基于关联数据的一致性和时效性清洗方法

杜岳峰 申德荣 聂铁铮 寇 月 于 戈
(东北大学计算机科学与工程学院 沈阳 110819)

摘 要 数据一致性和数据时效性是大数据质量管理所关注的两个重要内容.条件函数依赖(CFDs)和时效约束(CCs)分别是用于分析数据一致性和数据时效性的有效技术手段.现实生活中的数据会夹杂一些关于一致性和时效性的潜在错误,这些错误又无法为 CFDs 和 CCs 检测和修复,最终影响数据的整体质量.值得一提的是,这些数据通常是相互关联的,这种关联关系可以用来发现数据中的潜在错误.文中使用了一种条件合并的函数依赖(CCFDs)将关联数据放在一起进行处理.基于此,该文提出了一种基于关联数据的一致性和时效性清洗方法.在数据清洗过程中,数据的检测和修复是两个相互影响的过程.所以,该文设计了一种新的自动清洗框架,迭代地进行数据检测和数据修复.其次,该文对关联数据的一致性和时效性清洗的相关问题进行了分析,并且证明了关于 CCFDs 和 CCs 的最小代价修复问题是一个 Σ_2^p 完全(NP^{NP})问题.进而,该文采用一种启发式的修复方法对错误进行修复.为了提高修复的准确性,该文还提出了一种修复序列图的概念.最后,通过在两组真实数据上进行实验,验证了方法的实用性和高效性.

关键词 数据一致性;数据时效性;大数据质量;关联数据;数据清洗
中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2017.00092

A Cleaning Method for Consistency and Currency in Related Data

DU Yue-Feng SHEN De-Rong NIE Tie-Zheng KOU Yue YU Ge
(*School of Computer Science and Engineering, Northeastern University, Shenyang* 110819)

Abstract Data consistency and data currency are critical issues of big data quality management. Conditional functional dependencies (CFDs) and currency constraints (CCs) are two of techniques which analyzes data consistency and data currency. However, data in real world is always mixed with potential inconsistent and non-current errors which cannot be detected by the existing methods, even be intractable to be repaired. It results in low-quality data. Note that, the content expressed by these real-life data are related to each other. And this association contributes to discovering potential errors existing in data. To solve this problem, we employ condition-combined functional dependencies (CCFDs) which put related data together in error detection. In this paper, we propose a cleaning method for consistency and currency in related data. In practice, the detection and the repairing of data cleaning are interactive. A accuracy detection will provide a high-quality basis for repairs. As well the results of the repairs will feed back to the detection. Hence, we design an automatic cleaning framework which detects and repairs data errors iteratively. Futhermore, we discuss the fundamental problems of data cleaning mixed with consistency and currency. We prove that the problem of minimum repairing cost using CCFDs and CCs is Σ_2^p-

收稿日期:2016-04-27;在线出版日期:2016-09-26.本课题得到国家"九七三"重点基础研究发展规划项目基金(2012CB316200,2012CB316201)、国家自然科学基金(61033007,61472070,61672142)、中央高校基本科研业务费专项资金(N150408001-3,N150404013)资助.**杜岳峰**,男,1986 年生,博士研究生,中国计算机学会(CCF)会员,主要研究方向为数据集成、数据质量.E-mail:dr.duyuefeng@gmail.com.**申德荣**,女,1964 年生,博士,教授,博士生导师,主要研究领域为分布式计算、实体识别.**聂铁铮**,男,1980 年生,博士,副教授,主要研究方向为数据集成、数据质量.**寇 月**,女,1980 年生,博士,副教授,主要研究方向为实体识别、网络数据管理.**于 戈**,男,1962 年生,博士,教授,博士生导师,主要研究领域为数据库、大数据管理.

complete (NP^{NP}) so that we propose a heuristic repairing method which computes the minimum-cost target values for repairing the errors in each iterations. Otherwise, to improve the precision of data repairing, we present Repairing Sequences Graph. It calculates the errors which should be repaired preferentially. Our solution is approved more effective and efficient, even evidenced by our empirical evaluation on two real-life datasets.

Keywords　data consistency; data currency; big data quality; related data; data cleaning

1　引　言

数据一致性和数据时效性是大数据质量管理的重要事务[1]. 由数据一致性和数据时效性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. 2006 年,仅信誉卡信息管理一项事务中,由于冲突数据(不一致数据)造成的损失就高达 48 亿美元[2]. 在美国,时效错误数据占企业数据总数的 0.6%～18.5%[2]. 冲突数据和时效错误数据相叠加而产生的错误数据将更为严重,根据德国数据分析机构的调查显示:"美国每年因为劣质数据而造成的损失高达 6000 亿美元"[3]. 所以,从数据一致性和时效性着手修复错误数据,提升数据质量,是一个非常重要的问题.

在处理冲突数据和时效错误数据时,条件函数依赖 CFDs[4](Conditional Functional Dependencies)以及时效约束 CCs[5](Currency Constraints)都是进行数据管理的有效技术. 其中,条件函数依赖可以检测某些特定条件下的冲突数据;时效约束可以对时效错误数据进行检测. 进而,可以将条件函数依赖和时效约束混合使用,解决冲突数据和时效错误数据的叠加问题.

但是,数据中可能会夹杂一些关于一致性和时效性的潜在错误,这些错误无法为条件函数依赖和时效约束所检测并加以处理,进而对数据分析造成影响,导致决策失误. 如何解决数据中存在的潜在错误,就成为一个亟需解决的问题.

现实生活中的事物都是相互关联的. 对大数据而言,其数据的关联关系更加复杂. 并且,这些关联关系有助于解决数据中的潜在错误. 因此,本文的重点就是利用关联数据检测并修复数据中一致性和时序性问题. 例 1 中举例说明了数据中潜在一致性和时效性问题.

例 1.　Joe 和 Ada 是一对夫妇. 图 1 给出了关于实体 Joe(e_1)和 Ada(e_2)的个人信息关系实例,其关系模式 R 包含姓名(NM)、年份(Year)、婚姻状态(Sta)、性别(Gen)和家庭关系(Rel)这 5 个属性. 在实体 e_1 的关系实例中,高亮部分数据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. 现实生活中,实体信息通常来自不同的数据源,并且数据源的置信度存在差异. 所以,数据的修复代价也不尽相同. 本文使用文献[6]中提出的方法计算源数据的置信度作为实体的修复代价权重. 实体 e_1 的数据修复代价权重 wt 为 1 和 1.1,实体 e_2 的数据修复代价权重 wt 为 3.

e_{id}	t_{id}	NM	Year	Sta	Gen	Rel	wt
	t_1	Joe	1971	single	male	unmarried	
e_1	t_2	Joe	1972	single (married)	male	unmarried (husband)	1
	t_3	Joe	1973	single (married)	male	husband	
	t_4	Joe	1974	married	male	husband	
	t_5	Joe	1972	single (married)	male	unmarried (husband)	1.1

e_{id}	t_{id}	NM	Year	Sta	Gen	Rel	wt
e_2	t_6	Ada	1971	single	female	unmarried	3
	t_7	Ada	1972	married	female	wife	

图 1　实体 Joe 和 Ada 的个人信息

CFDs:

ϕ_1: NM, Year → Sta($tp\{tp_0$("Joe", _ ‖ _), tp_1("Ada", _ ‖ _)$\}$)

ϕ_2: Sta, Gen → Rel($tp\{tp_0$("married", _ ‖ _), tp_1("single", _ ‖ _)$\}$)

CCs:

φ_1: $\forall t_1, t_2 (t_1 \prec_{\text{Year}} t_2 \rightarrow t_1 \preceq_{\text{Sta}} t_2)$

φ_2: $\forall t_1, t_2 (t_1[\text{Sta}] = \text{"single"} \wedge t_2[\text{Sta}] = \text{"married"} \rightarrow t_1 \prec_{\text{Sta}} t_2)$

ϕ_1 和 ϕ_2 是 R 上的条件函数依赖 CFDs,tp 为条件模板,φ_1 和 φ_2 是 R 上的时效约束 CCs. ϕ_1 表示,对于 Joe 和 Ada,他们的年份可以唯一决定他们的婚姻状态. ϕ_2 表示,对于一个未婚或已婚的人而言,他的性别可以唯一决定他的家庭关系. φ_1 表示,一个人的婚姻状态会随时间的推移而保持或者发展,但是

不会出现倒退的情况.因而,一条记录的 Year 值越新,其所对应的 Sta 值或者保持不变,或者比其他记录的 Sta 值新.φ_2 表示,如果一条记录的 Sta 值为"single",那么它比其他 Sta 值为"married"的记录要陈旧.在现实生活中,一个人的婚姻状态只能由"single"发展成"married",或者保持"single"的状态,而不可能出现由"married"倒退回"single"的情况.

ϕ_1,ϕ_2,φ_1 和 φ_2 将作为用于检测实体 e_1,e_2 的一致性和时效性的约束规则.在进行一致性检测时,条件函数依赖模板中的每一项 tp_i 将会被当成一条独立的规则,被分别用于数据的一致性检测.其中,ϕ_2 的 tp_0 可以检测出 e_1 中存在的数据冲突:记录 t_1,t_2,t_3,t_5 的 Sta 属性值均为"single",并且它们的 Gen 属性值为"male".然而,t_1,t_2,t_5 的 Rel 属性值"unmarried"与 t_3 的 Rel 属性值"husband"却是相互冲突的.为了修复冲突数据,樊文飞等人[7]提出了一种代价最优的方法.该方法将 t_3 的 Rel 属性值由"husband"修改为"unmarried"(因为修改 t_3 需要改变 1 条记录,而修改 t_1,t_2,t_5 需要改变 3 条记录).修改完后,关系中的所有数据都满足约束规则的要求,结束整个数据修复的过程.但是,一些错误数据却没有被检测出来,比如 t_2,t_3,t_5 中的 Sta 属性值"single"(对于 Joe 和 Ada 夫妇,他们 1972 年的婚姻状态应该保持一致,同为"married"),这些数据将被遗漏并成为潜在错误.

现实生活上,数据之间是相互关联的,并且应该将关联数据放在一起进行检测,而不是分开地、独立地进行.对于 Joe 和 Ada 夫妇,他们之间的数据是相互关联的,可以将 e_1 和 e_2 放在一起进行检测.同时,eCFDs[8] 和 CCFDs[9](Condition Combined Functional Dependencies)验证了 CFDs 的可合并性.本文使用 CCFDs 对规则进行合并,得到的 CCFDs 如下所示:

$$\begin{aligned}\text{CCFDs}\quad & \psi_1:(\text{NM}|\text{Year}\rightarrow\text{Sta},Sc),\\ & Sc\{Sc_0\{\text{"Joe"},\text{"Ada"}\}\}.\end{aligned}$$

ψ_1 表示当把 Joe 和 Ada 的数据放在一起时,数据应该整体满足年份唯一决定婚姻状态的规则要求.

使用 ψ_1,ϕ_2,φ_1 和 φ_2 对实体 e_1,e_2 进行检测时,ψ_1 将 $t_1\sim t_7$ 放在一起检测,进而可以检测到 t_2,t_5 与 t_6 之间发生的冲突,并将 t_2,t_5 的 Sta 属性由"single"修改为"married".然后,φ_1 和 φ_2 可以检测出 t_3 与 t_2,t_5 之间发生的时序错误,将 t_3 的 Sta 属性由"single"修改为"married".接下来,ϕ_2 可以检测出 t_2,t_3,t_4,t_5 之间发生的冲突,将 t_2,t_5 的 Rel 属性由"unmarried"修改为"husband".到此,数据中的所有错误均已被修复.因此,将关联数据放在一起更有利于检测和修复数据中的潜在错误.另外,人工处理潜在数据的代价过大,而且人工处理在分析潜在错误的过程中也极易发生错误.基于此,本文提出了一种自动清洗关联数据中一致性和时序性错误的方法.

尽管,文献[5,10-11]提出了一些关于数据一致性问题和时序性问题的解决方法.但是,目前仍然缺少解决关于关联数据的一致性和时序性问题的相关方法.本文在研究关联数据的修复问题时,遇到了以下难题:

(1)如何计算关联数据的修复代价?由于数据修复过程使用了关联数据,因此需要考虑关联数据对修复代价的影响.

(2)如何计划合理的数据修复顺序?不同的修复顺序会产生不同的修复结果,其中的一些修复顺序可能导致错误的修复结果.对于之前给出的使用 ψ_1,ϕ_2 和 φ_1 的修复方法,如果按照 $\phi_2\rightarrow\psi_1\rightarrow\varphi_1\rightarrow\phi_2$ 的顺序进行检测和修复,就会将 t_3 和 t_4 的 Rel 属性值改为"unmarried"而出现错误.所以,需要选择正确的修复顺序来克服错误的修复.

(3)如何选取合适的修复目标值?从对上面修复顺序问题的描述中,可以看出某一次数据修改可能会对下一次的数据检测和修改造成影响.一方面,对于同一个错误,可能存在多个供选取的目标值;另一方面,某些目标值可能只是修复过程中的一个临时值,需要反复修改几次才能得到最终的修复状态.不论是多个目标值的选取还是临时状态值的选取,都需要选择一个合适的目标值,最终得到正确的修复结果.

本文利用 CCFDs 和 CCs,考虑数据之间的关联关系,提出了一种解决数据一致性和时效性混合问题的数据清洗方法,具体贡献如下:

(1)提出了一种 CCFDs 和 CCs 的混合清洗方法,可以解决数据一致性和时效性的混合问题,并且分析了 CCFDs 和 CCs 混合规则的可满足性问题和蕴含问题.

(2)设计了一种数据修复框架.考虑到数据检测和数据修复的相互作用,本框架迭代地进行执行这两个过程,直到得到一个一致的、时序正确的清洗结果.

(3)提出了一种"修复序列图"的结构.在数据

修复的过程中，错误的修复顺序可能导致错误的修复结果. 修复序列图可以计算出应该优先修复的数据错误，从而避免错误修复.

(4) 证明了关于 CCFDs 和 CCs 的最小代价一致性和时效性清洗问题是一个 Σ_2^p 完全(NP^{NP})问题，进而提出了一种启发式的修复方法.

(5) 通过在 HOSP 数据集和 NBA 数据集上进行实验，说明了本方法的实用性和高效性.

本文第 2 节介绍相关工作；第 3 节给出关于关联数据一致性和时效性清洗的相关概念；第 4 节提出一种一致性和时效性的清洗框架并介绍相应的检测方法和修复方法；第 5 节分析讨论实验结果；第 6 节总结全文工作并对未来工作进行概述.

2　相关工作

数据质量的研究是一个传统问题，现存大量相关文献，但是在该领域中仍然存在一些问题需要探询. 本文研究的问题亦在此领域范畴之内，主要研究的内容是：利用规则约束，解决关联数据中一致性和时效性问题. 目前的研究中，与本文研究密切相关的相关工作主要类别于以下 3 个方面：

(1) 约束规则. 约束规则是数据质量管理的一个重要技术，可以简单、快速地实现数据的检测与修复. 约束规则有很多种类，如函数依赖 FDs[1] (Functional Dependencies)、条件函数依赖 CFDs[4]、否定约束 DCs[12] (Denial Constraints)、时效约束 CCs[5] 等. 通过实践证明这些规则是非常有效的. 在 CFDs 的基础上，eCFDs[8] (extended Conditional Functional Dependencies)和 CCFDs[9] 分析了规则合并的可行性. 本文使用 CCFDs 来检测关联数据的一致性.

对于不同类型的规则，他们既可以独立运行，解决一类数据质量问题，也可以混合使用，同时解决多类数据质量的混合问题. 文献[13]通过将匹配规则 MDs(Matching Dependencies)与 CFDs 相结合，利用主关系中的记录指导从属关系中出现的错误修复. 文献[14]分析讨论了 CFDs 和 CINDs(Conditional Inclusion Dependencies)之间的相互作用关系，进而混合使用 CFDs 和 CINDs 对多数据源中的数据冲突进行检测. 文献[5]通过将 CFDs 和 CCs 结合使用，进而得到关于实体的一致的、最新的记录信息. 与以往的研究有所区别的是，本文关心的是关联数据中的一致性和时效性问题，所以本文使用的是 CCFDs 和 CCs.

(2) 数据一致性分析. 数据一致性是数据质量管理研究的重要事务之一. 解决数据一致性问题可以通过实体识别[15] 等方法来实现. 其中，使用规则约束来验证数据一致性并对其修复是最有效的一致性管理方法之一. 文献[16-17]提出了一致性规则的发现方法. 文献[18]研究了如何使用 CFDs 进行一致性检测和修复的相关问题.

(3) 数据时效性分析. 时序错误的检测和错误数据的修复是时效性分析的两个关键问题. 使用规则约束进行数据时效性分析是目前最有效的方法之一. 文献[5]提出一种，在没有时间戳的条件下，使用时效约束查询数据时效的方法，以此来判定实体的最新状态信息. 文献[19]提出了一种时效图的结构对数据的实效性进行判定. 文献[20]通过结合使用规则和统计的方法对数据进行修复.

此外，对于数据清洗而言，文献[11]提出了的多种修复过程和修复方法，对修复过程的相关问题进行了研究.

本文具体研究了在 CCFDs 和 CCs 混合规则作用下，关于关联数据的一致性和时效性检测和修复问题.

3　相关概念

本节针对关联数据的一致性和时效性清洗问题，首先对条件合并的函数依赖和时效约束进行介绍，然后提出了用于数据清洗的相关概念.

本文使用关系数据的形式对实体进行描述，假定相同实体的数据已经经过识别并集中在一起. 关系模式 $R=(\boldsymbol{e}_{\mathrm{id}},\boldsymbol{t}_{\mathrm{id}},A_1,\cdots,A_n)$，其中，$\boldsymbol{e}_{\mathrm{id}}$ 为实体编号，具有相同实体编号的记录都指向同一实体；$\boldsymbol{t}_{\mathrm{id}}$ 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 表示集合中的属性个数.

3.1　约束规则

(1) 条件合并的函数依赖 CCFDs(Condition-Combined Functional Dependencies)[9]. CCFDs 是一种可以同时对多个条件值进行一致性检测的完整性约束. 关系模式 R 上的一条 CCFD 记作

$$\psi:(C\mid Y\rightarrow A,Sc) \tag{1}$$

其中：C 为条件属性集合；Y 为变量属性集合，C 和 Y 由“|”分隔，并且 $C,Y\subset attr(R)$，$C\cap Y=\varnothing$，C,Y 合称为规则左部，记作 $LHS(\psi)$，单属性 A 称为规则右部，记作 $RHS(\psi)$；$Y\rightarrow A$ 是一个标准函数依

赖；Sc_i是关于C的属性值集合，$Sc_i \subset dom(C)$，Sc是Sc_i的集合，即$Sc=\bigcup Sc_i$. 并且对于任意元组t_i，t_j，如果$t_i[C]$，$t_j[C] \in Sc_i$，t_i和t_j需要放在一起进行检测. ψ_1是CCFDs的一个具体实例. 特别地，ϕ_2也可以表示成：

ψ_2：(Sta|Gen→Rel，Sc){Sc_0{"married"}，Sc_1{"single"}}.

I是R上的一个实例，t_i，t_j是I中的任意两个元组. 一条条件合并的函数依赖ψ关于I是成立的，当且仅当，在$t_i[C]$，$t_j[C] \in Sc_i$的条件下，如果$t_i[Y]=t_j[Y]$，则有$t_i[A]=t_j[A]$，并且记作$I \models \psi$. 否则，记作$I \not\models \psi$. Σ_ψ是条件合并的函数依赖的规则集合，对于$\forall \psi \in \Sigma_\psi$，都有$I \models \psi$，则称$\Sigma_\psi$关于$I$成立，记作$I \models \Sigma_\psi$.

对于CFDs，每一个$tp_i \in tp$都是一条独立的规则ϕ：$(X \to A, tp_i)$，那么ϕ也可以表示成ϕ：$(C|Y \to A, tp_i)$. 对于满足相同形式$C|Y \to A$的CFDs，尽管tp_i不同，它们的$Y \to A$中包含的信息可能存在重叠的部分. 相应地在现实生活中，隶属不同条件的事物，可能是相互关联的，从而共同遵从相同的规则.

需要说明的是：① 规则发现不是本文的主要工作，因而本文使用文献[9]提出方法，将规则合并在一起. 然后由数据专家从中选取重叠部分较高的CCFDs，用于关联数据的一致性分析；② 并不是所有的CFDs都能进行合并，有些CFDs可能存在冲突（比如，ϕ，ϕ'，ϕ''是3条CFDs规则，它们可能存在这样的情况：ϕ可以分别与ϕ'和ϕ''进行合并，但是ϕ'不能与ϕ''进行合并）. 本文采用不相交Sc_i的CCFDψ，即对于$\forall Sc_i, Sc_j \in Sc$，有$Sc_i \cap Sc_j = \varnothing$. 所以，同一CFD规则将不会被重复合并.

(2) 时序关系(Currency Order). I是关系模式R上的实例，I中的关联数据t_i和t_j满足关于属性A的时序关系当且仅当下述条件同时成立：

① $t_i[\boldsymbol{e}_{\mathrm{id}}]=t_j[\boldsymbol{e}_{\mathrm{id}}]$；

② $t_i \preceq_A t_j$.

其中，$t_i[\boldsymbol{e}_{\mathrm{id}}]=t_j[\boldsymbol{e}_{\mathrm{id}}]$表示$t_i$和$t_j$是关于同一实体$\boldsymbol{e}_{\mathrm{id}}$的不同记录. $t_i \preceq_A t_j$表示t_i关于属性A的时效性要高于t_j，即$t_i[A]$相比$t_j[A]$，更能表示$\boldsymbol{e}_{\mathrm{id}}$的最新情况.

如例1中所给出的关于Joe的信息，t_1，t_2分别是Joe在1971年和1972年的Sta状态，那么$t_2[\mathrm{Sta}]$要更新一些，即$t_1 \preceq_{\mathrm{Sta}} t_2$.

时序关系描述了关联数据之间的时间先后顺序，通常使用时序约束来具体描述数据之间的时序关系.

时效约束CCs(Currency Constraints)[5]. CCs是一种可以用来判定数据时效信息的规则约束. t_i，t_j是关系实例I中的任意两个元组，关系模式R上的一条CC记作

$$\varphi: \forall t_i, t_j (\omega \to t_i \preceq_{A_l} t_j) \tag{2}$$

其中，ω是约束的前件，使用断言的合取来进行表示，记作$LHS(\varphi)$. 断言包含以下3种具体形式：① 偏序$t_i \preceq_{A_l} t_j$，在t_i和t_j关于属性A_l的顺序关系不相等的情况下，$t_i \preceq_{A_l} t_j$也可以表示成$t_i \prec_{A_l} t_j$；② 二元谓词$t_i[A_l]\mathrm{op}\ t_i[A_l]$，op可以是=、≠、<、>、≤或≥中的其中一种；③ 常数谓词$t_i[A_l]$ op c或者$t_j[A_l]$ op c，c是一个常数. $t_i \preceq_{A_l} t_j$是规则右部，记作$RHS(\varphi)$.

接下来，基于时效约束CCs，本文提出时序关系断言和时序关系表达式的概念，用来判断实例中元组是否存在时序性上的错误.

定义1. 时序关系断言(Currency Assertion). 对于给定的元组t_i，t_j，它们关于时效约束φ的时序关系断言q_φ定义为

$$q_\varphi: t_i \preceq_{A_r} t_j \tag{3}$$

时序关系断言描述了经过φ推演后，元组t_i，t_j关于属性A_r的时序关系.

定义2. 时序关系表达式(Currency Expression). Σ_φ是时效约束的规则集合，元组t_i和t_j关于Σ_φ的时序关系表达式Q定义为

$$Q = \bigwedge_{\varphi_i \in \Sigma_\varphi} q_{\varphi_i} \tag{4}$$

元组t_i和t_j是时序正确的，当且仅当，他们的时序表达式Q的值为真(True)，即t_i和t_j的任意两个时序关系断言都不互为矛盾式. 换言之，t_i和t_j经过Σ_φ推理得到的所有时序关系断言中，不会同时出现类似$t_i \prec_{A_r} t_j$和$t_i \succ_{A_r} t_j$的情况.

实例I满足时效约束集合Σ_φ是成立的，当且仅当，对于$\forall t_i, t_j \in I$，它们的时序表达式Q的值都为真，并且记作$I \models \Sigma_\varphi$. 否则，记作$I \not\models \Sigma_\varphi$.

例2. 例1中给出的实体e_1的记录和时效规则集合CCs $\Sigma_\varphi=\{\varphi_1, \varphi_2\}$为例，由$t_1[\mathrm{Year}]=$"1971"$\prec t_2[\mathrm{Year}]=$"1972"可得$t_1$，$t_2$关于$\varphi_1$的时序关系断言$q_{\varphi_1}$：$t_1 \preceq_{\mathrm{Sta}} t_2$. 由$t_1[\mathrm{Sta}]=$"single"和$t_2[\mathrm{Sta}]=$"married"可得$t_1$，$t_2$关于$\varphi_2$的时序关系断言$q_{\varphi_2}$：$t_1 \preceq_{\mathrm{Sta}} t_2$. 那么，$t_1$，$t_2$关于$\varphi_1$和$\varphi_2$的时序关系表达式为$Q=q_{\varphi_1} \wedge q_{\varphi_2} = t_1 \preceq_{\mathrm{Sta}} t_2 \wedge t_1 \preceq_{\mathrm{Sta}} t_2 = \mathrm{True}$. 说明$t_1$和$t_2$之间没有发生时序性错误. 对于$e_1$中的所有记录$t_1 \sim t_5$，其中所有的时序表达式$Q$都为真，那么$e_1$中的元组都满足时序约束$\Sigma_\varphi$的要求，即$e_1 \models \Sigma_\varphi$.

用于数据清洗的混合规则集合记作 $\Sigma=\Sigma_\psi\cup\Sigma_\varphi$，它是由 CCFDs 和 CCs 组成的混合规则集合. 如果 $I\models\Sigma_\psi$ 并且 $I\models\Sigma_\varphi$ 同时成立，那么称实例 I 满足混合规则集合 Σ，记作 $I\models\Sigma$.

3.2　修复代价

修复代价是数据质量清洗过程中的一项基本衡量标准. 设 $\Sigma=\Sigma_\psi\cup\Sigma_\varphi$ 是由 CCFDs 和 CCs 组成的混合规则集合，t 是实例 I 中存在错误的元组，对于任意规则 $\eta\in\Sigma$，t' 是 t 关于 η 的修复目标元组，并且 t' 与 t 只在 $RHS(\eta)$ 属性上存在差别，4.3.3 节将讨论如何选取修复目标元组. 接下来，本文给出在 η 作用下，将 t 改为 t' 的修复代价的计算方法.

对于选取的不同类型的规则 η，修复代价计算方法也有所不同. 如果 η 是一条 CCFD 规则 $(C|Y\rightarrow A,Sc)$，在修复的过程中，需要考虑关联数据对数据修复产生的影响. 对于一个条件合并集合 $Sc_i\in Sc$，同一 Sc_i 支配下的元组都是相互关联的.

定义 3.　关联数据集(Related Date Set). 在实例 I 中，元组 t 关于 $|I|^2$ 的关联数据集定义 $I_C(t,\eta)$ 为

$$I_C(t,\eta)=\bigcup_{c_i\in Sc_i}\sigma_{C=c_i\wedge Y=t[Y]\wedge A=t[A]}(I) \tag{5}$$

其中：Sc_i 是 η 中包含 t 的条件合并集合；σ_f 是关系数据库中的选择操作，f 是进行选择的条件. $I_C(t,\eta)$ 选取 Sc_i 中所有与 t 享有相同 Y 属性值和 A 属性值的元组作为 t 的关联数据. 在进行数据修复的过程中，如果关联数据集的一个元组发生变化，集合中的其他元组也要随之变化.

关联数据的选取同样会对修复代价权重造成影响.

定义 4.　在关联数据集 $I_C(t,\eta)$ 中，元组 t 的关联数据的修复代价权重 $wt'(t,\eta)$ 定义为

$$wt'(t,\eta)=wt(t)+\frac{1}{|I_C(t,\eta)|}\sum_{t_i\in I_C(t,\eta)}wt(t_i) \tag{6}$$

其中，$wt(t_i)$ 表示元组 t_i 在实例 I 中的初始的修复代价权重. 由于考虑了关联数据的影响，初始的修复代价权重得到了进一步加强. 式(6)使用关联数据修复权重的平均值来描述元组 t 受关联数据的影响而导致的初始权重的变化情况. 如果与 t 相关的元组 t_i 的修复权重都比较高，那么 t 的修复代价权重的变化就比较大.

例 3.　以 t_1，t_2 和 CCFD ψ_1 为例，其中的 Sc_0 是包含 t_1，t_2 的条件合并集合，t_1 关于 ψ_1 的关联数据集 $I_C(t_1,\psi_1)=\{t_1,t_6\}$，其关联数据的修复代价权重 $wt'(t_1,\psi_1)=1+0.5\times(1+3)=3$；$t_2$ 关于 ψ_1 的关联数据集 $I_C(t_2,\psi_1)=\{t_2,t_5\}$，其关联数据的修复代价权重 $wt'(t_2,\psi_1)=2.05$. 在考虑关联数据的影响的情况下，t_1 和 t_2 的修复代价权重都得到了相应的增强. 并且，关联数据的修复代价权重越大，t_1 和 t_2 的权重变化也越大.

如果 η 是一条 CC 规则，在修复代价的计算过程中，修复代价权重不会受其他元组的影响，可以直接计算求解.

定义 5.　鉴于 CCFDs 和 CCs 对修复代价的影响不同，在 η 的作用下，元组 t 改为 t' 的修复代价定义为

$$cost(t,t',\eta)=\begin{cases}wt'(t,\eta)\cdot Isrepair(t,t'), & \text{如果 } \eta \text{ 是 CCFD}\\ wt(t)\cdot Isrepair(t,t'), & \text{其他}\end{cases} \tag{7}$$

其中，$Isrepair(t,t')$ 是一个判别函数：

$$Isrepair(t,t')=\begin{cases}0, & \text{如果 } t=t'\\ 1, & \text{其他}\end{cases} \tag{8}$$

如果目标值 t' 与初始值 t 不相同，那么将元组 t 改为 t'，并计算其修复代价. 并且，修复代价权重越高，修复的代价也越大.

例 4.　基于例 3 得到的关联修复代价权重，假设 t_2' 是修复的目标值，并且 $t_2'[\text{Sta}]=\text{“married”}$，那么把 t_2 修改为 t_2'，对于单个元组进行修复的代价为 $cost(t_2,t_2',\psi_1)=2.05$. 在考虑到关联数据的情况下虽然增加了修复代价，但是使用关联数据为修改提供参考可以提高修复的准确性.

定义 6.　I' 是实例 I 关于规则集合 Σ 的修复目标实例，对于 $\forall t\in I$，$\forall\eta\in\Sigma$，I' 存储了所有与之相对应的修复目标元组 t'. 那么实例 I 关于规则集合 Σ 的修复代价定义为

$$cost(I,I',\Sigma)=\sum_{t\in I}\sum_{\eta\in\Sigma}cost(t,f(t,\eta,I'),\eta) \tag{9}$$

其中，$f(t,\eta,I')$ 是一个查询函数，利用索引从 I' 中返回元组 t 关于规则 η 的修复目标元组.

3.3　问题定义

在给出约束规则和修复代价的基本定义描述之后，本文需要解决的问题描述如下：

最小代价修复. 给定一个关系模式 R 上的多实体实例 I 以及条件合并的函数依赖 Σ_ψ 和时效约束 Σ_φ 的混合规则集合 $\Sigma=\Sigma_\psi\cup\Sigma_\varphi$，找到一个修复目标实例 I'，使得 $I'\models\Sigma$ 并且 $cost(I,I',\Sigma)$ 最小，即不存在 $I''\models\Sigma$ 并且 $cost(I,I'',\Sigma)<cost(I,I',\Sigma)$. 进而，最小代价修复的判定问题表述为：对于给定的实例 I、规则集合 Σ 及一个正整数 k，是否存在一个修复

目标实例 I'，使得 $I' \models \Sigma$ 并且 $cost(I, I', \Sigma) \leq k$.

定理 1. 最小代价修复的判定问题是一个 Σ_2^p 完全(NP^{NP})问题.

证明. 文献[5]通过“$\exists * \forall *$DNF”问题规约证明了在 CFDs 和 CCs 混合使用的情况下，进行最小代价修复的判定问题是一个 Σ_2^p 完全问题.“$\exists * \forall *$DNF”即 $\tau = \exists X \forall Y \delta$，其中 X 和 Y 的真值集合为 $X = \{x_1, \cdots, x_n\}$ 和 $Y = \{y_1, \cdots, y_m\}$，$\delta = C_1 \vee \cdots \vee C_r$，对于任意 $j \in [1, r]$，$C_j = l_1^j \wedge l_2^j \wedge l_3^j$，并且 $l_{k=1,2,3}^j$ 是一个变量或者 $X \cup Y$ 的补集中的一个元素，在此情况下，对 τ 进行真值判定就是“$\exists * \forall *$DNF”问题. 对于一个由 s 条 CFDs 合并得到的 CCFD ψ，其修复问题可以在 $O(s)$ 内转化成为 CFDs 的修复问题. 同理对于 CCFDs 集合 Σ_ψ，其修复问题可以在 $O(|\Sigma_\psi|)$ 内转化成为 CFDs 的修复问题，其中 $|\Sigma_\psi|$ 表示 Σ_ψ 中包含的 CFDs 规则数量. 并且，CFDs 规则的合并与 CFDs-CCs 的混合使用是相互独立的. 规则合并不会对规则的混合造成影响. 进而，CCFDs-CCs 的规则混合可以通过 CFDs-CCs 的混合来实现. 所以，使用 CCFDs 和 CCs 混合规则的修复问题可以在 $O(|\Sigma_\psi|)$ 等价地转化成为 CFDs 和 CCs 混合规则的修复问题. 那么，在 CCFDs 和 CCs 混合使用的条件下的最小代价修复的判定问题也是一个 Σ_2^p 完全(NP^{NP})问题. 证毕.

4 一致性和时效性混合清洗

对于给定数据实例 I 和用于清洗的约束规则 Σ，为了修复 I 中存在的一致性和时效性的问题，本文提出了一种一致性和时效性混合清洗框架，并讨论了清洗过程中的相关问题.

4.1 清洗框架

在描述清洗框架之前，首先给出异常规则的定义.

定义 7. 给定数据实例 I 及混合规则集合 Σ，$\eta \in \Sigma$，η 是一条异常规则当且仅当 η 至少满足以下其中一个条件：

(1) η 是一条 CCFD ψ 并且 $I \not\models \psi$.

(2) η 是一条 CC φ：$\forall t_i, t_j (\omega \rightarrow t_i \prec_{A_r} t_j)$，如果 $\exists u, v \in I$，$\exists \varphi' \in \Sigma$，使得 u 和 v 关于 φ 和 φ' 的时序关系表达式 $Q = q_\varphi \wedge q_{\varphi'} =$ False.

异常规则表示的是在检测过程中发现数据错误的规则.

接下来，给出一致性和时效性混合清洗框架的基本结构.

如图 2 所示，框架中包含数据检测组件和数据修复组件两个部分. 其中，数据检测组件利用 CCFDs 和 CCs 的混合规则进行检测，返回发现检测错误的异常规则. 数据修复组件通过：(1) 修复序列判别；(2) 错误元组定位；(3) 错误元组修复这 3 个过程对错误数据进行修复. 然后，框架对修复的结果再次进行检测和修复，迭代此过程，直到数据的错误全部被修复并返回清洗后的实例 I'. 算法 1 描述了清洗框架的执行过程.

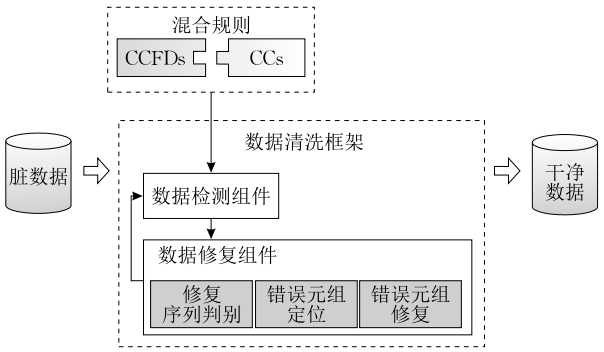


图 2 一致性和时效性混合清洗框架

算法 1. 数据清洗方法 $cc\text{-}Cleaning(I, \Sigma)$.

输入：I, Σ

输出：I'

1. $I' = I$;
2. WHILE TRUE DO
3. $\Sigma' = cc\text{-}Detect(I, \Sigma)$;
4. IF $\Sigma' \neq \varnothing$ THEN
5. $\Sigma'_p = vr\text{-}Find(\Sigma')$,
6. $\Gamma_s = e\text{-}Locate(I, \Sigma'_p)$,
7. $I' = e\text{-}Repair(I, \Gamma_s)$,
8. $I = I'$;
9. END IF
10. ELSE
11. RETURN I';
12. END ELSE
13. END WHILE

其中，$cc\text{-}detect(I, \Sigma)$ 是数据检测组件对应的方法. 数据修复组件 $cc\text{-}repair(I, \Sigma')$ 包含修复序列判别方法 $vr\text{-}Find(\Sigma')$、错误元组定位方法 $e\text{-}Locate(I, \Sigma'_p)$ 和错误元组修复方法 $e\text{-}Repair(I, \Gamma_s)$. 方法的细节将在 4.2 节和 4.3 节中进行介绍.

讨论. 本文将从以下 4 点对数据清洗框架进行讨论：

(1) 数据检测的返回值. 不同于以往的检测过程[11,18]，框架中的数据检测组件返回的是发现检测错误的异常规则，而不是实际发生错误的元组. 因

为，在后续的修复过程中，需要判定哪些是需要优先修复的数据错误，哪些是不需要优先修复的数据. 例 1 中，与 φ_1，φ_2 有关的数据错误应该在 ψ_1 的错误数据解决之后进行. 对于那些不需要优先修复的数据，实际元组的计算会造成一种资源的浪费. 因此，使用异常规则可以减少这些不必要的计算代价.

(2) 检测和修复过程的迭代. 一方面，数据中的潜在错误只能在经过一次修复之后才能被发现. 如例 1 中，t_3 的潜在错误只有在 φ_1，φ_2 修复完后再次检测才能会发现. 所以，需要使用迭代的方法解决数据中的潜在错误. 另一方面，修复过程中可能会产生新的错误，这就需要重新进行一次检测来验证是否产生了新的错误. 文献[1]中提出了一种迭代清洗的修复方法，针对检测得到的错误需要人为参与进行修复，并且由人为判定迭代的终止条件. 区别于文献[1]的清洗过程，本框架使用一种自动迭代的数据清洗方法，在数据修复和程序终止的问题上可以避免某些人为因素产生的修复错误.

(3) 混合规则的可满足性和蕴含性. 这两个问题是规则约束的两个基本问题.

CCFDs 和 CCs 的混合规则可满足性问题可以描述为：给定一个混合规则集合 $\Sigma=\Sigma_\psi\cup\Sigma_\varphi$，是否存在一个完整实例 I，使得 $I\models\Sigma$. 对于一个关系模式 R，$dom(A)$ 表示属性 A 的有限阈值范围，那么 R 上的完整值域空间可以由所有属性的 $dom(A)$ 的组合得到. 完整实例是指包含一个完整值域空间的实例. 规则的可满足性问题研究的是规则之间是否存在冲突的问题. 例如，η，η' 是 Σ 中的两条规则，但是无论如何也找不到一个完整实例 I，使 I 同时满足 η 和 η'. 这说明 η 和 η' 本身是冲突的，那么 Σ 不满足规则可满足性的要求.

CCFDs 和 CCs 的混合规则蕴含性问题可以描述为：给定一个混合规则集合 Σ 和一条规则 η，对于所有实例 I，如果同时有 $I\models\Sigma$ 和 $I\models\eta$，那么称 Σ 蕴含 η，记作 $\Sigma\models\eta$. 并且，不包含蕴含关系的规则集合被成为最小规则集合. 规则蕴含研究的是规则之间的推导关系：一条规则是否可以由其它规则的组合来推导得到. 规则的蕴含性可以用来判别规则集合中是否包含冗余的规则.

文献[5]通过“3-SAT”问题规约证明了 CFDs 和 CCs 混合规则的可满足性问题和蕴含问题分别是一个 NP 完全问题和一个 co-NP 完全问题. CCFDs 是一类特殊的 CFDs，并且 CCFDs 的相关问题可以在 $O(|\Sigma_\phi|)$ 时间内转化成为 CFDs 的相关问题. 所以，CCFDs 和 CCs 混合规则的可满足性问题和蕴含问题同样是 NP 完全问题和 co-NP 完全问题.

本文将使用同时满足可满足性的最小混合规则集合进行清洗.

(4) 程序的终止性. 由于在修复过程中使用的是可满足的最小规则集合，所以框架在迭代修复后最终会达到终止的状态，并且终止的条件是数据中不再检测到新的错误.

4.2　数据检测

对于给定的数据实例 I 和混合规则集合 $\Sigma=\Sigma_\psi\cup\Sigma_\varphi$，针对框架中的数据检测组件，本文提出了一种数据检测方法 *cc-Detect*(I,Σ)，用以返回 Σ 的异常规则集合 Σ'. 算法 2 描述了数据检测的执行过程.

第 3 行中，$I\not\models\eta$ 用来判断 η 是否是一条异常规则. 首先区别规则 η 是一条 CCFD 还是一条 CC，然后判断 η 是否满足 I.

算法 2.　数据检测方法 *cc-Detect*(I,Σ).

输入：I，Σ

输出：Σ'

1. $\Sigma'=\varnothing$；
2. FOR each η in Σ DO
3. IF $I\not\models\eta$ THEN
4. $\Sigma'=\Sigma'\cup\eta$；
5. END IF
6. ENDFOR
7. RETURN Σ'；

复杂度分析. 如果 η 是一条 CCFD，$I\not\models\eta$ 的检测过程可以在 $\Theta(|I|^2)$ 完成，如果 η 是一条 CC，$I\not\models\eta$ 的检测过程可以在 $\Theta(|\Sigma_\varphi|\times|I|^2)$ 完成. 其中，CCFDs 和 CCs 规则数量的上界均为 $|\Sigma|$. 那么，算法 2 的计算复杂度为 $O(|\Sigma|^2\times|I|^2)$.

4.3　数据修复

数据修复组件针对数据检测组件返回的异常规则集合，通过修复序列判别，错误数据定位以及目标值选取 3 个过程来实现数据修复.

4.3.1　修复序列判别

为了计算出异常规则集合 Σ' 中需要优先修复的规则 Σ'_p，本文提出了规则序列图的定义.

定义 8.　对于给定的规则集合 Σ，其规则序列图定义为 $G_s(V,E)$，其中 $V=\Sigma$. 对于任意 η_1，$\eta_2\in\Sigma$，如果 $RHS(\eta_1)\subset LHS(\eta_2)$，则在 η_1，η_2 之间存在一条由 η_1 指向 η_2 的边，记作 e_{ij}，则 $E=\{e_{ij}\mid i,j\in[1,|\Sigma|]\}$.

例 5.　针对例 1 中出现的问题，使用 $\Sigma=\{\psi_1,\psi_2,\varphi_1,\varphi_2\}$ 作为数据清洗的混合规则集合. 第 1 次迭

代的过程中，$\Sigma'=\{\psi_1,\psi_2\}$是得到的异常规则集合. Σ和Σ'的规则序列图如图 3 所示.

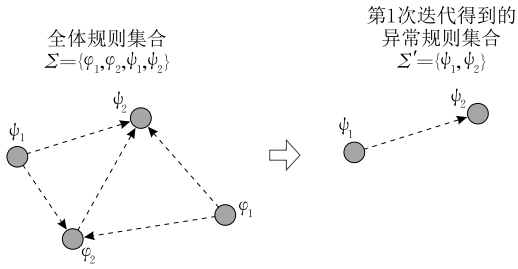


图 3 规则序列图

从图 3 右半部中可以看出，对ψ_2的修复过程会受到ψ_1的影响. 所以，数据修复的过程需要找到需要优先修复的异常规则. 值得一提的是，规则序列图中入度为 0 的规则不会受到其他规则的影响. 所以，修复过程只需要选取这些入度为 0 的规则作为优先修复的规则即可. 对于序列图非空，又不包含入度为 0 的规则的情况，即序列图中只存在回路. 回路情况下求解最小修复代价时，就需要考虑回路中所有修复情况，这些修复会对后续的修复产生影响，进而通过这种动态规划的方法得到所有的修复策略并计算其修复代价. 定理 1 中证明了最小代价修复是一个Σ_2^p完全问题. 所以，本文采用了一种启发式方法，计算回路中每一条规则所覆盖的错误元组修复代价权重总和，并选取总和最小的规则. 算法 3 描述了修复顺序的判别方法的执行过程.

算法 3. 修复序列判别方法 $vr\text{-}Find(\Sigma')$.

输入：Σ'

输出：Σ'_p

1. $\Sigma'_p=\varnothing$, $G_s(V,E)=G\text{-}gen(\Sigma')$;
2. FOR each η in V DO
3. IF degree(η)==0 THEN
4. $\Sigma'_p=\Sigma'_p\cup\eta$;
5. END IF
6. END FOR
7. IF $\Sigma'_p=\varnothing$ THEN
8. $\Sigma'_p=\Sigma'_p\cup minwt(V)$;
9. END IF
10. RETURN Σ'_p;

第 1 行中，$G\text{-}gen(\Sigma')$按照定义的方法生成Σ'的规则序列图. 第 2～6 行计算G_s中入度为 0 的规则. 第 8 行中，$minwt(V)$将从V中选取修复代价权重总和最小的规则.

复杂度分析. 规则序列图的生成过程可以在$\Theta(|\Sigma'|^2)$完成. 计算入度为 0 的规则可以在$\Theta(|\Sigma'|)$完成. 随机挑选异常规则的过程可以在$\Theta(1)$完成. 并且$|\Sigma'|$的上界为$|\Sigma|$. 那么，算法 3 的计算复杂度为$O(|\Sigma|^2)$.

4.3.2 错误元组定位

确定需要优先修复的规则以后，需要找到与规则相对应的具体的错误元组. 错误元组不是单一存在的，他们都是与其它互相冲突而出现的，所以这些发生冲突的元组应该放在一起进行修复. 进而，本文提出错误元组集合的定义.

定义 9. 假设t是实例I存在错误的元组，η是关于t的异常规则，那么元组t关于η的错误元组集合$S(t,\eta)$定义为

$$S(t,\eta)=\{t,s\in I|(t,s)\models\!\!\!\not\;\;\eta\} \tag{10}$$

其中，$(t,s)\models\!\!\!\not\;\;\eta$表示$t,s$关于$\eta$是相互冲突的. $S(t,\eta)$中存储了所有与t相冲突的元组s以及那些与t类似的元组. 这些与t类似的元组虽然不与t发生冲突，但是它们与s发生冲突，所以它们都属于相同类别的错误，应该放在一起. 另外，这些元组的错误元组集合是相同的，所以在实际使用的时候，会将它们合并在一起，只保留一个即可. 最后，将所有的错误数据集合存储在集合Γ_s中.

例 6. 回顾例 1 中的实例，t_1与t_3是关于ψ_2相互冲突的. 那么t_1关于ψ_2的错误数据集合$S(t_1,\psi_2)=\{t_1,t_2,t_3,t_5\}$. 虽然$t_2,t_5$与$t_1$不发生冲突，但是$t_2,t_5$与$t_3$却是冲突的，所以将它们放在一起. 并且对于$t_1,t_2,t_3,t_5$，只保留$S(t_1,\psi_2)$即可.

算法 4 描述了错误元组定位的执行过程.

算法 4. 错误元组定位方法 $e\text{-}Locate(I,\Sigma'_p)$.

输入：I,Σ'_p

输出：Γ_s

1. $\Gamma_s=\varnothing$;
2. FOR each η in Σ'_p DO
3. $I_t=I$;
4. FOR each t_i in I_t DO
5. $S(t_i,\eta)=\varnothing$;
6. FOR each t_j in I_t DO
7. IF $(t_i,t_j)\models\!\!\!\not\;\;\eta$ THEN
8. $S(t_i,\eta)=S(t_i,\eta)\cup t_j$, $I_t=I_t/t_i$;
9. END IF
10. END FOR
11. IF $S(t_i,\eta)\neq\varnothing$ THEN
12. FOR each t_k in I_t DO
13. IF $t_i\sim t_k$ THEN
14. $S(t_i,\eta)=S(t_i,\eta)\cup t_k$, $I_t=I_t/t_k$;
15. END IF
16. END FOR
17. $\Gamma_s=\Gamma_s\cup S(t_i,\eta)$;

18.　END IF
19.　END FOR
20. END FOR
21. RETURN Γ_s;

第 3 行，I_t是一个临时实例，用来控制合并相同的错误元组集合. 第 6～9 行用于找到所有与 t_i发生冲突的元组. 第 13 行中，t_i～t_k用来判断 t_i与 t_k是否相似. 第 12～16 行使用 t_i～t_k找到所有与 t_i类似的元组.

复杂度分析. $(t_i,t_j)\not\models\eta$与 t_i～t_k都可以在 $\Theta(1)$完成. 发现冲突元组和发现类似元组的过程都可以在 $\Theta(|I|^2)$完成. 所以，算法 4 的计算复杂度为 $O(|\Sigma_p'|\times|I|^2)$.

4.3.3　目标值选取

对于数据修复过程而言，修复目标值的选取是一个关键问题. 给定错误元组，选定不同的修复目标值，修复的结果不尽相同，与之对应的修复代价也有所不同. 问题定义部分证明了最小代价修复问题是一个 Σ_2^p完全(NP^{NP})问题. 所以，本文设计了一种启发式方法来选取修复目标值并对错误元组进行修复. 算法 5 描述了目标值选取和错误修复的执行过程.

算法 5.　错误元组修复方法 *e-Repair*(I,Γ_s).

输入：I,Γ_s

输出：I'

1.　$I'=I$;
2.　FOR each $S(t,\eta)$ in Γ_s DO
3.　　$tvalue$=NULL, $count=\infty$;
4.　　FOR each v_i in $dom(RHS(\eta))$ DO
5.　　$cost=0$;
6.　　FOR each t_i in $S(t,\eta)$ DO
7.　　$t_i'=t\text{-}gen(t_i,v_i,RHS(\eta))$,
　　$cost=cost+cost(t_i,t_i',\eta)$;
8.　　END FOR
9.　　IF $cost<count$ THEN
10.　　$tvalue=v_i$, $count=cost$;
11.　　END IF
12.　END FOR
13.　FOR each t_i in $S(t,\eta)$ DO
14.　　$I'=repair(t_i,tvalue,I')$;
15.　END FOR
16. END FOR
17. RETURN I';

第 3 行，$tvalue$ 和 $count$ 分别用来存储最小代价修复的目标值及其代价. 第 7 行，$t\text{-}gen(t_i,v_i,RHS(\eta))$方法将 t_i的 $RHS(\eta)$属性值改为 v_i，以此生成 t_i的目标元组. 第 4～12 行计算关于 $S(t,\eta)$的最小代价的修复目标值 $tvalue$. 第 13～15 行使用 $repair(t_i,tvalue,I')$将 $S(t,\eta)$中的错误元组 t_i修改为 $tvalue$.

复杂度分析. 对于如何选取最小修复代价的目标值，其计算代价为 $\Theta(|\Gamma_s|\times|S(t,\eta)|\times|\pi_{RHS(\eta)}(I)|)$，其中 $|\Gamma_s|\times|S(t,\eta)|$表示所有错误数据的数量，其上限为 $|I|\times|N|$. $|\pi_{RHS(\eta)}(I)|$表示 I关于属性 $RHS(\eta)$的不同值的个数，其上限为 $|I|$. 对于所有错误数据的修复过程可以在 $\Theta(|\Gamma_s|\times|S(t,\eta)|)$完成. 另外，在目标值选取的过程中，$cost(t_i,t_i',\eta)$的计算代价为 $O(|I|)$. 那么，算法 5 的计算复杂度为 $O(|I|^3\times|N|)$. 但是，实际的错误量和不同值的个数分别远小于 $|I|\times|N|$和 $|I|$，并且实验结果可以更好地说明这一点.

5　实　验

本文通过在两组真实数据上进行实验，来验证一致性和时效性混合清洗方法的性能.

5.1　实验设置

硬件方面，本实验使用 Intel Core i7-2600 (3.4 GHz)CPU，搭载 8 GB RAM 的主机. 程序方面，本实验使用 Java 语言进行实现，并且对于每组实验均重复 5 次，取平均值进行比较.

实验数据集. 本文使用 HOSP 和 NBA 两组真实数据，并使用实验中发现的规则对数据中本身存在的错误进行检测.

(1) HOSP①. 该数据集是一个真实数据集，来自于美国国民卫生服务中心，记录了病人在住院期间的医疗统计信息. HOSP 数据集包含 17 个属性和 1.3 K 个实体，并且包含了超过 200 KB 的信息记录.

(2) NBA. 队员的信息数据 Players 和赛季比赛统计信息数据 Stat 是从体育信息数据库网站②上抽取得到真实数据表. NBA 是由 Players 和 Stat 集成得到的数据集. 在集成的过程中，使用 e_{id}作为连接属性，对两个表进行等值连接. 融合后的 NBA 数据集的关系模式为(pid, name, team, league, tname, points, poss, allpoints, min, arena, opened, capacity, state)，涉及 758 个实体，并且包含 18753 条记录.

需要说明的是，HOSP 数据集中本身存在冲突

① http://www.hospitalcompare.hhs.gov/
② http://www.databasebasketball.com/

数据和时序错误的数据，NBA 是由于数据融合的过程而产生的数据的不一致和时序错误. 对错误数据的甄别使用样本分析的方法，从数据中抽取样本采用人工甄别的方法找出数据中的错误. 在具体的甄别过程中采用“封闭世界”原理，即只找出能够被确认是错误的数据，对于无法辨别的数据，且当做真实数据进行处理.

规则集. 本实验首先从 HOSP 和 NBA 数据集上分别抽取高质量的数据样本，然后使用文献[9]中提出的方法计算出所有的一致性候选规则，由领域专家从候选规则中选出适合的规则进行清洗. 时效规则则由领域专家采用人工方法对数据进行分析得到. 对于 HOSP 数据集，选取 120 条支持率大于 0.05 的 CFDs(即每条规则至少可以控制 200 条记录)合并成 19 条 CCFDs，然后与 22 条 CCs 进行混合. 对于 NBA 数据集，选取 190 条支持率大于 0.05 的 CFDs(即每条规则至少可以控制 90 条记录)合并成 27 条 CCFDs，然后与 35 条 CCs 进行混合. 以 NBA 数据集为例，表 1 给出了其中的部分规则.

表 1 NBA 数据集的混合规则范例

混合规则范例
CCFDs
ψ_1:(arena $\mapsto$ state, Sc)
$Sc\{Sc_0\{$ “Conseco Fieldhouse”,“ToyotaCenter”, “American Airlines Center”,“Bankers Life Fieldhouse”$\}$, $Sc_1\{$“United Center”$\}\}$
CCs
$\varphi_1: \forall t_1, t_2(t_1[\text{tname}]=\text{“New Orleans Jazz”} \wedge t_2[\text{tname}]=\text{“Utah Jazz”} \rightarrow t_1 \prec_{\text{tname}} t_2)$
$\varphi_2: \forall t_1, t_2(t_1[\text{allpoints}] \prec t_2[\text{allpoints}] \rightarrow t_1 \preceq_{\text{tname}} t_2)$

对比方法. 为了便于观察分析修复方法的性能，本文使用如下的 4 种方法进行对比实验：(1) CCFD 方法只使用 CCFDs 进行检测和修复；(2) CC 方法只是用 CCs 进行检测和修复；(3) CFD+CC 方法和 (4) CCFD+CC 方法分别使用 CFDs-CCs 和 CCFDs-CCs 的混合规则进行检测和修复. 4 种方法均按照本文提出的框架进行清洗，具体实现时只需要将输入部分的规则约束替换成上述规则即可.

同时，本文使用元组 t 关于属性 A 的属性值 $t[A]$ 作为修复的基础计数单位，称为元子. 在此基础之上，使用准确率、召回率和 F-measure 来评价检测方法的效果，其中检测准确率 $P_d=\dfrac{\text{检测错误数}}{\text{元子总数}}$，检测召回率 $R_d=\dfrac{\text{检测错误数}}{\text{实际错误数}}$，$F_d\text{-measure}=\dfrac{2P_dR_d}{P_d+R_d}$. 对于修复过程，只需要评价方法中的检测错误数换成正确修复数，得到修复准确率 P_r、修复召回率 R_r 以及对应的 F_r-measure.

其中，检测错误数和正确修复数均由每次迭代检测和正确修复的属性值个数求并集得到. 并且，召回率越高说明检测和修复的效果越好.

5.2 实验结果

本文从总体性能、扩展性两个方面对修复方法进行评估.

5.2.1 总体性能

首先，从检测和修复效果、运行时间、迭代情况对修复方法的总体性能进行评价.

检测和修复效果. 表 2 和表 3 分别描述了在 HOSP 和 NBA 数据集上的检测效果和修复效果. 从实验中可以看出：(1) 检测和修复方法的召回率比对应的准确率和 F-measure 高. 准确率由数据中的错误比例所决定，对于 HOSP 和 NBA 数据集，他们的错误率分别在 15% 和 11% 左右，另外，实际检测到和修复正确的错误数还要略少一些，所以得到的准确率较小，对应的 F-measure 也较小；(2) 对于 CCFD+CC，其修复召回率和检测召回率分别在 92% 以上和 88% 以上. 这说明本文提出的方法具有非常显著的清洗效果. 相比 CFD+CC，CCFD+CC 的检测结果和修复结果分别高出 4%～6% 和 5%～6%. 清洗过程中，考虑数据之间的关联关系更有利于发现数据中的潜在错误；(3) 从召回率的角度来看，相比单独使用 CCFD 和 CC，CCFD+CC 可以得到更加准确的清洗结果，其检测结果和修复结果分

表 2 不同数据集上的检测效果

	检测准确率 P_d				检测召回率 R_d				F_d-measure			
	CCFD	CC	CFD+CC	CCFD+CC	CCFD	CC	CFD+CC	CCFD+CC	CCFD	CC	CFD+CC	CCFD+CC
HOSP	0.072	0.059	0.132	0.139	0.482	0.392	0.882	0.925	0.125	0.103	0.230	0.242
NBA	0.058	0.046	0.099	0.106	0.527	0.417	0.901	0.965	0.104	0.083	0.178	0.191

表 3 不同数据集上的修复效果

	修复准确率 P_r				修复召回率 R_r				F_r-measure			
	CCFD	CC	CFD+CC	CCFD+CC	CCFD	CC	CFD+CC	CCFD+CC	CCFD	CC	CFD+CC	CCFD+CC
HOSP	0.066	0.056	0.125	0.132	0.452	0.372	0.831	0.882	0.115	0.086	0.217	0.230
NBA	0.054	0.043	0.095	0.101	0.491	0.389	0.862	0.923	0.097	0.077	0.171	0.181

别高出二者总和的 6%～10%和 3%～4%. 这说明规则之间是相互作用的，并且相互促进.

运行时间. 图 4 描述了 4 种算法中各个过程的运行时间分配情况. 总体上看，CCFD＋CC 的运行时间最高，比 CFD＋CC 高出 7%. 这种开销是由于在计算相关数据过程中，一些数据被重复计算所产生的，并且开销的程度是可以被接受的. 从具体的时间分配上看，错误的检测、定位和目标值的选取这 3 个过程占主要时间开销.

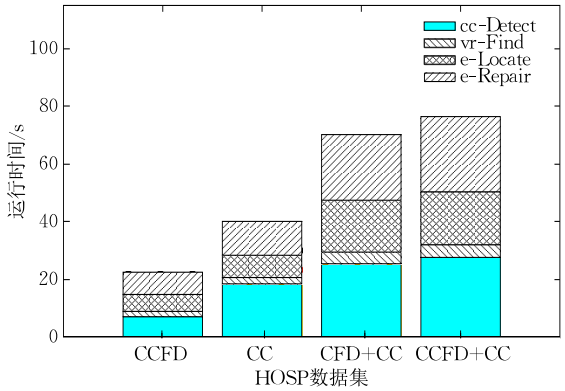


图 4　HOSP 数据集运行时间

迭代情况. 图 5 描述了 HOSP 数据集上的迭代情况. 每一次迭代都会在上一次迭代的基础上进行. 所以，随着迭代次数的增加，检测和修复的效果越来越准确. 对于 HOSP 数据集，从第 5 次迭代开始清洗效果趋于稳定，到第 7 次迭代时，清洗过程终止.

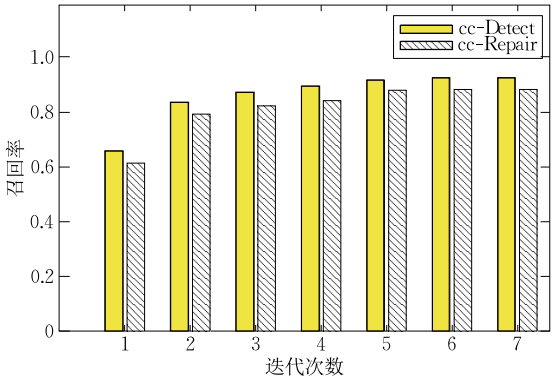


图 5　HOSP 数据集迭代情况

5.2.2　扩展性

接下来，本文在不同规则数量和不同元组数量情况下，研究修复方法的扩展性. 本实验分别改变规则数量和元组数量，观察 HOSP 数据集上修复召回率、运行时间和迭代次数的变化情况.

变化的规则数量. 图 6 描述了在规则数量变化的情况下，清洗方法的性能变化. 横坐标表示混合规则的数量，“＋”的前后分别表示混合规则中 CCFD 的数量和 CC 的数量. 在清洗的过程中，CCFD＋CC 方法使用规则总和进行修复，CCFD 方法和 CC 方法分别使用与之对应的规则进行修复，CFD＋CC 方法则将其中的 CCFD 规则拆分成等价的 CFD 规则进行修复. 从图 6(a)可以看出，修复方法的召回率受规则数量的影响比较明显. 这说明规则集合的数量越多、越完备，其修复的效果越好. 实验中所给出的 19＋22 条规则可以检测出数据中 92%以上的错误，说明这些规则与数据中的错误是最为相关的. 虽然可以通过增加规则数量的方法来提高检测的准确性，但是效果不会非常明显. 所以，制定高质量的清

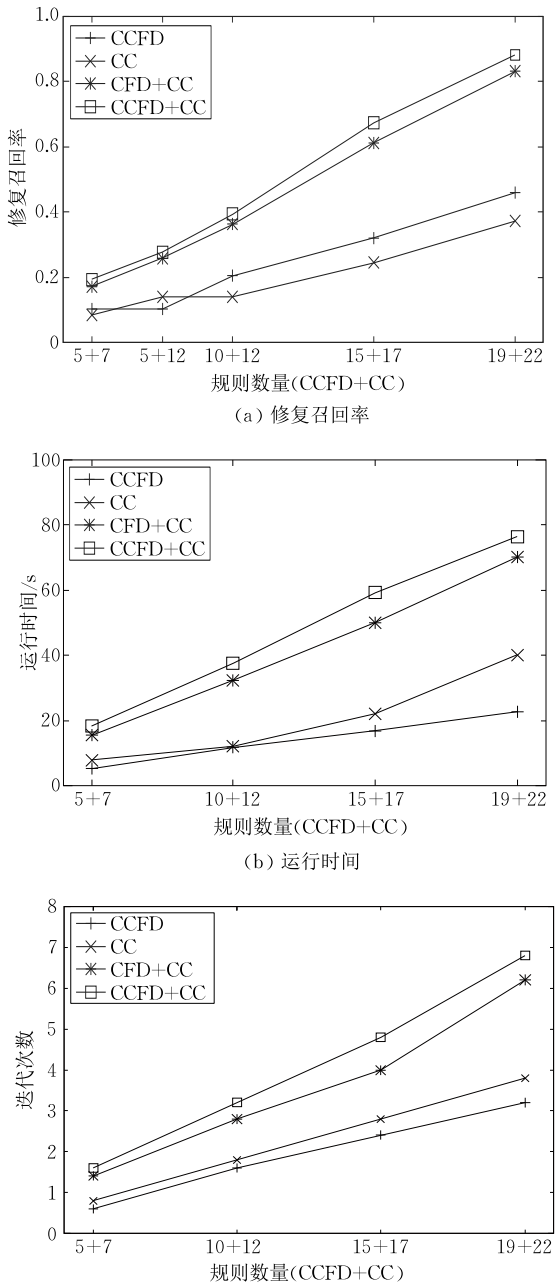
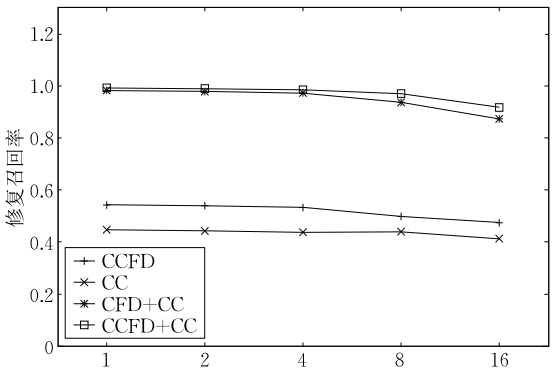


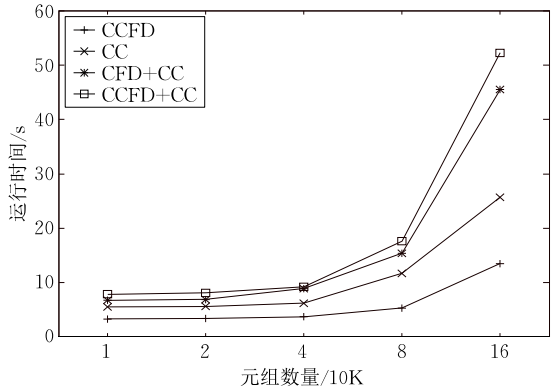
图 6　不同规则数量的清洗结果(CCFD＋CC)

洗规则,对进行数据修复至关重要.从图 6(b)可以看出,CCFD 和 CC 分别呈现正比增长和二次幂数增长,混合后的 CFD+CC 和 CCFD+CC 介于两者之间并大于 CCFD 的增长速度.从图 6(c)可以看出,迭代次数随规则数量增加快速上升.需要说明的是,规则数量的增加可能增加修复序列的长度,进而增加清洗的迭代次数和清洗时间.另一方面,由于每次迭代后都需要重新对修复序列进行计算,所以无法对清洗的迭代次数和运行时间进行估计.在实际操作时,如果运行的时间过长,可以在清洗效果满足用户需求的情况下人为地进行程序终止.

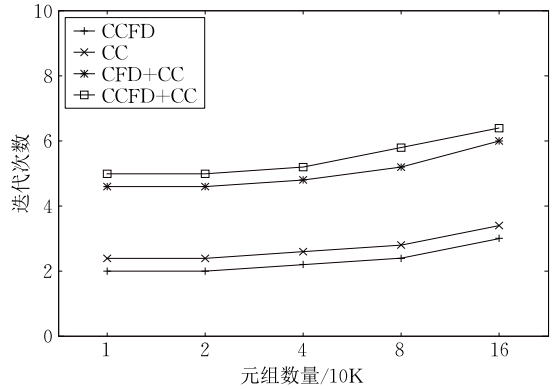
变化的元组数量.图 7 描述了在 HOSP 数据集



(a) 修复召回率



(b) 运行时间



(c) 迭代次数

图 7 不同元组数量的清洗结果

上元组数量的变化对清洗方法的性能影响.从图 7(a)可以看出,元组数量对修复召回率影响较小.新添加的元组中可能会包含一些关联数据,也可能会夹杂一些新的错误数据,这些会对请洗的结果产生波动.从图 7(b)可以看出,4 种修复方法的运行时间呈二次幂数增长.因为在第 4 节中各项算法复杂度分析中,关于元组数量 I,数据检测和错误定位的复杂度均为 $|I|^2$,而目标值选取的复杂度虽然为 $|I|^3$,但实际的错误量会趋近于 $|I|^2$,所以整个方法的运行时间呈二次幂数增长.特别,在处理"大数据"的一致性和实效性的清洗问题时,并且结合图 6 给出的结论,可以通过适当减少迭代次数进行清洗,既可以缩短运行时间,又不会损失很大的清洗精度.对于本文使用的数据集,建议迭代次数缩短到原来的 70% 以上即可.如果采用并行计算,可以将数据按照 80 KB 进行分片.从图 7(c)可以看出,迭代次数随元组数量的增长而缓慢上升.这是因为新增加的元组中可能会夹杂一些新的潜在错误,从而增加了迭代次数.

通过上述的总体性能实验和扩展性实验可以看出,本文提出的方法可以有效地解决数据中一致性和时效性的混合错误,并保持较高的清洗效率.

6 结束语

基于条件合并的函数依赖 CCFDs 和时效约束 CCs,本文提出了一种关于关联数据的一致性和时效性的清洗方法.利用 CCFDs 将关联数据放在一起,用以检测出数据中的潜在错误;同时将 CCFDs 和 CCs 进行混合,用以解决一致性和时效性的混合问题.本文对 CCFDs 和 CCs 混合的基本问题进行分析,并且分别证明了混合规则的可满足性问题和蕴含问题均是 co-NP 完全问题.为了对错误数据进行修复,本文提出了一种新的自动清洗框架.由于清洗中的检测过程和修复过程的相互作用关系,本框架采用一种迭代的方法对数据进行修复.基于本文提出的关联数据的修复代价模型,进一步证明了关于关联数据的一致性和时效性的最小代价修复问题是一个 Σ_2^p 完全(NP^{NP})问题.所以,本文采用了一种启发式方法对数据进行修复.此外,为了得到正确的修复顺序,本文提出了修复序列图的概念,有助于找到需要优先修复的错误数据.最后,通过 HOSP 和 NBA 两组真实数据集上进行实验,证明了本文提出的修复方法的实用性和高效性.

未来我们将继续致力于研究数据检测和数据修复的相关问题. 我们将重点研究：对于检测出的错误，是选择对错误数据进行修复还是对规则进行修复，并且如何修复.

参 考 文 献

[1] Fan W, Geerts F. Foundations of data quality management. Synthesis Lectures on Data Management, 2012, 4(5): 1-217

[2] Marjanovic O, Ariyachandra T, Dinter B. Introduction to organizational issues for big data, business analytics and business intelligence minitrack//Proceedings of the 48th Hawaii International Conference on System Sciences (HICSS). Hawaii, USA, 2015: 4710-4711

[3] Eckerson W. Data warehousing special report: Data quality and the bottom line. Applications Development Trends, 2002, 1(1): 1-9

[4] Bravo L, Fan W, Ma S. Extending dependencies with conditions //Proceedings of the 33rd International Conference on Very large Databases. Vienna, Austria, 2007: 243-254

[5] Fan W, Ma S, Tang N, et al. Conflict resolution with data currency and consistency. Journal of Data and Information Quality, 2014, 5(1-2): 1-38

[6] Dong X L, Saha B, Srivastava D. Less is more: Selecting sources wisely for integration. Proceedings of the VLDB Endowment, 2012, 6(2): 37-48

[7] Bohannon P, Fan W, Flaster M, et al. A cost-based model and effective heuristic for repairing constraints by value modification//Proceedings of the 31st ACM SIGMOD International Conference on Management of Data. Baltimore, USA, 2005: 143-154

[8] Bravo L, Fan W, Geerts F, et al. Increasing the expressivity of conditional functional dependencies without extra complexity// Proceedings of the 24th IEEE International Conference on Data Engineering. Cancún, México, 2008: 516-525

[9] Du Y, Shen D, Nie T, et al. Discovering condition-combined functional dependency rules. Web Technologies and Applications, 2014, 1: 247-258

[10] Fan W, Geerts F, Wijsen J. Determining the currency of data. ACM Transactions on Database Systems, 2012, 37(4): 1-25

[11] Yang Dong-Hua, Li Ning-Ning, Wang Hong-Zhi, et al. The optimization of the big data cleaning based on task merging. Chinese Journal of Computers, 2016, 39(1): 97-108(in Chinese)
(杨东华，李宁宁，王宏志等. 基于任务合并的并行大数据清洗过程优化. 计算机学报，2016，39(1)：97-108)

[12] Chu X, Ilyas I F, Papotti P. Discovering denial constraints. Proceedings of the VLDB Endowment, 2013, 6(13): 1498-509

[12] Fan W, Ma S, Tang N, et al. Interaction between record matching and data repairing. Journal of Data and Information Quality, 2014, 4(4): 1-16

[14] Ma S, Fan W, Bravo L. Extending inclusion dependencies with conditions. Theoretical Computer Science, 2014, 515(1): 64-95

[15] Wang T, Kou Y, Shen D, et al. SIER: An efficient entity resolution mechanism combining SNM and iteration// Proceedings of the 11th Web Information System and Application Conference (WISA). Tianjin, China, 2014: 238-241

[16] Chiang F, Miller R J. Discovering data quality rules. Proceedings of the VLDB Endowment, 2008, 1(1): 1166-1177

[17] Papenbrock T, Ehrlich J, Marten J, et al. Functional dependency discovery: An experimental evaluation of seven algorithms. Proceedings of the VLDB Endowment, 2015, 8(10): 1082-1098

[18] Wang J, Tang N. Towards dependable data repairing with fixing rules//Proceedings of the 40th ACM SIGMOD International Conference on Management of Data. Hangzhou, China, 2014: 457-468

[19] Li Mo-Han, Li Jian-Zhong, Gao Hong. Evaluation of data currency. Chinese Journal of Computers, 2012, 35(11): 2348-2360(in Chinese)
(李默涵，李建中，高宏. 数据时效性判定问题的求解算法. 计算机学报，2012，35(11)：2348-2360)

[20] Li Mo-Han, Li Jian-Zhong. Algorithms for improving data currency. Journal of Computer Research and Development, 2015, 52(9): 1992-2001(in Chinese)
(李默涵，李建中. 数据时效性修复问题的求解算法. 计算机研究与发展，2015，52(9)：1992-2001)

DU Yue-Feng, born in 1986, Ph. D. candidate. His research interests include data quality and data integration.

SHEN De-Rong, born in 1964, Ph. D., professor. Her research interests include entity search and distributed computing.

NIE Tie-Zheng, born in 1980, Ph. D., associate professor. His research interests include data quality and data integration.

KOU Yue, born in 1980, Ph. D., associate professor. Her research interests include entity search and web data management.

YU Ge, born in 1962, Ph. D., professor. His research interests include databases and big-data management.

Background

Data consistency and data currency are central issues of big data quality management. Data cleaning on consistency and currency contributes to improving big data quality. Constraints are widely used for repairing such as conditional functional dependencies (CFDs) and currency constraints (CCs). CFDs are consistency constraints, which catch inconsistencies effectively. CCs identify the currency relationship of instances about the same entity. CCs check out non-current errors. Moreover, cleaning using mixed rules with CFDs and CCs appears more accurate results than using either rules singly. However, data integration may bring new potential errors which are ignored by mixed rules.

In fact, data in real life are content-related, which helps to catch the potential errors. Based on CFDs, content-related conditional functional dependencies (CCFDs) are a new type of consistency constraints. CCFDs detect inconsistencies by putting these content-related data together. Our work is to repair inconsistencies and currency errors in content-related data using CCFDs. We propose a repairing framework which iteratively detects and repairs data errors. Moreover, we prove that the problem of minimum cost repairing with CCFDs and CCs is Σ_2^p-complete. Hence, we adopt a heuristic method for repairing. To avoid some incorrect repairing, we present rules sequence graph. Our solution is approved more effective and efficient, even evidenced by our empirical evaluation on two real-life datasets.

Our work is a partly research of project "The Research of Basic Serviceability Theorem and Critical Techniques in Massive Data" (No. 2012CB316200). This project mainly researches data acquirement, data integration and data analysis in massive data.

This research was supported by the National Basic Research 973 Program of China under Grant Nos. 2012CB316200, 2012CB316201, the National Natural Science Foundation of China under Grant Nos. 61033007, 61472070, 61672142, and the Fundamental Research Funds for the Central Universities under Grant Nos. N150408001-3, N150404013.