数据时效性判定问题的求解算法

李默涵 李建中 高 宏
(哈尔滨工业大学计算机科学与技术学院 哈尔滨 150001)

摘 要 数据的时效性问题是影响数据质量的重要因素之一.时效性差的数据会对企业决策和人们的日常生活带来许多不利影响,这使得判定数据的时效性成为必要.许多应用数据库中都没有完整、清洁、可用的时间戳,从而导致数据时效性的判定非常困难.冗余记录和时效约束能够在时间戳缺失的情况下有效地辅助恢复数据的时序关系,因而能够帮助数据时效性的判定.文中研究包含冗余记录的集合在给定时效约束下的时效性判定问题,并首次提出了时效性判定问题的求解算法.首先,文中定义了查询相关时效性和用户相关时效性.在判定查询相关时效性时,文中将查询归结为最新值查询和时效序列查询两类,并分别根据两类查询的特点,对每类查询定义了查询结果时效性和平均时效性.然后,文中提出了时效图的概念.利用时效图,文中给出了查询相关时效性和用户相关时效性判定问题的求解算法.最后给出了真实数据和虚拟数据上的实验结果,验证了文中算法较高的执行效率,并分析了各个参数对算法的影响.

关键词 数据质量;数据时效性;相关时效性
中图法分类号 TP311 **DOI 号**: 10.3724/SP.J.1016.2012.02348

Evaluation of Data Currency

LI Mo-Han LI Jian-Zhong GAO Hong
(*School of Computer Science and Technology, Harbin Institute of Technology, Harbin* 150001)

Abstract The problem of data currency is one of the most important issues in the area of data quality. The data with poor currency can badly influence the business decision and people's daily life. That highlights the needs of the evaluation of data currency. A big challenge of data currency evaluation is absence of valid timestamps. However, redundant records and currency constraints can recover the currency orders of data without using timestamps thus can be helpful when evaluating data currency. This paper investigates the methods of currency evaluation with redundant records and currency constraints. First, this paper defines data currency relative to queries and data currency relative to users. When evaluating data currency relative to queries, all the queries are classified as 2 categories, which are Current Value Query and Currency Sequence Query. For each query category, this paper discusses the definition of the currency of query result and the average currency of the entire query category. Second, the definition of currency graph is proposed in this paper. The methods of evaluating data currency relative to queries and users using currency graphs are presents. Experimental results on real and synthetic datasets are given to analyze the effect of parameters and the efficiency of algorithms.

Keywords data quality; data currency; relative data currency

收稿日期:2012-06-30;最终修改稿收到日期:2012-08-25.本课题得到国家"九七三"重点基础研究发展规划项目基金(2012CB316202)资助.**李默涵**,女,1987 年生,博士研究生,主要研究方向为数据质量.E-mail: limohan.hit@gmail.com.**李建中**,男,1950 年生,教授,博士生导师,主要研究领域为数据库、海量数据处理、物联网和无线传感器网络等.**高 宏**,女,1966 年生,教授,博士生导师,主要研究领域为无线传感器网络、物联网、海量数据管理和数据挖掘等.

1 引　言

数据的时效性问题普遍存在于各类实际应用中，是影响数据质量的重要因素之一. 随着时间的推移，数据质量会快速的下降，有统计称，在商业数据库中，约有 2% 的客户信息会在一个月内变的陈旧[1]，即是说，在 2 年内，会有近 50% 的记录因为过时而使其可用性受到影响. 在企业决策时，企业往往会因为使用了陈旧的数据而做出错误的决策，而在日常生活中，银行可能会将信用卡账单寄送到持有人搬家前的旧地址. 劣质数据每年给美国造成的经济损失高达 6000 亿美元[1]，而时效性差正是造成数据劣质的罪魁祸首之一. 为此，确保数据时效性是一个十分重要的问题.

数据时效性判定问题是确保数据时效性的一个关键问题. 无论是数据时效性问题的发现还是数据时效性问题的修复，都首先需要对数据时效性加以判定. 在用户使用数据时，往往也首先要求判定数据的时效性. 本文集中研究数据时效性判定问题.

通常，很多应用数据库中都没有时间戳. 即使存在时间戳，由于数据缺乏及时有效的维护或数据集成等原因，这些时间戳经常不可用或不精确[2]. 而且，同一记录的不同属性随时间变化的频率不同，使得同一记录的不同属性的新旧程度也会不同，为每条记录的每个属性都保存相应的时间戳并非易事. 所以，数据时效性判定问题面临的主要挑战是数据集合中没有完整、精确、可用的时间戳.

虽然在时间戳缺失或不精确的情况下，数据时效性的判定问题变得困难，但是我们仍能够利用一些冗余记录和时效约束(currency constraint)来实现数据时效性的判定. 冗余记录是指数据集合中存在着多条记录描述同一实体的情况. 例如，在网络环境下，用户搜索某人的信息，往往得到多条不同的结果；数据集成环境下，会有多条来自不同数据源的记录描述同一实体；企业周期性地备份数据库，在被备份的历史数据中，同一实体也可能存在多条历史记录. 时效约束是数据的语义信息，可以辅助恢复同一实体不同属性值的部分时序关系[3]，进而判定数据集合的时效性. 我们用例 1 来说明如何结合冗余记录和时效约束来恢复时序关系.

例 1.　考虑图 1 所示的数据集合 D. D 中有 4 条记录描述实体 Alice 在过去某段时间的工作情况，其中，tID、EID、FN、LN、$City$、$Salary$ 和 $Status$ 分别表示记录 ID、实体 ID、姓、名、工作城市、工资和婚姻状况.

tID	EID	FN	LN	$City$	$Salary$	$Status$
t_1	1	Alice	Smith	Beijing	50 k	Single
t_2	1	Alice	Smith	Shanghai	70 k	Single
t_3	1	Alice	Green	Guangzhou	80 k	Married
t_4	1	Alice	Green	Harbin	80 k	Married

图 1　Alice 过去几年的工作情况

假设 D 中无可用时间戳可以表示数据的生成时间及插入时间，但有时效约束如下：

cc_1：同一个人的工资不会随时间下降.

cc_2：存有工资最新值的记录也存有工作城市的最新值.

cc_3：同一个人的婚姻状况只能由 Single 变成 Married，再变成 Divorced.

我们用 $t_i \prec_A t_j$ 表示属性值 $t_i[A]$ 比 $t_j[A]$ 旧，将 t_i 和 t_j 的 A 属性值新旧程度相同记为 $t_i =_A t_j$[3].

由约束 cc_1 可知，因为 Alice 的 $Salary$ 取值从小到大为 $50\text{k} < 70\text{k} < 80\text{k} = 80\text{k}$，所以 $t_1 \prec_{Salary} t_2 \prec_{Salary} t_3 =_{Salary} t_4$. 同理，由 cc_3 可得 $t_1 =_{Status} t_2 \prec_{Status} t_3 =_{Status} t_4$.

结合 cc_1 和 cc_2，在 $City$ 属性上有 $t_1 \prec_{City} t_2 \prec_{City} t_3$ 和 $t_1 \prec_{City} t_2 \prec_{City} t_4$. 注意，在 $Salary$ 属性上，由 $t_3[Salary]=t_4[Salary]$ 可得 $t_3 =_{Salary} t_4$，但同样的判断在 $City$ 属性上并不成立，因为 $t_3[City] \neq t_4[City]$. 在这种情况下，我们只能恢复出 $City$ 属性上的部分时序关系.

如例 1 所示，当数据集合中不存在可用时间戳，但存在冗余记录和时效约束时，我们能够有效地恢复该数据集合上的部分时序关系，并判定数据集合在给定的时效约束下的时效性. 文献[3]是当前数据质量领域中唯一一篇研究数据时效性的工作，该文献利用由冗余记录和时效约束恢复得到的时序关系对数据的时效性进行了建模，但遗憾的是，文献[3]并没有给出任何能判定数据时效性的算法.

本文在文献[3]的基础上，研究包含冗余记录的数据集合在给定的时效约束下的时效性判定问题，主要解决了数据相对于查询的时效性判定问题、数据相对于用户的时效性判定问题.

我们把数据相对于查询的时效性称为查询相关时效性. 当用户向时效性不佳的数据发出查询时，查询结果的时效性也不能保证，所以需要将查询结果及其时效性判定结果一同返回给用户. 为了求解此问题，我们将查询归结为 2 类：最新值查询和时效序

列查询.对每类查询,分两步求解数据相对于该类查询的时效性判定问题:第1步,对于给定的数据集合D和查询Q,判定Q在D上的查询结果的时效性;第2步,基于第1步的判定方法,将每类查询作为一个整体,判定数据集合D相对于该类所有查询的平均时效性.

我们将数据相对于用户的时效性称为用户相关时效性.面对同一个数据集合,不同的用户会有不同的需求,这些需求可以通过查询来体现.例如,企业中薪酬分析人员的查询集中于"职位"和"工资"属性,而销售经理则更倾向于查询员工在不同城市的销售业绩.对于特定用户,其查询往往集中于某一类,甚至某几个查询,那么只要保证数据相对于这些常用查询的时效性良好,数据对于用户来说,时效性就高.因此,数据相对于用户的时效性判定问题可以归结为常用查询的查询结果的时效性判定问题.我们给出了此类判定问题的求解方法.

本文贡献如下:

(1)研究了包含冗余记录的数据集合在给定时效约束下的时效性判定问题,并给出了查询相关时效性和用户相关时效性的定义.

(2)将查询归结为2类,提出了时效图的概念,给出了数据相对于查询的时效性判定问题的求解算法.

(3)给出了数据相对于用户时效性的判定方法.

(4)通过大量实验,研究了各个参数对时效性判定结果及算法效率的影响.

本文第2节介绍相关工作;第3节给出查询相关时效性和用户相关时效性的定义;第4节提出查询相关时效性和用户相关时效性的判定方法;第5节讨论实验结果;第6节给出本文结论及未来研究方向.

2 相关工作

文献[3]首次研究了数据库的时效性建模问题,该文献分3部分对数据的时效性建模:时序关系、时效约束、不同数据源间的拷贝函数.时序关系$t_i \prec_A t_j$的定义与例1中相同.时效约束定义为一阶逻辑语句,能够描述数据的语义信息.当数据来自于多个数据源时,拷贝函数可以描述不同数据源间的依赖关系(如数据源D_1的某些数据可能拷贝自数据源D_2).

本文继承了文献[3]中时序关系和时效约束的定义,但研究的内容有所不同.首先,本文研究数据的查询相关时效性和用户相关时效性判定问题,文献[3]没有研究这两个问题.其次,本文给出了查询相关时效性和用户相关时效性的判定算法,而文献[3]没有给出任何数据时效性判定的算法.第三,文献[3]研究的判定问题多为NP完全或更难,这给该模型应用于实际带来了不小的困难,而本文对时效约束进行了更具体的分析,使得在本文定义的场景下,能够在多项式时间内判定数据的时效性.

真值发现[4-6]研究如何发现多个数据源之间的依赖,即前文所述的拷贝函数.这些工作能够在多数据源环境下,根据数据源的依赖关系发现过时数据,但因为其目标在于对不一致数据发现真值,并不研究数据时效性,所以与本文工作不同.

时间数据库[7-8]研究数据库对时间维度的支持.这些工作与本文工作有三点不同:(1)时间数据库假设数据有完整、清洁、可用的时间戳,而本文不假设数据有时间戳;(2)时间数据库侧重于研究如何让数据库对时间有更好的支持,并不考虑数据时效性能否满足用户需求;(3)时间数据库假定同一记录的所有属性时效性相同,而本文没有这一假定.

时间约束(temporal constraints)研究[9]看似与时效性研究相关,实际不同.时间约束研究主要集中在约束与满足约束的变量之间的关系,与本文研究的时效性判定问题不同.时间约束的定义也与本文研究的时效约束不同.

3 数据时效性定义

3.1 预备知识

本文假设数据集合中可以有多条记录描述同一实体,且实体识别工作[10]已经完成.令数据模式$R=(tID, EID, A_1, \cdots, A_n)$,其中,$tID$表示记录ID;$EID$表示实体ID,如果$t_i[EID]=t_j[EID]$,则$t_i$和$t_j$描述同一实体;$A_1, \cdots, A_n$是$n$个属性,$t_i[A]$表示记录$t_i$的$A$属性值.

当描述同一实体的记录在同一属性上有多种取值时,时序关系可以描述各值的新旧情况,形式化定义如下.

定义1. 时序关系(Currency Order).设D是数据模式R的数据集合实例.D中记录t_i和t_j满足属性A上时序关系$t_i \prec_A t_j$当且仅当下述条件同时成立:

① $t_i[EID]=t_j[EID]$;

② $t_i[A]$先于 $t_j[A]$出现在 D 中.

例如，在图 1 所示的数据集合中，如果 Alice 2009 年的 $City$ 值为 Beijing，2010 年的 $City$ 值为 Shanghai，则 $t_1 \prec_{City} t_2$.

时序关系是一个传递关系. 记录间的时序关系需要通过时效约束来判定. 时效约束的形式化定义如下.

定义 2.　时效约束(Currency Constraints). 设 A_i 是模式 R 的属性. R 在 A_i 上的时效约束定义为如下一阶逻辑语句：

$$\forall t_1,\cdots,t_k:$$
$$R(\bigwedge_{j\in[1,k]}(t_1[EID]=t_j[EID])\wedge\psi\rightarrow t_u\prec_{A_i}t_v),$$

其中，R 是数据模式；$t_1,\cdots,t_k$ 是模式 R 的任意实例的 k 条记录；ψ 是谓词；$\bigwedge_{j\in[1,k]}(t_1[EID]=t_j[EID])\wedge\psi$ 称为时效约束前件；$t_u\prec_{A_i}t_v$ 称为时效约束后件.

例 1 中的 cc_1、cc_2 可以表示为式(1)、(2)所示的约束，cc_3 可写成(3)、(4)两条约束.

$$\forall s,t: Emp(t[EID]=s[EID]\wedge t[Salary]<s[Salary]\rightarrow t\prec_{Salary}s) \quad (1)$$

$$\forall s,t: Emp(t[EID]=s[EID]\wedge t\prec_{Salary}s\rightarrow t\prec_{City}s) \quad (2)$$

$$\forall s,t: Emp(t[EID]=s[EID]\wedge(t[Status]=Single\wedge s[Status]=Married)\rightarrow t\prec_{Status}s) \quad (3)$$

$$\forall s,t: Emp(t[EID]=s[EID]\wedge(t[Status]=Married\wedge s[Status]=Divorced)\rightarrow t\prec_{Status}s) \quad (4)$$

3.2　查询相关时效性

根据 R 在 A_i 上的时效约束能够得到描述某实体的记录在 A_i 上的时序关系，因此使用时效约束时只需知道待判断的实体和属性分别是什么即可. 设查询为 Q，那么基于时效约束判定查询相关时效性时就需要把 Q 抽象成三元组 $Q=(e_Q,Attr(Q),QType)$，其中，e_Q 表示被查询的实体；$Attr(Q)$ 表示被查询的属性集合；$QType$ 表示 Q 所属的查询类别. 查询类别有最新值查询和时效序列查询 2 种，用 cur_Q 表示 Q 的查询结果的时效性，cur_Q 的值域为[0,1]，其值越接近 1 则查询结果的时效性越好，越接近 0 则查询结果的时效性越差，两种查询相对应的查询相关时效性分别定义如下.

3.2.1　最新值查询

最新值查询(Current Value Query，CVQ)只关心某实体的最新状态，即 e_Q 的 $Attr(Q)$ 中属性的“最新值”. 典型的 CVQ 形如查询 Q_1：“Alice 当前的工作城市和工资”，即 $Q_1=$(Alice，{$City$，$Salary$}，CVQ).

为了判定 CVQ 查询相关时效性，首先要对给定的数据集合 D 和查询 Q，判定 Q 在 D 上的查询结果的时效性. 设 $Q=(e_Q,Attr(Q),CVQ)$，则为了回答 Q，需要给 e_Q 的 $Attr(Q)$ 中每个属性找到其唯一确定的最新值. 如果这样的最新值不能被找到，那么 Q 就无法被准确地回答，其查询结果的时效性自然也受到影响. 因此在判定 CVQ 查询结果的时效性时，首先需要为 $Attr(Q)$ 的属性 A_i 找到一个记录集合 T_{A_i}，使得对任意记录 $t\in T_{A_i}$，$t[A_i]$ 是 e_Q 的一个可能的 A_i 最新值. T_{A_i} 称为 A_i 相对于 Q 的最新记录集合，形式化定义如下.

定义 3.　A_i 相对于 Q 的最新记录集合. 设 D 是数据模式 R 的数据集合实例，e_Q 是 D 中某个实体，查询 $Q=(e_Q,Attr(Q),CVQ)$，记录集合 T_{A_i} 称为 A_i 相对于 Q 的最新记录集合，当且仅当对 $\forall t\in T_{A_i}$，t 满足下述两个条件：

① $t[EID]$等于 e_Q 的 ID；

② 不存在记录 s，使得根据 R 在 A_i 上的时效约束能够推出 $t\prec_{A_i}s$.

理想情况下，e_Q 的 A_i 最新值唯一确定，即下述两种情况其中之一成立：T_{A_i} 只包含一条记录；或 T_{A_i} 包含多条记录，但对 $\forall s,t\in T_{A_i}$ 有 $s[A_i]=t[A_i]$. 这时可认为 e_Q 的 A_i 最新值的确定度为 1. 但因时效约束作用有限，所以可能存在 $s,t\in T_{A_i}$，有 $s[A_i]\neq t[A_i]$，这时 A_i 有多个可能的最新值. 设 T_{A_i} 中的记录共有 $cnt(T_{A_i})$ 种不同的 A_i 值，那么 e_Q 的 A_i 最新值的确定度就是 $1/cnt(T_{A_i})$. $1/cnt(T_{A_i})$ 称为 A_i 对 Q 的时效性贡献，对 $Attr(Q)$ 中所有属性的时效性贡献求平均，就得到 Q 查询结果的时效性. 下面给出 CVQ 查询结果时效性的形式化定义.

定义 4.　CVQ 查询结果的时效性. 设 D 是模式 R 的数据集合实例，查询 $Q=(e_Q,Attr(Q),CVQ)$，Q 的查询结果的时效性 cur_Q 定义如下：

$$cur_Q=\frac{1}{|Attr(Q)|}\sum_{A_i\in Attr(Q)}\frac{1}{cnt(T_{A_i})} \quad (5)$$

其中，$|Attr(Q)|$ 是 $Attr(Q)$ 中属性的数目；T_{A_i} 是 A_i 相对于 Q 的最新记录集合.

当 $Attr(Q)$ 中各属性重要性不同时，在判定时效性时，可对属性按重要性加权. 设属性 A_i 权值为 w_{A_i}，则 $\sum_{A_i\in Attr(Q)}w_{A_i}=1$. 式(5)等价于各属性权值均

为 $1/|Attr(Q)|$ 的情况，当 $Attr(Q)$ 中各属性权值不相等时，可以将式(5)修改为式(6)，用式(6)来计算查询结果的时效性.

$$cur_Q=\sum_{A_i\in Attr(Q)} w_{A_i}\times\frac{1}{cnt(T_{A_i})} \tag{6}$$

解决 CVQ 的查询相关时效性判定问题的第 2 步是将所有的 CVQ 查询作为一个整体，判定数据相对于所有 CVQ 查询的平均时效性(称为 CVQ 平均时效性). CVQ 平均时效性与 CVQ 查询结果的时效性的定义非常相似，区别仅在于 CVQ 平均时效性需要对数据集合 D 中的每个实体的每个属性计算时效性贡献，对所有的时效性贡献求加权和，以之作为 D 的 CVQ 平均时效性.

3.2.2 时效序列查询

时效序列查询(Currency Sequence Query, CSQ)关心实体的历史信息，即描述 e_Q 的记录在 $Attr(Q)$ 中的属性上的时序关系. 典型的 CSQ 形如查询 Q_2："Alice 先后在哪些城市工作过". 即 $Q_2=$(Alice,{$City$},CSQ). 在图 1 所示的数据集合实例中，Alice 的 $City$ 属性有 4 个不同值："Beijing"、"Shanghai"、"Guangzhou"、"Harbin". 回答 Q_2 时，需要找到相应的 4 条记录在 $City$ 属性上，根据时序关系"$\prec_{City}$"排成的序列，如 $t_1\prec_{City}t_2\prec_{City}t_3\prec_{City}t_4$，我们将这样的序列称为时效序列.

为了判定 CSQ 查询相关时效性，首先要对给定的数据集合 D 和查询 Q，判定 Q 在 D 上的查询结果的时效性. CSQ 不关心最新值，而关心记录的时序关系，如果所需的时序关系不能被唯一确定，那么 CSQ 就无法被准确回答，其查询结果的时效性也受到影响. 设 $Q=(e_Q,Attr(Q),\text{CSQ})$，为了回答 Q，需要为 e_Q 在 $Attr(Q)$ 中每个属性上找到最长的时效序列. 设 e_Q 在 A_i 上的最长时效序列为 Sq_{A_i}，当前数据集合实例中共有 l 条不同的记录描述实体 e_Q，则理想情况下 Sq_{A_i} 的长度(记为 $|Sq_{A_i}|$)应当为 l. 这时 e_Q 在属性 A_i 上的时效序列的确定度为 1. 但因为时效约束作用有限，所以通常情况下 $|Sq_{A_i}|<l$，此时 e_Q 在 A_i 上的时效序列的确定度为 $|Sq_{A_i}|/l$. $|Sq_{A_i}|/l$ 称为 A_i 对 Q 的时效性贡献，对 $Attr(Q)$ 中所有属性的时效性贡献求平均，就得到 Q 查询结果的时效性. CSQ 查询结果时效性的形式化定义如下.

定义 5. CSQ 查询结果的时效性. 设 D 是模式 R 的数据集合实例，查询 $Q=(e_Q,Attr(Q),\text{CSQ})$，$Q$ 的查询结果的时效性 cur_Q 定义如下：

$$cur_Q=\frac{1}{|Attr(Q)|}\sum_{A_i\in Attr(Q)}|Sq_{A_i}|/l \tag{7}$$

其中，$|Attr(Q)|$、Sq_{A_i} 和 l 的定义与前文相同.

与 CVQ 类似，当属性重要性不同时可以对 $Attr(Q)$ 中属性按重要性加权，并以 $Attr(Q)$ 中属性时效性贡献的加权和作为 Q 的查询结果的时效性.

判定 CSQ 查询相关时效性的第二步是将所有的 CSQ 查询作为一个整体，判定其查询结果的平均时效性(称为 CSQ 平均时效性). 与 CVQ 类似，可对数据集合 D 中的每个实体的每个属性计算时效性贡献，对所有的时效性贡献求加权和，以之作为数据集合 D 的 CSQ 平均时效性.

3.3 用户相关时效性

如引言中所述，用户需求往往通过查询来体现，具体表现在用户使用数据时往往偏好某一类、甚至某几个查询. 可以按照对查询的偏好将用户分为 3 类：第 1 类用户对查询没有偏好；第 2 类用户对 CVQ 和 CSQ 的平均时效性要求不同，若用户更关心实体当前的状态，则对 CVQ 平均时效性要求更高，若用户更关心记录的时序关系，则对 CSQ 平均时效性要求更高；第 3 类用户对某些常用查询的结果时效性要求较高，例如，薪酬分析人员经常查询员工的工资信息，其常用查询集合里就包含那些和"$Salary$"属性相关的 CVQ 或 CSQ 查询.

第 1 类用户对查询没有偏好，故在判定该类用户相关时效性时，所有查询的重要性权值相等. 所以，第 1 类用户相关时效性定义为 CVQ 和 CSQ 的平均时效性的平均数.

对于第 2 类用户来说，CVQ 和 CSQ 的重要程度不同，因此，第 2 类用户相关时效性定义为 CVQ 和 CSQ 平均时效性的加权和，权值取决于该类查询对用户的重要程度，CVQ 和 CSQ 的权值之和等于 1.

第 1 类和第 2 类用户相关时效性(记为 cur_{user})可统一表示为式(8).

$$cur_{\text{user}}=w_{\text{CVQ}}\times cur_{\text{CVQ}}+w_{\text{CSQ}}\times cur_{\text{CSQ}} \tag{8}$$

在判定对第 1 类用户相关时效性时只需令式(8)中的 $w_{\text{CVQ}}=w_{\text{CSQ}}=0.5$ 即可.

第 3 类用户对某些常用查询的结果时效性要求较高，设这些查询组成的集合为 $\mathbb{Q}$，则需要为 $\mathbb{Q}$ 中的查询一一判定其结果的时效性. 再根据查询的重要程度给 $\mathbb{Q}$ 中的查询赋予相应的权值，并计算其查询结果时效性的加权和，就得到第 3 类用户相关时效性. 第 3 类用户相关时效性定义如式(9)所示：

$$cur_{\text{user}}=\sum_{Q_i\in\mathbb{Q}} w_{Q_i}\times cur_{Q_i} \tag{9}$$

其中，cur_{user} 表示用户相关时效性，Q_i 是 $\mathbb{Q}$ 中的查询，w_{Q_i} 是 Q_i 的权值且 $\sum_{Q_i \in \mathbb{Q}} w_{Q_i} = 1$，$cur_{Q_i}$ 是 Q_i 的结果时效性.

3.4　讨　论

对于同一个数据集合，用不同的时效性判定方法得到的结果可能完全不同. 查询结果时效性描述的是数据相对于查询的时效性，其仅能反映查询涉及到的那一小部分数据的时效性如何，如果该部分数据时效性好，则相应的查询结果时效性就高. 而查询平均时效性和用户相关时效性侧重于描述数据整体的时效性，其中两类查询平均时效性分别从最新值和时序关系两个角度描述了数据整体的时效性状况，而用户相关时效性则是针对不同用户需求，描述当前数据集合的时效性状况.

在得到查询相关时效性和用户相关时效性后，可以进一步给出能够提高时效性的数据修复建议.

对于某些结果时效性较差的查询，可以按照查询所属类别，着重修复查询涉及到的记录和属性，即确定其最新值，或恢复时序关系.

对于第 1 类和第 2 类用户，如果用户关心的某一类查询的平均时效性特别差，则可以着重针对这类查询来修复数据. 例如，在判定第 1 类用户相关时效性时，当前数据集合实例的 CVQ 平均时效性为 0.9，而 CSQ 的平均时效性仅 0.5，则应当着重恢复数据的时序关系. 对于第 3 类用户，可以根据 $\mathbb{Q}$ 中查询的查询相关时效性来修复结果时效性较差的查询涉及到的那部分数据，从而提高用户相关时效性.

4　数据时效性判定问题求解

4.1　时效图

不失一般性地，可设数据集合实例 D 中描述实体 e 的记录构成的集合为 $S_e = \{t_1, \cdots, t_k\}$，对 S_e 中记录 t_i 和 t_j，可能存在 $t_i[A] \neq t_j[A]$ 的情况，假设数据中没有错误，则时序关系 $t_i \prec_A t_j$ 和 $t_j \prec_A t_i$ 必有一个成立. 时效图 $G_{e,A}$ 用于描述 S_e 中记录在 A 上的时序关系，形式化定义如下.

定义 6.　时效图(currency graph). 设 e 为数据模式 R 的数据集合实例 D 中的实体，A 是 R 中属性，$S_e = \{t_1, \cdots, t_k\}$ 是 D 中描述 e 的记录构成的集合，有向图 $G_{e,A} = (V, E)$ 称为 e 在 A 上的时效图，当且仅当下述条件同时满足：

① $|V| = |S_e|$，且 V 中的任意结点 v_i 对应 S_e 中的记录 t_i；

② $t_i \prec_A t_j$ 当且仅当 E 中存在有向边 (v_i, v_j).

时效图 $G_{e,A}$ 的构造算法如算法 1 所示. 算法的输入为实体 e，属性 A，描述 e 的记录构成的集合为 S_e 及数据模式 R 上的时效约束的全体构成的集合 CC_R，输出为 e 在 A 上的时效图 $G_{e,A}$. 算法第 1 行初始化图 G，令其结点和边集和均为空；第 2 到 5 行向结点集合 V 中添加结点，使得 V 中结点和 S_e 中记录一一对应；第 6 到 12 行检查 CC_R 中每一条约束，如果当前约束 cc 是 A 上的时效约束(A 上的时效约束可能不止一条)，且存在 $t_i, t_j \in S_e$，使得根据 cc 能判定 $t_i \prec_A t_j$ 成立，则向边集合 E 中添加有向边 (v_i, v_j)；循环结束后，图 G 即为时效图 $G_{e,A}$.

算法 1.　时效图 $G_{e,A}$ 的构造算法.

输入：e, A, S_e, CC_R

输出：$G_{e,A}$

```
1.  G←(V,E); V←∅; E←∅;
2.  for each t_i ∈ S_e do
3.      add v_i into V;
4.      tag(v_i)←t_i;
5.  end for
6.  for each cc∈CC_R do
7.    if cc is a currency constraint of A then
8.      for each t_i≺t_j that can be inferred from cc do
9.          add (v_i,v_j) into E;
10.     end for
11.   end if
12. end for
13. G_{e,A}←G;
14. return G_{e,A}.
```

下面我们用一个例子来说明算法 1 的执行过程，并在 4.3 节讨论算法 1 的时间复杂度.

例 2.　考虑图 1 所示数据及式(1)和式(2)所示的时效约束. 式(1)和式(2)分别是属性 *Salary* 和 *City* 上的时效约束，则可以对实体 Alice 的属性 *Salary* 和属性 *City* 构造时效图，如图 2(a)和(b)所示. 当数据集合中不存在可以表示属性各个取值的插入和生成时间的时间戳时，时效图可以用来判断哪些值是(或更有可能是)最新的.

在图 2(a)中，我们将代表 t_3 和 t_4 的结点合并成为一个结点，这是因为 t_3 和 t_4 的 *Salary* 属性值相同，则根据式(1)，t_3 和 t_4 的时效性应当相同(它们都具有最新的 *Salary* 值)，在下文的算法中，我们会在需要时将这样的结点合并.

对于任意实体 e 及其属性 A，其时效图在时效

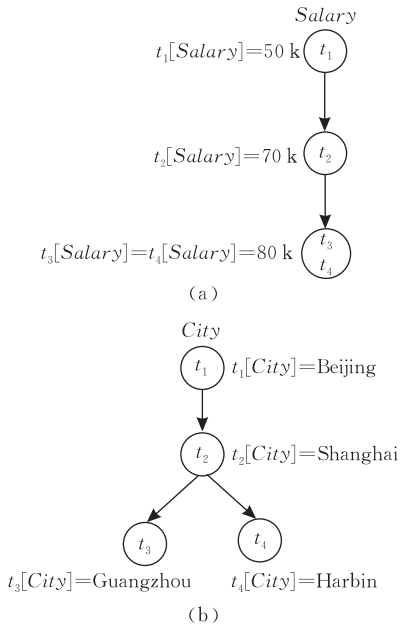


图 2 实体 Alice 的 *Salary* 属性和 *City* 属性对应的时效图

约束给定后就确定了,因此可以将计算时效图的工作作为数据预处理,对每个实体的每个属性预先生成时效图并存储,从而在判定时效性时减少因反复生成时效图而产生的时间开销.

4.2 查询相关时效性判定问题求解

4.2.1 CVQ 查询结果时效性及平均时效性判定

我们首先给出 CVQ 查询结果的时效性判定算法.

设查询 $Q=(e_Q,Attr(Q),\text{CVQ})$,如 3.2.1 节所述,$Q$ 要求 e_Q 的 $Attr(Q)$ 中每个属性有唯一确定的最新值.而在时效图中,某结点出度为 0 表示该点不比图中任何其它结点旧.所以在判定查询结果的时效性时,首先应当找到 $Attr(Q)$ 中每个属性 A_i 的时效图 G_{e_Q,A_i} 中所有出度为 0 的点,这些点对应的记录构成定义 4 中的 T_{A_i} 集合.当时效约束相互间没有冲突时,时效图是有向无环图(因为时序关系是传递关系,所以如果时效图中有环,则说明存在 $t_i \prec_A t_j$ 且 $t_j \prec_A t_i$ 的情况,即约束出现了冲突),此时时效图中至少有 1 个点出度为 0.在找到所有出度为 0 的点后,利用式(6)就可判定 Q 的查询结果的时效性.

CVQ 查询结果的时效性判定算法如算法 2 所示.算法输入为 $Q=(e_Q,Attr(Q),\text{CVQ})$ 和权值向量 $\boldsymbol{W}$,其中 $\boldsymbol{W}$ 用于设定 $Attr(Q)$ 中属性的权值,输出为二元组 (ans_Q,cur_Q),ans_Q 用于记录 $Attr(Q)$ 中各个属性的最新值,cur_Q 为 Q 查询结果的时效性.算法首先用 FindCurrentValue(G_{e_Q,A_i}) 为 $Attr(Q)$ 中每个属性 A_i 找到所有可能的最新值,并计算 A_i 对 Q 的时效性贡献,分别记录于 (ans_{A_i},cur_{A_i}),然后对所有的 cur_{A_i} 求加权和,得到 Q 的查询结果的时效性 cur_Q.

算法 2. CVQ 查询结果的时效性判定.

输入:$Q=(e_Q,Attr(Q),\text{CVQ}),\boldsymbol{W}$

输出:(ans_Q,cur_Q)

1. $cur_Q \leftarrow 0$;
2. for each $A_i \in Attr(Q)$ do
3. load G_{e_Q,A_i};
4. $(ans_{A_i},cur_{A_i}) \leftarrow$ FindCurrentValue(G_{e_Q,A_i});
5. $cur_Q \leftarrow cur_Q + w_{A_i} \times cur_{A_i}$;
6. end for
7. $ans_Q \leftarrow (ans_1,\cdots,ans_{|Attr(Q)|})$;
8. return (ans_Q,cur_Q).

FindCurrentValue(G_{e_Q,A_i}) 如算法 3 所示.算法 3 将时效图 G_{e_Q,A_i} 中的每个出度为 0 的点 v_i 对应的记录 t_i 加入集合 T_{A_i},并将 $t_i[A]$ 加入集合 ans_{A_i}.在检查完所有出度为 0 的点之后,令 $cnt(T_{A_i})$ 等于 ans_{A_i} 中不同的 A_i 值的个数,并以 $1/cnt(T_{A_i})$ 作为属性 A_i 的时效性贡献,即 cur_{A_i}.

算法 3. FindCurrentValue(G_{e_Q,A_i}).

输入:$G_{e_Q,A_i}=(V,E)$

输出:(ans_{A_i},cur_{A_i})

1. $ans_{A_i} \leftarrow \varnothing$, $T_{A_i} \leftarrow \varnothing$;
2. for each $v_i \in V$ do
3. if $outdegree(v_i)==0$ then
4. add t_i into T_{A_i}
5. add $t_i[A]$ into ans_{A_i}
6. end if
7. end for
8. $cnt(T_{A_i}) \leftarrow$ the number of unique values in ans_{A_i}
9. $cur_{A_i} \leftarrow 1/cnt(T_{A_i})$;
10. return (ans_{A_i},cur_{A_i}).

下面用一个例子来给出算法 2 的直观解释.

例 3. 考虑图 2 中的两个时效图,设查询为“Alice 当前的工作城市和工资”,即 $Q=(\text{Alice},\{City,Salary\},\text{CVQ})$,权值向量 $\boldsymbol{W}=(0.5,0.5)$.根据图 2,$T_{City}=\{t_3,t_4\}$,$ans_{City}=\{\text{Guangzhou},\text{Harbin}\}$,$T_{Salary}=\{t_3,t_4\}$,$ans_{Salary}=\{80\text{k}\}$.进而可计算得到 $cur_{City}=0.5$,$cur_{Salary}=1$,$cur_Q=0.5\times 0.5+0.5\times 1=0.75$,$ans_Q=\{\{\text{Guangzhou},\text{Harbin}\},\{80\text{k}\}\}$.

接下来我们讨论 CVQ 平均时效性的判定方法.如 3.2.1 节所述,判定 CVQ 平均时效性需要对数据集合 D 中的每个实体的每个属性计算时效性贡献,并对所有的时效性贡献求加权和.上述工作可

以分 3 步完成：

(1) 构造一个特殊的查询 Q'，令 $Attr(Q')$ 等于模式 R 的所有属性，并为每个属性赋予权值.

(2) 对于数据集合 D 的每个实体 e，令 $e_{Q'}=e$，并执行一遍算法 2.

(3) 将所有的第 2 步中得到的结果根据实体的重要性做加权和，作为 CVQ 平均时效性.

4.2.2　CSQ 查询结果时效性及平均时效性判定

我们首先讨论 CSQ 查询结果的时效性判定方法.

设查询 $Q=(e_Q, Attr(Q), \text{CSQ})$，如 3.2.2 节所述，为了判定 Q 的查询结果的时效性，需要对 e_Q 的 $Attr(Q)$ 中每个属性求最长时效序列. 因为时效图中的边表示时序关系“$\prec_A$”，所以求最长时效序列等价于在时效图中寻找最长路径. 如果能够完美地回答 Q，则 $Attr(Q)$ 中每个属性的时效图的最长路径都是哈密顿路. 那么，最长路径长度与哈密顿路长度之比就等于定义 5 中的 $|Sq_{A_i}|/l$，对 $Attr(Q)$ 中每个属性计算 $|Sq_{A_i}|/l$，再使用式(7)即可判定 Q 的查询结果的时效性.

最长路径问题是一个 NP 难问题，但是由于时效图均为有向无环图，所以可以利用拓扑排序在多项式时间内解决该问题.

CSQ 查询结果的时效性判定问题的求解算法如算法 4 所示. 算法输入为 $Q=(e_Q, Attr(Q), \text{CSQ})$ 和权值向量 $\boldsymbol{W}$，其中 $\boldsymbol{W}$ 用于 $Attr(Q)$ 中属性权值的设定. 输出为二元组 (ans_Q, cur_Q)，ans_Q 记录 $Attr(Q)$ 中各个属性的对应的最长时效序列，cur_Q 为 Q 查询结果的时效性. 算法先用 FindLongestSequence(G_{e_Q,A_i}) 对 $Attr(Q)$ 中每个属性 A_i，求时效图 G_{e_Q,A_i} 中的最长路径并计算 A_i 的时效性贡献，记录于 (ans_{A_i}, cur_{A_i})，然后对所有的 cur_{A_i} 求加权和，得到 Q 查询结果的时效性 cur_Q.

算法 4.　CSQ 查询结果的时效性判定.

输入：$Q=(e_Q, Attr(Q), \text{CSQ}), \boldsymbol{W}$

输出：(ans_Q, cur_Q)

1. $cur_Q \leftarrow 0$;
2. for each $A_i \in Attr(Q)$ do
3. 　load $G_{e_Q(A_i)}$;
4. 　$(ans_{A_i}, cur_{A_i}) \leftarrow$ FindLongestSequence(G_{e_Q,A_i});
5. 　$cur_Q \leftarrow cur_Q + w_{A_i} \times cur_{A_i}$;
6. end for
7. $ans_Q \leftarrow (ans_1, \cdots, ans_{|Attr(Q)|})$;
8. return (ans_Q, cur_Q).

FindLongestSequence(G_{e_Q,A_i}) 如算法 5 所示，其基本思想是按照拓扑排序给图中的结点赋予相应的层次编号，在对所有的结点都赋予层次编号之后，最大层次编号就是最长路径的长度. 注意，同一层的结点对应的属性值可能相同(如图 2(a)中 t_3 和 t_4 对应的 $Salary$ 属性值均为 80k)，从而其新旧程度相同，需要将这样的结点合并.

算法 5.　FindLongestSequence(G_{e_Q,A_i}).

输入：$G_{e_Q,A_i}=(V,E)$

输出：(ans_{A_i}, cur_{A_i})

1. $ans_{A_i} \leftarrow \varnothing$, $level \leftarrow 0$, $cnt \leftarrow 0$;
2. while $V \neq \varnothing$ do
3. 　$vset \leftarrow$ Find0IndegreeNodes(G_{e_Q,A_i});
4. 　merge the nodes with same A_i value in $vset$
5. 　$cnt \leftarrow cnt + |vset|$
6. 　for each $v_i \in vset$ do
7. 　　$v_i.lv = level$;
8. 　end for
9. 　$level \leftarrow level + 1$;
10. 　add $vist$ into ans_{A_i};
11. 　remove the nodes in $vset$;
12. end while
13. $cur_{A_i} = level/cnt$
14. return (ans_{A_i}, cur_{A_i}).

下面用例 4 来直观地说明算法 4.

例 4.　考虑查询“Alice 先后在哪些城市工作过”，则查询 $Q=(\text{Alice}, \{City\}, \text{CSQ})$. 图 2(b)所示为 Alice 的属性 $City$ 的时效图，可将其等价的转化为图 3(b)所示的时效序列的形式，其中横轴上的点表示层次编号(如 t_1 属于层次 1)，层次编号越大，属性值越新，如果两条存有不同值的记录属于同一层，则说明在当前约束下二者的时序关系无法判定.

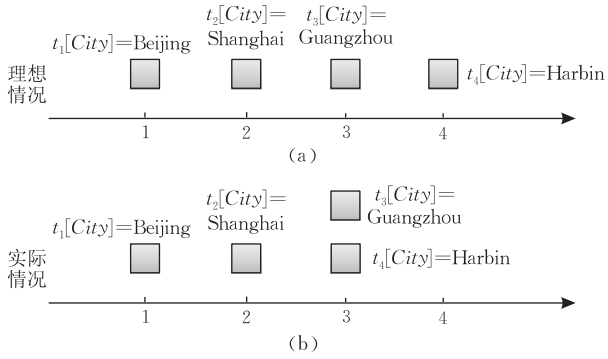


图 3　时效序列

理想的时效序列长度为 4，如图 3(a)所示，为 $t_1 \prec_{City} t_2 \prec_{City} t_3 \prec_{City} t_4$，但实际情况如图 3(b)所示，最长时效序列长度为 3. 因此属性 $City$ 对于 Q 的时效性贡献为 3/4=0.75. 又因为 $Attr(Q)$ 中仅包

含一个属性，所以 Q 的查询结果的时效性为 0.75.

CSQ 平均时效性的判定方法与 CVQ 平均时效性的判定方法基本相同，即可以类似地定义查询 Q'，并在数据集合的所有实体上反复执行算法 4，最后对所有得到的时效性贡献求加权和得到 CSQ 平均时效性. 在此我们不再赘述方法细节.

4.2.3 时间复杂度分析

设数据库模式 R 有 n 个属性，数据集合 D 中共有 m 条记录，共有 c 条时效约束.

对于算法 1，设检查一条时效约束的可满足性所需的最长时间为 r，则最坏情况下算法 1 需要对每对记录检查所有的时效约束是否满足，时间复杂度为 $O(m^2\times c\times r)$. r 取决于时效约束的具体形式，复杂的规则虽然有较强的表达能力，但往往需要很长时间来检查其前件是否成立，反而会给时效性判定带来负面影响. 因此，本文要求时效约束定义(定义 2)中的条件 ψ 的形如规则 1，当约束均满足规则 1 时，可以在多项式时间内完成时效图的构建.

规则 1. 时效约束定义中的条件 ψ 必须满足下述条件：

① ψ 是若干谓词的合取，即形如 $p_1\wedge\cdots\wedge p_h$(可将 $(p_i\vee p_j)\wedge p_k$ 改写为 $(p_i\wedge p_k)\vee(p_j\wedge p_k)$ 并进一步拆为两条规则)，其中，h 非常小，可以认为是常数；p_i 表示第 i 个条件，如 $t[Salary]<s[Salary]$；

② 对于 ψ 中的任意 p_i，可以在常数时间或多项式时间内判定其是否为真.

规则 1 有两个作用. 首先，规则 1 能帮助用户或领域专家写出更规范的约束，从而更方便计算机判定约束的可满足性. 其次，规则 1 虽然使得时效约束的表达能力受到了一定限制，但是该规则能够保证约束的可满足性可以在多项式时间内被判定(至多只需检查 h 个谓词，每个谓词至多需多项式时间的).

算法 2 调用了算法 3，因此首先分析算法 3 的时间复杂度. 算法 3 中 2 至 7 行的循环至多进行 $|V|$ 次，第 8 行在计算 ans_{A_i} 中不同的 A_i 值的个数时，朴素的两两对比的方法至多比较 $|V|^2$ 次，而 $|V|$ 在最坏情况下等于 m，因此算法 3 的时间复杂度为 $O(m^2)$.

算法 2 中 2 至 6 行的循环共执行 $|Attr(Q)|$ 次，每次循环调用一遍算法 3，在最坏情况下，$|Attr(Q)|=n$，故算法 2 的时间复杂度为 $O(n\times m^2)$.

注意，我们将生成时效图作为数据预处理，因此在算法 2 的第 3 行只读取时效图，而没有考虑时效图生成的时间. 除此之外，算法 3 中，可以利用 Hash 函数将不同的属性值映射到不同的桶，则第 8 行的时间开销可以降到 $O(1)$，从而将算法 3 和算法 2 的时间复杂度分别降至 $O(m)$ 和 $O(n\times m)$.

算法 4 调用了算法 5，因此首先分析算法 5 的时间复杂度. 算法 5 的第 4 行至多需执行 $|V|^2$ 次结点的两两比对，故 2～12 行的循环最坏情况需执行 $|V|^3$ 次. $|V|$ 在最坏情况下等于 m，因此算法 5 的时间复杂度为 $O(m^3)$.

算法 4 中 2～6 行的循环共执行 $|Attr(Q)|$ 次，在最坏情况下 $|Attr(Q)|=n$，故算法 4 的时间复杂度为 $O(n\times m^3)$.

与算法 2 和算法 3 同理，可以使用 Hash 函数将算法 5 的第 4 行开销降至 $O(1)$，从而将算法 4 和算法 5 的时间复杂度降至 $O(n\times m^2)$ 和 $O(m^2)$.

4.3 用户相关时效性判定问题求解

用户相关时效性定义如 3.3 节所述. 对于 3.3 节所述的第 1 类和第 2 类用户，可以利用 4.2 节的方法求得 CVQ 平均时效性和 CSQ 平均时效性，再利用式(8)就可求得相应的用户相关时效性.

对于 3.3 节所述的第 3 类用户，需要对 $\mathbb{Q}$ 中每个查询，根据其所属的查询类别使用算法 2 或算法 4 求得查询结果时效性，再使用式(9)求得相应的用户相关时效性.

5 实验结果及分析

5.1 实验配置

实验采用的硬件为 Intel Core i5 CPU，4 GB 内存，操作系统为 Windows 7，代码使用 C++ 编写，软件开发环境为 Microsoft Visual Studio 2010.

因为之前没有其它工作能够给出数据时效性的判定算法，所以本节实验主要研究本文的时效性判定方法中各个参数对算法的影响. 我们在真实数据集合和虚拟数据集合上分别对算法 1、算法 2、算法 4 及 4.3 节的用户相关时效性判定方法进行了实验. 其中真实数据用于研究数据时效性判定结果对规则数目的敏感程度，虚拟数据用于验证本文算法的执行效率.

真实数据集由 100 条个人信息组成，数据模式 $Stu=(tID,EID,Name,Gender,Edu,Occu,Ent/Sch,Addr,Salary,Age)$，各属性分别表示记录 ID、实体 ID、姓名、性别、学历、职业、工作单位/学校地址、住址、工资和年龄. 真实数据集合中包含 10 个实

体，每个实体有 10 条不同的记录描述其信息. 数据模式 Stu 上共定义了 15 条时效约束，例如 $\forall s,t$: $Stu(s[EID]=t[EID] \wedge t[Age]<s[Age] \rightarrow t \prec_{Age} s)$ 和 $\forall s,t$: $Stu(s[EID]=t[EID] \wedge t[Edu]=\text{Bachelor} \wedge s[Edu]=\text{Master} \rightarrow t \prec_{Edu} s)$. 我们通过改变约束的数目来研究其对数据库时效性判定结果的影响.

虚拟数据集合采用自己编写的数据生成工具生成，其 5 个主要参数如下：$entityNum$ 用于控制实体的个数；为了方便衡量记录数对实验效率，我们将所有实体对应的记录数设为相等，用参数 $tupleNum$ 控制；$attrNum$ 用于控制数据库的模式的属性数目；$constraintNum$ 用于控制约束的数目，生成工具生成的约束仅对一对记录生效，且均可以在常数时间内被判定可满足性；$queryNum$ 用于控制查询的数目. 我们每次变动一个参数，固定其余 4 个参数，并研究该参数的变化对于算法效率的影响.

5.2　规则数目对时效性判定结果的影响

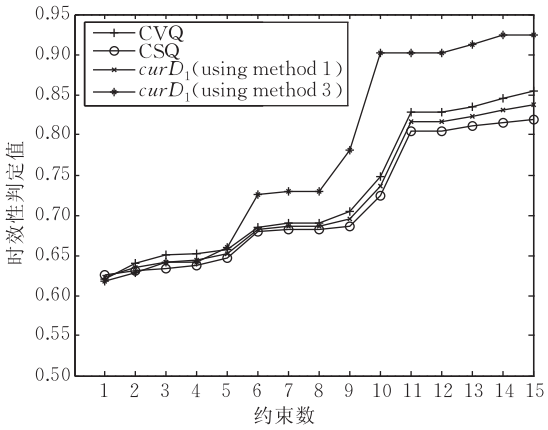
在真实数据集上有 15 条时效约束，根据被这 15 条时效约束的覆盖的程度（覆盖程度越高，则时效约束影响的记录越多，因而越有助于恢复数据的时序关系），可以将真实数据集合 D 划分成两个子集合 D_1 和 D_2，D_1 和 D_2 各包含 50 条记录，D_1 被时效约束覆盖得较好，D_2 被时效约束覆盖得较差.

查询集合 $\mathbb{Q}$ 被用来判定 D 的第 3 类用户相关时效性. $\mathbb{Q}$ 可被划分为子集合 $\mathbb{Q}_1$ 和 $\mathbb{Q}_2$，其中 $\mathbb{Q}_1$ 中的查询仅针对 D_1 中的实体，而 $\mathbb{Q}_2$ 中的查询仅针对 D_2 中的实体.

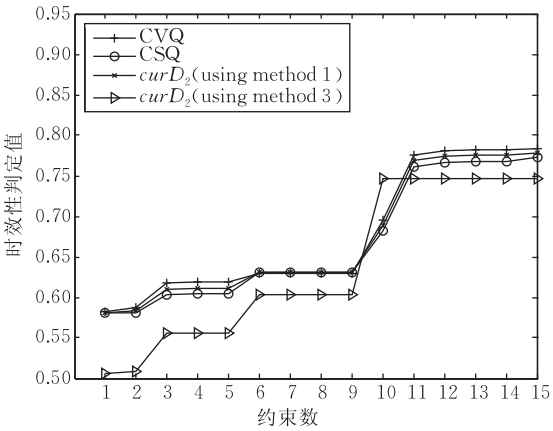
我们在 D_1、D_2 和 D 上分别进行实验，并比较这 3 种情况下查询相关时效性以及用户相关时效性判定结果的变化.

图 4 中(a)～(c) 3 幅图分别为时效约束增加时，D_1、D_2 和 D 的时效性判定结果变化情况. CVQ 和 CSQ 两条折线表示当约束增加时，数据集合的 CVQ 和 CSQ 平均时效性上升情况. 图 4(a)中折线 $curD_1$(using method 1)表示 D_1 的第 1 类用户相关时效性判定结果，折线 $curD_1$(using method 3)表示 D_1 的第 3 类用户相关时效性判定结果，图 4(b)同理. 图 4(c)中的 $curD$ (using method 3 and $\mathbb{Q}_1$)和 $curD$ (using method 3 and $\mathbb{Q}_2$)两条折线分别是使用 $\mathbb{Q}_1$ 和 $\mathbb{Q}_2$ 对 D 做第 3 类用户相关时效性判定的结果. 因第 1 类用户相关时效性相当于第 2 类用户相关时效性权值相等的情况，所以不必专门对第 2 类用户相关时效性进行实验. 由本组实验可以得到 3 个结论.

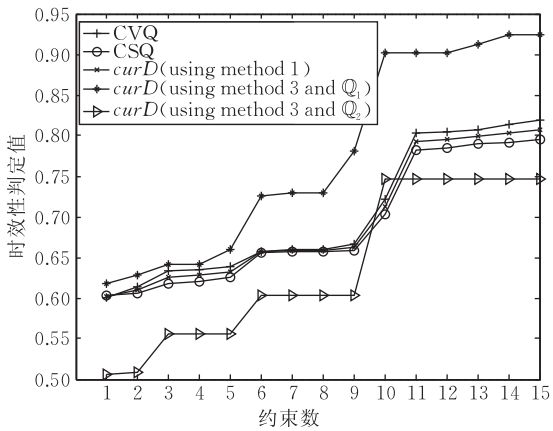
(1) 当时效性约束增多时，各类时效性判定结果在数值上均呈上升的趋势，但约束的效力各不相同. 从图 4 中可以看出，第 6 和第 10 条时效约束加入时，D_1、D_2 和 D 的各类时效性上升最快，这是因为这两条时效性约束能够影响的记录最多；而第 7 和第 11 条约束加入时，数据时效性并没有明显的上升，这是因为这两条约束只影响了非常少量的记录，因此对数据时效性几乎无影响.



(a) 数据集合D_1的时效性判定结果



(b) 数据集合D_2的时效性判定结果



(c) 数据集合D的时效性判定结果

图 4　时效约束数对 D_1、D_2 和 D 的时效性判定结果的影响

(2) 随着时效约束的增多,第 3 类用户相关时效性提升最快.这种现象的出现是因为第 3 类用户相关时效性仅关心查询涉及到的实体和属性,因此受时效约束影响最大.

(3) 被时效约束覆盖越好的数据集合的时效性越高.D_1 被时效约束覆盖的最好,当时效约束数目增加时,其各类时效性判定结果始终最高,约束数目等于 15 时,各类时效性均能达到 0.8 以上,第 3 类用户相关时效性更是达到了 0.92.与之相比,D_2 被约束覆盖的较差,则各类时效性均较低,约束数目等于 15 时,各类时效性都只有 0.75 左右.$D=D_1 \cup D_2$,其被约束覆盖的程度介于 D_1 和 D_2 之间,因此各类时效性判定结果也介于二者之间.

图 5 所示为使用 $\mathbb{Q}_1$、$\mathbb{Q}_2$、$\mathbb{Q}$ 分别对 D 判定第 3 类用户相关时效性得到的结果.可以观察到,使用不同的查询集合判定得到的第 3 类用户相关时效性差异很大.如前文所述,查询集合反映用户需求,该结果说明需求不同时,数据相对于用户来说,时效性也不同.除此之外还可以观察到,当使用 $\mathbb{Q}_1$ 来判定 D 的用户相关时效性时,某些时效约束(如约束 9)能较大幅度地提升时效性,但使用 $\mathbb{Q}_2$ 时该约束不起任何作用.这说明需求不同时约束的作用也不同,并非所有约束都能真正提升数据时效性.

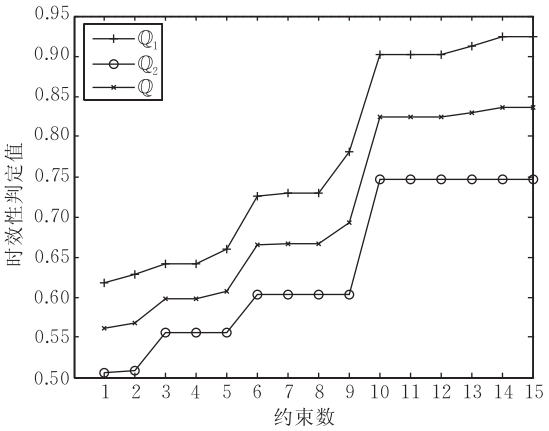


图 5 使用不同的查询集合判定 D 的第 3 类用户相关时效性

5.3 各参数对算法执行效率的影响

本小节用虚拟数据集合分析不同参数的改变对算法执行效率的影响,各参数意义如 5.1 节所述.在每组实验中,我们让被研究的参数从 1 k 变到 10 k,并将实验分为两步.

第 1 步,研究 *entityNum*、*tupleNum*、*attrNum*、*constraintNum* 对算法执行时间的影响,每次改变 1 个参数取值,并固定其余 4 个参数,按照当前参数取值,用数据生成工具随机生成数据集合和约束,并做如下 3 步工作:(1) 为数据集合中所有实体在所有属性上构造时效图(build currency graph);(2) 判定数据集合的 CVQ 平均时效性(evaluate CVQs);(3) 判定数据集合的 CSQ 平均时效性(evaluate CSQs).第 1 和第 2 类用户相关时效性判定正好由上述 3 个过程构成.3 个过程的执行时间如图 6~图 9 所示,其中各图的横轴表示参数取值的变化,纵轴表示算法执行时间,图 6、图 8、图 9 的纵轴单位为 ms,图 7 的纵轴单位为 10^4 ms.

第 2 步,利用随机生成的包含 *queryNum* 个查询的集合判定第 3 类用户相关时效性,并研究 *queryNum* 的变化对其执行时间的影响,结果如图 10 所示,其中各图的横轴表示 *queryNum* 的取值,纵轴表示执行时间,单位为 ms.

图 6 所示为 *entityNum* 对算法执行效率的影响.本组实验中,为了把其它参数的影响尽量降到最低,我们令 *tupleNum*、*attrNum* 和 *constraintNum* 都等于 10,则可认为算法 1、2、4 对于单个实体执行都只需常数时间.从图 6 可以看出,3 种工作的执行时间随 *entityNum* 的变化基本呈线性增长,并且均能在 0.12 s 内完成.

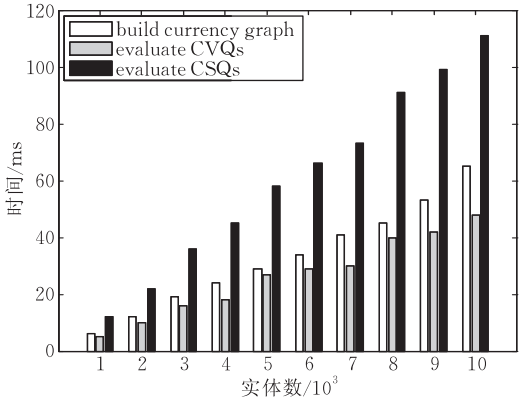


图 6 参数 *entityNum* 对算法执行效率的影响
(*attrNum*=*tupleNum*=*constraintNum*=10)

图 7 所示为 *tupleNum* 对算法执行效率的影响.本组实验中 *entityNum*、*attrNum* 和 *constraintNum* 都等于 10. 4.2.3 节中的理论分析表明,CVQ 和 CSQ 相关时效性判定算法(不使用 Hash 函数优化)的时间复杂度是记录数目的平方或三次方量级,这与图 6 所示的实验结果相符.在 *tupleNum*=10 k 时,判定 CVQ 平均时效性大约需 29 s.本图没有给出 build currency graphs 的执行时间,这是因为本实验中随机生成的约束仅对一对记录生效,且可满足性判定只需常数时间,故 *tupleNum* 不影响 build currency graphs 的执行时间.事实上,*tupleNum*=

10 k 时 build currency graphs 也只需约 10 ms，和其它两个过程相比几乎可以忽略不计.

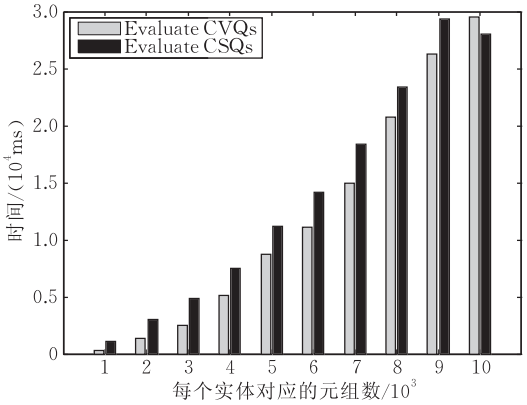


图 7　参数 *tupleNum* 对算法执行效率的影响
(*attrNum*=*entityNum*=*constraintNum*=10)

图 8 所示为 *attrNum* 对算法执行效率的影响. 本组实验中 *entityNum*、*tupleNum* 和 *constraintNum* 都等于 10. 算法 2 和算法 3 的执行时间随 *attrNum* 增长基本呈线性增长，而 build currency graphs 因为要对每个实体在每个属性上构建时效图，故其运行时间也随 *attrNum* 的增长呈线性增长. 在图中可以看出，*attrNum*=10 k 时，最多只需 0.12 s 就可以完成 3 个过程中的任何一个，算法执行效率较高.

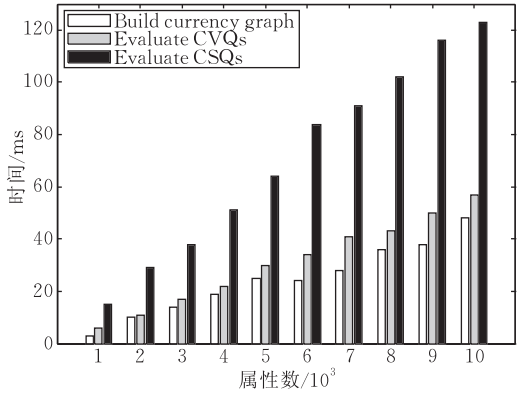


图 8　参数 *attrNum* 对算法执行效率的影响
(*tupleNum*=*tupleNum*=*constraintNum*=10)

图 9 所示为 *constraintNum* 对算法执行效率的影响. 本组实验中 *entityNum*、*tupleNum* 和 *attrNum* 都等于 10. 约束数目影响时效图的构建，而 build currency graphs 是数据预处理过程，如图 9 所示，其执行时间随 *constraintNum* 的增长呈线性增长. 算法 2 和算法 4 中不构建时效图，所以 evaluate CVQs 和 evaluate CSQs 的执行时间只随随机生成的数据集合不同而略有波动，与 *constraintNum* 无关.

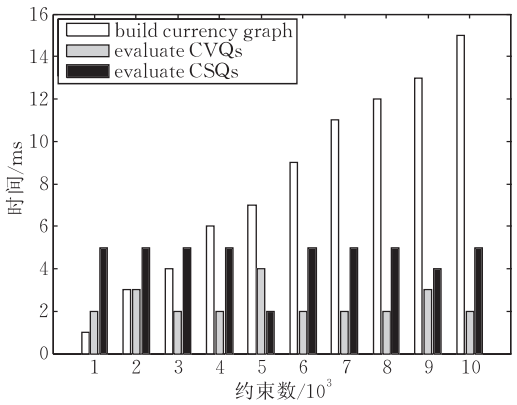


图 9　参数 *constraintNum* 对算法执行效率的影响
(*attrNum*=*tupleNum*=*entityNum*=10)

图 10 所示为 *queryNum* 对第 3 类用户相关时效性判定的执行时间的影响. 该组实验中 *entityNum*、*tupleNum* 和 *constraintNum* 都等于 10，*attrNum* 等于 1000(查询较多时，涉及到的属性也相应的会变多，因而 *attrNum* 设置的较大). 由于除 *queryNum* 外其它参数都是常数，所以算法执行时间随 *queryNum* 的增长呈线性增长. *queryNum*=10 k 时判定第 3 类用户相关时效性只需约 0.46 s，算法执行效率较高.

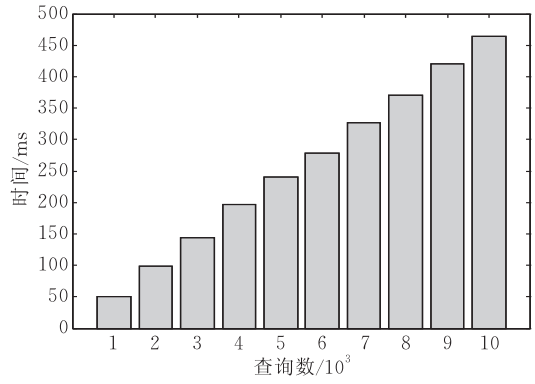


图 10　参数 *queryNum* 对算法执行效率的影响
(*attrNum*=1000, *tupleNum*=*entityNum*=*rNum*=10)

6　结论及未来研究方向

本文研究包含冗余记录的集合在给定时效约束下的时效性判定问题，并给出了查询相关时效性判定问题和用户相关时效性判定问题的定义及算法. 在判定查询相关时效性时，将查询分为 CVQ 和 CSQ 两类，分别给出了这两类查询的结果时效性和平均时效性判定问题的算法. 在判定用户相关时效性时，将用户分为 3 类，针对每类用户的需求，提出了不同的用户相关时效性判定方法，并进一步讨论了如何根据用户相关时效性判定结果来指明数据修复的方向. 在研究上述时效性判定问题的过程中，本文提出

了时效图的概念,将时序关系以图的形式建模,使之能够帮助时效性的判定.最后用实验分析了各个参数对算法的影响,并验证了本文算法的高效性.

在本文工作的基础之上,未来我们将着力于研究下述问题:(1)如何在数据集合动态变化时准确地判定数据时效性;(2)如何在数据不完整或有错误时判定数据时效性;(3)在数据时效性不佳时,如何自动修复数据.

参 考 文 献

[1] Eckerson W W. Data quality and the bottom line: Achieving business success through a commitment to high quality data. Data Warehousing Institute: Technical Report TDWI Report Series, 2002

[2] Zhang H, Diao Y, Immerman N. Recognizing patterns in streams with imprecise timestamps. Proceedings of the VLDB Endowment, 2010, 3(1-2): 244-255

[3] Fan W, Geerts F, Wijsen J. Determining the currency of data//Proceedings of the ACM Symposium on Principles of Database Systems (PODS). Athens, Greece, 2011: 71-82

[4] Berti-Equille L, Sarma A D, Dong X, Marian A, Srivastava D. Sailing the information ocean with awareness of currents: Discovery and application of source dependence//Proceedings of the Conference on Innovative Data Systems Research (CIDR). Asilomar, CA, USA, 2009

[5] Dong X, Berti-Equille L, Hu Y, Srivastava D. Global detection of complex copying relationships between sources. Proceedings of the VLDB Endowment, 2010, 3(1-2): 1358-1369

[6] Dong X, Berti-Equille L, Srivastava D. Truth discovery and copying detection in a dynamic world. Proceedings of the VLDB Endowment, 2009, 2(1): 562-573

[7] Clifford J, Dyreson C E, Isakowitz T, Jensen C S, Snodgrass R T. On the semantics of "now" in databases. ACM Transactions on Database Systems (TODS), 1997, 22(2): 171-214

[8] Snodgrass R T, Gao D, Zhang R, Thomas S W. Temporal support for persistent stored modules//Proceedings of the IEEE International Conference on Data Engineering (ICDE). Washington, DC, USA, 2012

[9] Bodirsky M, Kara J. The complexity of temporal constraint satisfaction problems//Proceedings of the 40th Annual ACM Symposium on Theory of Computing. Victoria, British Columbia, Canada, 2008: 29-38

[10] Elmagarmid A K, Ipeirotis P G, Verykios V S. Duplicate record detection: A survey. Transactions on Knowledge and Data Engineering (TKDE), 2007, 19(1): 1-16

LI Mo-Han, born in 1987, Ph. D. candidate. Her research interests focus on data quality.

LI Jian-Zhong, born in 1950, professor, Ph. D. supervisor. His research interests include database, massive data processing, wireless sensor networks, cyber-physical systems etc.

GAO Hong, born in 1966, professor, Ph.D. supervisor. Her research interests include wireless sensor networks, cyber-physical systems, massive data management and data mining etc.

Background

The problem of data currency is one of the most important issues in the area of data quality. The data with poor currency can badly influence the business decision and people's daily life. For example, a company will make wrong decision because of the stale data. Also, bank may send the credit card bill to someone's old address.

That highlights the needs of the evaluation of data currency. A big challenge of data currency evaluation is absence of valid timestamps. However, redundant records and currency constraints can recover the currency orders of data without using timestamps thus can be helpful when evaluating data currency. Some previous work has focused on how to model partial currency orders in terms of simple constraints, and studied the complexities of a lot of fundamental decision problems. But, little work can give algorithms to evaluate the data currency.

This paper investigates the algorithms of data currency evaluation using currency constraints. The methods can be described from three aspects. First, this paper defines data currency relative to queries and data currency relative to users. When evaluating data currency relative to queries, all the queries are classified as 2 categories, which are Current Value Query and Currency Sequence Query. For each query category, this paper discusses the definition of the currency of query result and the average currency of the entire query category. Second, the definition of currency graph is proposed in this paper. The methods of evaluating data currency relative to queries and users using currency graphs are presents. Third, experimental results on real and synthetic datasets are given to analyze the effect of parameters and the efficiency of algorithms.

This research is supported by the National Basic Research Program (973 Program) of China under Grant No. 2012CB316202.

Our group has been focusing on the research of data management for a long time. Many outstanding works have been published in worldwide conferences and transactions, such as SIGMOD, VLDB, ICDE, KDD, INFOCOM, TKDE, VLDB Journal et al. This paper proposes a novel way on evaluation of data currency. It gives solutions on evaluating the data currency relative to queries and users respectively.