杂度为 $O(|\Sigma|^2)$。

进而，对这些可能发生修复死锁的数据进行修复时，需要把这些规则所支配的数据放在一起，赋予统一的目标值来进行修复。

5 实验结果及分析

本文通过在两组真实数据集上进行实验，来说明和验证修复序列判定方法的实用性和高效性。

5.1 实验设置

本文使用 HOSP[1] 和 Adults[2] 两组真实数据集进行实验。HOSP 数据集是美国国民卫生服务中心统计的病人住院期间的医疗信息，包含 17 个属性，超过 20 万条记录。Adults 数据集记录了 1994 年美国国民收入的基本信息，包含 15 个属性和 34481 条记录。

在硬件方面，使用 Intel Core i7-2600 (3.4GHz) CPU，搭

[1] http://www.hospitalcompare.hhs.gov/

[2] http://archive.ics.uci.edu/ml/

载 8GB RAM 的主机;在程序方面,使用 Java 语言进行实现。每组实验均重复 5 次,取 5 次实验结果的平均值进行比较。

使用文献[12]提出的方法,分别从 HOSP 和 Adults 数据中抽取和设计了 190 条和 120 条 CFDs 作为数据清洗时使用的规则。为了方便观察规则序列对清洗效果的影响,使用 3 种方法进行对比:1)CFDs+DL 表示使用 CFDs 进行修复死锁的检测,并随机选取 CFDs 中的规则进行修复;2)CFDs+DL+RS 表示在 CFDs 和修复死锁检测的基础上,使用修复序列图判定规则的修复顺序进行修复;3)Fix 是文献[4]提出的使用 Fixing 规则以及修复标识进行数据修复的方法,该方法对修复过的元组进行标记,并不再进行修复。使用修复准确率来评价修复结果。

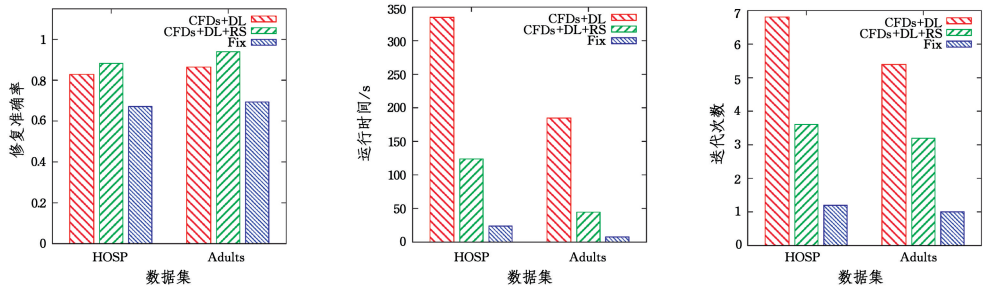
$$\text{修复准确率}=\frac{\text{正确修复的错误数}}{\text{实际错误数}}$$

5.2　实验结果

从总体性能和可扩展性两个方面对规则修复序列的判定方法进行评估。

5.2.1　总体性能

从修复效果、运行时间和迭代次数 3 个方面对规则修复序列的判定方法进行评价,如图 3 所示。从图 3(a)中看出,使用 CFDs 规则进行一致性修复的准确率可以达到 80%以上,说明 CFDs 规则的修复效果较好;Fix 方法因为无法对无法确定的错误进行检测和修复,修复准确率只能达到 70%左右;另外,通过对比 CFDs+DL 和 CFDs+DL+RS 可以看出,在使用修复序列图的情况下修复准确率分别上升了 6%和 8%,说明一个合理的修复序列有助于提高修复的准确性。从图 3(b)和图 3(c)中看出,Fix 使用修复标识只对元组进行一次修复,其迭代次数和运行时间都是最少的;其次,使用规则修复序列图进行规则顺序判定后的情况与不进行规则判定的情况相比,运行时间节省了 50%以上,迭代次数也减少到了原来的 60%左右,这是因为进行修复序列的判定可以有效地避免错误修复和冗余修复,进而提高清洗过程的运行速度。



(a)修复准确率　(b)运行时间　(c)迭代次数

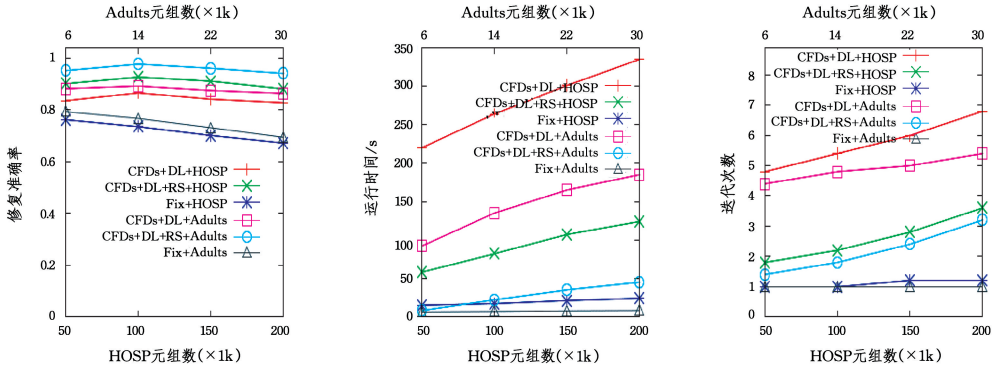
图 3　不同数据集中修复序列判定方法的总体性能

Fig. 3　Overall performance of decision methods for repairing sequences in different data sets

5.2.2　可扩展性

从元组数量和规则数量两方面来评价规则修复序列判定方法的可扩展性。

1)元组数量。图 4 给出了在元组数量变化的情况下规则修复序列判定方法的性能。从图 4(a)中看出,随着元组数量的变化,CFDs+DL 和 CFDs+DL+RS 的修复准确率变化不明显,说明两种方法修复的效果非常稳定,具有很高的可靠性。Fix 的修复准确性会因为无法检测出元组中增加的不确定错误而下降。从图 4(a)和图 4(b)中看出,随着元组数量的增加,CFDs+DL 和 CFDs+DL+RS 运行时间和迭代次数的增加较为平稳,Fix 的迭代次数不受影响,因此运行时间的变化较小。



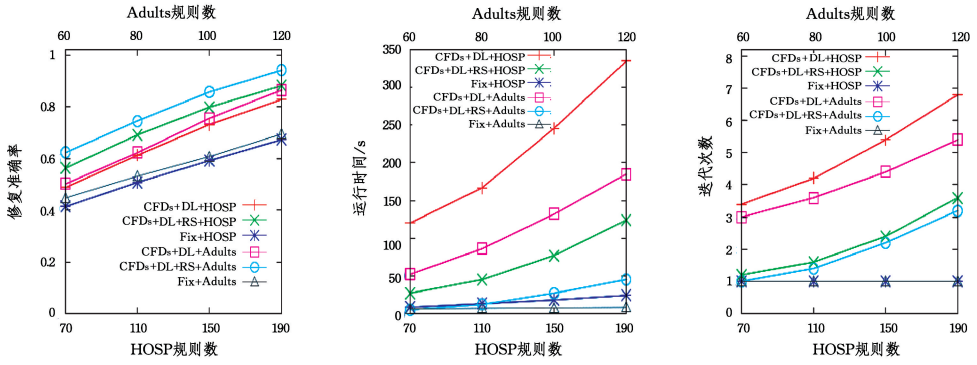
(a)修复准确率　(b)运行时间　(c)迭代次数

图 4　元组数量变化情况下修复序列判定方法的性能

Fig. 4　Performance of repairing sequence decision method under the change of number of tuples

2)规则数量。图 5 给出了规则数量变化情况下规则修复序列判定方法的性能。从图 5(a)中看出,随着规则数量的变化,3 种方法的修复准确率变化比较明显,说明规则对修复效果有着主要作用。因此,设计一组高质量的数据规则,是提升

数据质量的根本途径。从图 5(b)和图 5(c)中看出,随着规则数量的变化,CFDs+DL 和 CFDs+DL+RS 的运行时间和迭代次数的变化都比较平稳;Fix 不会进行重复性的修复,迭代次数和运行时间基本没有变化;此外,使用修复序列图进行修复顺序的判定相比不进行序列判定的情况,运行时间和迭代次数的变化幅度更小,说明进行修复序列判定的方法更稳定。



(a)修复准确率 (b)运行时间 (c)迭代次数

图 5 规则数量变化情况下规则修复方法的性能

Fig. 5 Performance of regular repairing method under the change of rule quantity

以上实验结果表明,本文提出的规则修复序列的判定方法可以有效提升一致性修复的准确性,并能快速完成修复过程。

结束语 CFDs 是一种维护一致性检测和修复的有效技术手段。通过对一致性清洗的详细研究发现,使用 CFDs 进行修复的过程中,修复顺序也是影响修复效果的一个重要因素。针对如何计算一个合理的修复顺序,本文提出了一种基于 CFDs 规则的修复序列判定方法,该方法既保证了修复结果的准确性,又提升了修复的效率。本文提出了一种修复序列图的结构,并设计了一种使用修复序列图判定修复序列的方法。同时,针对修复过程中可能出现的修复死锁问题进行了修复死锁的检测,避免了该问题的发生。最后,通过大量的实验验证了所提方法的实用性和高效性。

参 考 文 献

[1] FAN W, GEERTS F. Foundations of data quality management [M]. Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2012.

[2] ECKERSON W W. Data warehousing special report: data quality and the bottom line[OL]. http://www.adtmag.com/aspx?id=6321.

[3] BOHANNON P, FAN W, GEERTS F, et al. Conditional functional dependencies for data cleaning[C]// Proceedings of the 2007 IEEE International Conference on Data Engineering. 2007: 746-755.

[4] WANG J, TANG N. Towards dependable data repairing with fixing rules[C]// Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data. 2014: 124-136.

[5] HUANG Y, GUERRA-HOLLSTEIN J D, BRUSILOVSKY P. Modeling skill combination patterns for deeper knowledge tracing[C]// Proceedings of the 2016 Personalization Approaches in Learning Environments. 2016: 359-368.

[6] BOHANNON P, FAN W, FLASTER M, et al. A cost-based model and effective heuristic for repairing constraints by value modification[C]// Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data. Baltimore, Maryland, 2005: 143-154.

[7] FEI C, MILLER R J. A unified model for data and constraint repair[C]// Proceedings of the 2014 IEEE International Conference on Data Engineering. 2011: 446-457.

[8] JIN C Q, LIU H P, ZHOU A Y. Functional dependency and conditional constraint based data repair[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.

[9] ZHANG X Y, MENG X F, MA Z M, et al. Attribute weight evaluation approach based on approximate functional dependencies[J]. Computer Science, 2013, 40(2): 172-176. (in Chinese)
张霄雁,孟祥福,马宗民,等. 基于近似函数依赖的关系数据属性权重评估方法[J]. 计算机科学, 2013, 40(2): 172-176.

[10] HAN J Y, CHEN K J. Ranking data quality of web article content by extracting facts[J]. Computer Science, 2014, 41(11): 247-251. (in Chinese)
韩京宇,陈可佳. 基于事实抽取的 Web 文档内容数据质量评估[J]. 计算机科学, 2014, 41(11): 247-251.

[11] EBAID A, ELMAGARMID A, ILYAS I F, et al. NADEEF: a generalized data cleaning system[J]. Proceedings of the 2013 VLDB Endowment, 2013, 6(12): 1218-1221.

[12] FEI C, MILLER R J. Discovering data quality rules[J]. Proceedings of the 2008 VLDB Endowment, 2008, 1(1): 1166-1177.