条件依赖理论及其应用展望

胡艳丽 张维明

(国防科学技术大学 C^4ISR 技术国防科技重点实验室 长沙 410073)

摘 要 介绍了条件函数依赖理论及如何用于检测不一致数据。首先介绍了条件函数依赖的概念及其推理系统,以及如何通过依赖传播实现视图的规范化;阐述了条件函数依赖的一致性和蕴含判定问题,并在此基础上介绍了基于条件函数依赖检测关系数据库数据一致性的技术;最后讨论了条件函数依赖的扩展及应用。

关键词 数据质量,数据清洗,条件函数依赖,推理规则,依赖传播,一致性判定,蕴含判定

中图法分类号 TP311.131 **文献标识码** A

Theory of Conditional Functional Dependencies and its Application for Improving Data Quality

HU Yan-li ZHANG Wei-ming

(C^4ISR Technology National Defense Science and Technology Key LAB, National University of Defense Technology, Changsha 410073, China)

Abstract The theory of conditional functional dependencies was introduced and its application for improving data quality was discussed. Firstly, conditional functional dependencies (CFDs) were defined and a set of sound and complete inference rules for CFDs were proposed; and dependency propagation was studied to find CFDs holding on views. Secondly, the consistency and implication problems for CFDs were analyzed. Then SQL techniques were proposed to detect data inconsistency in relational database. Finally, extensions of CFDs were discussed.

Keywords Data quality, Data cleaning, Conditional functional dependencies, Inference rule, Dependency propagation, Consistency, Implication

数据管理是企业信息化的基础和核心环节,正确的决策依赖于可靠、准确的数据,因此如何提高数据质量(data quality)一直是学术界和工业界孜孜不倦研究的课题。本文主要从数据集成(data integration)和数据清洗(data cleaning)的角度探讨数据质量管理。

数据集成技术融合来自不同数据源的数据,通过建立统一的数据视图向用户提供透明操作异构数据的能力[1]。在数据集成的研究中,很多工作的重点放在如何解决模式冲突上。其实,在数据实例层次上同样有很多数据质量问题发生[2]。数据清洗的目的就是要解决"脏数据(dirty data)"问题,检测并纠正数据中存在的错误和不一致,从而提高数据质量[3]。虽然研究人员对数据清洗技术开展了大量研究,提出了多种解决相似重复记录(approximate duplicates)[4-7]、结构冲突(structural conflicts)[8]的方法,但针对实际应用依然缺乏行之有效的系统解决方案。大量的数据清洗工作依然通过人工或底层的应用程序完成,工业界开发的很多数据抽取、转化和装载工具(ETL, extraction, transformation, loading)普遍缺乏有效的数据清洗能力。

1 研究背景

数据依赖(data dependency)定义了关系数据库属性间的语义关系,是现实世界数据间内在联系在数据库中的抽象反映。作为数据库重要理论之一,数据依赖理论在数据库规范化设计、模式集成、查询优化、数据库更新等方面发挥了举足轻重的作用[9,10]。

然而,良好的规范化设计依然无法保证所存放数据的质量,例如数据库中存在不破坏完整性约束和数据依赖的脏数据。数据集成的过程中,数据来自多个异构数据源,已有的脏数据会传播、累加到目标数据中,使这个问题更加复杂。

近年来研究人员开始关注将约束用于数据清洗,基于传统的数据依赖理论(如函数依赖)指定数据间的语义关系,不一致的数据以破坏依赖的形式被检测出来,通过约束修复(constraint repair)技术提高数据质量[11-18]。但传统的数据依赖只涉及关系模式层次的信息,面对数据实例层次的数据质量问题往往无能为力。下面的示例充分说明了这个问题。

考虑某跨国公司的客户信息数据[19]。为简化问题不妨假设该数据库主要通过关系(CC, AC, PN, NM, STR, CT, ZIP)存储客户信息(如表1所列),其中CC, AC, PN, NM, STR, CT, ZIP分别表示国家代码、地区代码、电话号码、姓名、街道、所在城市、邮政编码。定义该关系满足的函数依赖为:

f_1:[CC, AC, PN]→[STR, CT, ZIP]

f_2:[CC, AC]→[CT]

其中,f_1 定义具有相同国家代码、地区代码和电话号码字段的客户记录也具有相同的街道、城市和邮政编码字段,f_2 定

到稿日期:2009-01-20 返修日期:2009-05-13 本文受国家自然科学基金(70701038和70771109)资助。

胡艳丽(1979—),女,博士研究生,主要研究领域为信息管理、数据库理论与应用技术,E-mail:smilelife1979@163.com;张维明(1962—),男,教授,博士生导师,主要研究领域为信息管理、智能决策。

义具有相同国家代码和地区代码字段的客户记录也具有相同的城市名称字段。如果依据上述函数依赖检测表1所含的数据，会得出数据"一致"的结论。然而事实上，当国家代码是"44"(英国)时，ZIP决定了STR，即在英国不同的街道具有不同的邮政编码。显然表1的最后3条记录 t_3, t_4, t_5 不满足这条约束，因此存在错误，但却无法被传统的数据依赖检测出来。

表1 实例层次数据不一致性示例[19]

CC	AC	PN	NM	STR	CT	ZIP
01	908	11111111	Mike	Tree Ave.	NYC	07974
01	212	2222222	Joe	Elm Str.	NYC	01202
44	131	3333333	Ian	High St.	EDI	EH4 1DT
44	131	4444444	Rick	Princess St.	EDI	EH4 1DT
44	131	5555555	Ben	Royal Mile	EDI	EH4 1DT

上述约束可以表示为下列形式：

Φ_0:[CC=44,ZIP]→[STR]

Φ_0 定义当"CC=44"时，ZIP决定STR，"CC=44"构成了这条约束成立的条件。易见，Φ_0 适用于英国的客户数据，对检测这部分数据子集的一致性非常有效。

事实上，这样的约束在现实生活中普遍存在。如某些企业中员工的职位决定了薪资，而另外一些企业并非如此；再如某些国家公民的收入决定了他的个人所得税率，但其他一些国家则有不同的制度。这类约束对于定义、检测数据实例层次上的一致性非常有效，但因为约束中包含了特定的数据而无法通过传统的数据依赖表达。针对这一问题，樊文飞教授提出了条件依赖理论[19-24]，它通过绑定关系属性及其语义相关的数据，定义数据子集上成立的约束，用于细粒度的数据不一致性检测，并将条件依赖理论用于数据清洗。国内对数据清洗的研究主要针对数据清洗框架的设计和实现[25,26]，基于数据依赖的清洗技术尚不多见。本文从提高数据质量角度阐述条件依赖理论的研究工作，以促进研究人员对这一方向的关注和研究。

2 条件函数依赖理论

数据清洗的关键问题之一是如何对数据一致性进行建模，即如何定义数据的一致性。为了检测数据实例层次的不一致性，条件依赖理论绑定关系中属性及其语义相关的数据，定义了满足约束的正确数据的模式。实例中存在的错误或不一致数据通过破坏约束的形式被检测出来，从而应用条件函数依赖指导数据清洗。

2.1 基本概念

定义1(条件函数依赖)[19] 关系 R 上成立的条件函数依赖(CFD, Conditional Functional Dependency)Φ 形式为 $(R: X\to Y, T_p)$，其中：

(1) $X, Y\in attr(R)$，$attr(R)$ 表示 R 的属性集合；

(2) $X\to Y$ 为嵌入 Φ 的函数依赖；

(3) T_p 为定义在属性集 $X\cup Y$ 上的模式组(pattern tableau)，由若干条模式元组 t_p 构成。对于 $X\cup Y$ 中的每个属性 A，模式元组 t_p 定义了 A 的取值 $t_p[A]$ 为定义域 $Dom(A)$ 中的某个常数 a，或者任意值，用'_'表示。

由定义可见，T_p 通过绑定属性 A 及其取值 $t_p[A]$ 指定了嵌入式函数依赖成立的条件，如示例中 Φ_0 表示的条件函数依赖为 $(R:[CC, ZIP]\to STR, (44, \_\|\_))$。

易见，函数依赖是条件函数依赖的特例，即函数依赖是 t_p 中所有属性取值均为'_'的条件函数依赖。为简单起见，下文均用 $(R: X\to A, t_p)$ 表示条件函数依赖，因为 $(R: X\to Y, T_p)$ 可以在线性时间内转化成一组形式为 $(R: X\to A, t_p)$ 的等价的条件函数依赖，其中 $A\in Y$ 且 $t_p\in T_p$。如果属性 A 同时出现在条件函数依赖的左部和右部，则分别用 A_L 和 A_R 表示。

定义2(匹配)[19] 对于属性 A 的取值 η_1, η_2，η_1 与 η_2 匹配是指 $\eta_1=\eta_2$ 或者 η_1, η_2 中取值存在'_'，用 $\eta_1\asymp\eta_2$ 表示。

定义3(序关系)[19] 对于上述 η_1, η_2，如果 $\eta_1=\eta_2=$'a' 或者 $\eta_2=$'_'，则 $\eta_1\preceq\eta_2$。

匹配是对函数依赖中属性取值相等的扩展，主要用于判断实例与模式元组或者模式元组之间相同的属性取值是否存在对应关系。序关系进一步细化了属性取值间的关系，可以扩展到元组之间，如(44, $EH4\ 1HD$)$\preceq$(44, _)。

定义4(条件函数依赖语义)[19] 对于定义在关系 R 上的条件函数依赖 $\Phi=(R: X\to A, t_p)$，如果实例 I 中任意两个元组 t_1, t_2 都满足以下条件：$t_1[X]=t_2[X]\asymp t_p[X]$，则 $t_1[A]=t_2[A]\asymp t_p[A]$，那么称 I 满足 Φ，记作 $I\vdash\Phi$。

究其本质，条件函数依赖 Φ 的模式元组 t_p 指定了满足 Φ 的数据元组 t_1, t_2 的形式，即若 $t_1[X]=t_2[X]$，则(a) $t_1[A]=t_2[A]$，且(b) $t_1[A]\asymp t_p[A]$。其中，(a)执行了嵌入式函数依赖的语义，(b)保证数据元组与模式元组取值一致，从而满足与属性绑定的语义相关数据。约束由于在满足模式元组所定义条件的数据子集上成立，因此成为条件函数依赖。

2.2 推理规则集

如何实现一类数据依赖的公理化是数据依赖理论的一个重要研究课题，有效的公理系统可以通过可靠且完备的推理规则(inference rules)支持逻辑蕴含(logical implication)的符号证明，对于揭示数据依赖的实质、提出有效的逻辑蕴含算法具有重要意义。

Armstrong公理系统就是对函数依赖的公理化，是证明函数依赖之间存在蕴含关系、计算函数闭包的重要依据。但并非所有的数据依赖都能公理化，如连接依赖(join dependencies)就没有对应的公理系统。

定理1[19] 条件函数依赖存在可靠且完备的推理系统 $\mathcal{I}$。

条件函数依赖的推理系统 $\mathcal{I}$ 是对Armstrong公理系统的扩展，包括以下推理规则。

FD1：若 X 为一属性集且 $A\in X$，则有 $(R: X\to A, t_p)$，其中 $t_p[A_L]=t_p[A_R]=$'a'，$a\in Dom(A)$，或者 $t_p[A_L]=t_p[A_R]=$'_'。

FD2：若有(1) $\forall i, j\in[1,k]$，$(R: X\to A_i, t_i)$ 满足 $t_i[X]=t_j[X]$；(2) $(R: [A_1, \cdots, A_k]\to B, t_p)$；且(3) $(t_1[A_1], \cdots, t_k[A_k])\preceq(t_p[A_1, \cdots, A_k])$，则有 $(R: X\to B, t_p')$，其中 $t_p'[X]=t_1[X]$ 并且 $t_p'[B]=t_p[B]$。

FD3：若有 $(R: [X, B]\to A, t_p)$，其中 $t_p[B]=$'_'，$t_p[A]$ 为常数，则有 $(R: X\to A, t_p')$，其中 $t_p'[X\cup\{A\}]=t_p[X\cup\{A\}]$。

FD4：若有(1) $\forall i\in[1,k]$，$\Sigma\vdash_I(R: [X, B]\to A, t_i)$，(2) $Dom(B)=\{b_1, \cdots, b_k, b_{k+1}, \cdots, b_m\}$，且仅当 $l\in[1,k]$ 时 $(\Sigma, B=b_l)$ 可满足，(3) $\forall i, j\in[1,k]$，$t_i[X]=t_j[X]$，$t_i[B]=b_i$，则有 $\Sigma\vdash_I(R: [X, B]\to A, t_p)$，其中 $t_p[B]=$'_'，$t_p[X\cup\{A\}]=t_1$

$[X \cup \{A\}]$。

FD1 和 FD2 分别是对 Armstrong 公理系统中自反律(reflexivity)和传递律(transitivity)的扩展。FD1 定义了平凡的条件函数依赖;FD2 规定了如果一个条件函数依赖的右部属性包含于另一条件函数依赖的左部属性集合,且该属性在前者模式元组中的取值与在后者中的取值存在序关系时可以应用传递律;FD3 说明当一个条件函数依赖的右部属性取值为常数时,如果其左部存在某个属性取值为任意值,即无论该属性取什么值右部属性取值均为特定常数,那么左部这个属性对于这条依赖是冗余的,可以省略,从而为简化条件函数依赖提供了依据;FD4 说明如果条件函数依赖的某个属性取值为有限域(finite domain),且模式元组中该属性在有限域中的每个取值对应的条件函数依赖都被一组条件函数依赖 Σ 蕴含,那么该属性在模式元组中取值为任意值'_'时对应的条件函数依赖依然被 Σ 蕴含,从而避免了枚举该属性的所有取值。

2.3 条件函数依赖传播

视图(View)是根据用户需要及数据库查询结果对数据库进行访问、操作的基本方式,对实现数据独立性、简化查询、维护数据安全等具有重要意义。给定数据源上成立的依赖以及视图定义,依赖传播(dependency propagation problem)研究数据源依赖如何通过视图定义生成目标数据应满足的依赖。其核心意义在于规范化的传播[27]。

定义 5(依赖传播问题)[19] 给定数据源满足的依赖集合 Σ,视图 V 和 V 上的依赖 Φ,依赖传播判断 Φ 是否是 Σ 传播到 V 上的依赖,记作 $\Sigma \vdash_V \Phi$,即对于数据源对应的任意实例 I,如果 $I \vdash \Sigma$,则 $V(I) \vdash \Phi$。

依赖传播问题的复杂度与视图定义语言相关。合取查询(conjunctive query)是一类基本的查询语言,是其他复杂查询语言的基础,因此这里主要讨论合取查询。它包含选择(selection,S)、投影(projection,P)和笛卡尔积(cross-product,C)操作,简写为 SPC 查询,其标准形式为 $\pi_Y(\sigma_F(R_1 \times \cdots \times R_n))$,其中 π 表示投影操作,Y 指定投影的属性集合,σ 表示选择操作,F 构成选择的条件,$R_1 \times \cdots \times R_n$ 表示 n 个关系的笛卡尔乘积,不妨设 $E_s = \sigma_F(R_1 \times \cdots \times R_n)$。

定理 2[28] 如果 Φ_i 是数据源 R_i 满足的约束,且 $R_i \in (R_1, \cdots, R_n)$,那么 Φ_i 在视图 E_s 上成立,记作 $\Sigma \vdash_{E_s} \rho(\Phi_i)$,其中 ρ 是对 $(R_1 \times \cdots \times R_n)$ 所含属性的重命名。

易于理解,如果只有笛卡尔乘积和选择操作,数据源满足的数据依赖依然在视图上成立。

定理 3[28]

(1)如果 $A=B \in F$,那么其所对应的条件函数依赖为 $\Phi=(A \to B, (x \mid x))$;

(2)如果 $A='a' \in F$,那么其所对应的条件函数依赖为 $\Phi=(A \to A, (\_ \mid a))$。

定理 3 说明了选择操作的条件所生成的依赖的形式。易见,条件 $A=B$ 表示视图中任意元组 t 都满足 A 属性等于属性 B,即如果 $t[A]=x$,那么必然有 $t[B]=x$,即形式(1);条件 $A='a'$ 表示视图中任意元组 t 都满足 $t[A]=a$,即形式(2)。

定理 4[28] 给定数据源满足的依赖集合 Σ,视图 V 和 V 上的依赖 Φ,如果 Φ 是下列形式之一:

(1)$\Phi=(A \to B, (x \mid x))$,表示 $A=B$;

(2)$\Phi=(A \to A, (\_ \mid a))$,表示 $A='a'$;或者

(3)$\Phi=(X \to A, (\_ \mid a))$。

其中,$A \in Y$,$B \in Y$ 且 $X \subseteq Y$,那么 $\Sigma \vdash_V \Phi$ 当且仅当 $\Sigma \vdash_{E_s} \Phi$。

定理 4 表明投影操作要求视图上成立的约束 Φ 所涉及的属性必须包含在投影的属性集合 Y 内;在此基础上,Φ 是 Σ 传播到 V 上的依赖当且仅当 Φ 在笛卡尔积和选择操作定义的视图上成立。

3 性质分析

要利用条件函数依赖进行数据清洗,首先必须保证给定的一组依赖本身是一致的,即彼此之间不互相矛盾。在此基础上为了实现有效的数据清洗,可以通过蕴含分析从条件依赖集合中排除冗余的依赖,从而提高数据清洗的效率。这就涉及数据依赖理论的两个基本问题,即数据依赖的一致性和蕴含判定问题。

3.1 一致性问题

定义 6(一致性判定问题)[19] 数据依赖的一致性判定问题(the consistency problem)是指给定一组数据依赖,判断是否存在满足该集合的非空数据集。

相应地,条件函数依赖的一致性判定问题是指给定一组条件函数依赖,判断是否存在满足该集合的非空数据集。

如果一组条件函数依赖是一致的,彼此不存在矛盾,那么一定存在满足该集合的非空数据集。虽然一组函数依赖总是一致的,但条件函数依赖通过模式元组绑定了语义相关的数据,而模式元组指定的属性取值可能存在矛盾,从而导致可能不存在满足约束的实例,因此其一致性判定问题是非平凡的。

定理 5[19] 条件函数依赖的一致性判定是 NP-complete 问题。

分析可以发现,条件函数依赖的不一致只会由右部属性取值为常数的那些条件函数依赖产生,如 $(R:B \to A, (\_, a))$,$(R:B \to A, (\_, b))$ 是不一致的,因为一个属性的取值不可能同时既是 a 又是 b,所以一组条件函数依赖的一致性等价于它所包含的右部属性取值为常数的条件函数依赖子集的一致性。

定理 6[19] 对于已知、确定的关系,条件函数依赖的一致性判定是 PTIME 问题。

3.2 蕴含问题

定义 7(蕴含判定问题) 数据依赖的蕴含判定问题(the implication problem)是指给定一组数据依赖 Σ 和一条数据依赖 Φ,判断集合 Σ 是否蕴含 Φ,记作 $\Sigma \vdash \Phi$,即对于所有的实例 I,如果 $I \vdash \Sigma$,那么 $I \vdash \Phi$。

相应地,条件函数依赖的蕴含判定问题是指给定一组条件函数依赖和一个条件函数依赖,判断该集合是否蕴含该条件函数依赖。

定理 7[19] 条件函数依赖的蕴含判定问题是 coNP-complete 的。

定理 8[19] 对于已知、确定的关系,条件函数依赖的蕴含判定是 PTIME 问题。

虽然普遍情况下的条件函数依赖蕴含判定是难解的,但具体应用中数据库通常是确定的,此时条件函数依赖的一致性和蕴含是多项式时间可判定的。

定义 8(条件函数依赖的极小覆盖)[19] 给定关系 R 上成立的一组条件函数依赖 Σ,Σ 的极小覆盖(minimal cover)

Σ_{mc} 满足以下条件：

(1)$\Sigma_{mc} \equiv \Sigma$；

(2)$\nexists\ \exists \Sigma'_{mc} \subset \Sigma_{mc}$ 且 $\Sigma'_{mc} \vdash \Sigma_{mc}$；

(3)对于 Σ_{mc} 中任意条件函数依赖$(R: X \rightarrow A, t_p)$，$\Sigma_{mc}$ 中都不存在形式为$(R: X' \rightarrow A, t_p[X'A])$的条件函数依赖使得 $X' \subset X$。

由定义可见，一组条件函数依赖 Σ 的极小覆盖 Σ_{mc} 必须与 Σ 等价，不存在 Σ_{mc} 的真子集蕴含 Σ_{mc}，且 Σ_{mc} 包含的每个条件函数依赖都是最简化的，不存在冗余的属性。

实际应用中可以通过蕴含分析，首先删除每个条件函数依赖的冗余属性，进而删除依赖集合所包含的冗余依赖，计算得到一组条件函数依赖的极小覆盖，用于提高数据清洗的效率。

4 条件依赖理论在数据清洗中的应用

数据清洗首先需要检测不一致数据，给定关系 R 的实例 I 和 R 上成立的一组条件函数依赖 Σ，需要检测 I 中不满足 Σ 所包含条件函数依赖的不一致数据记录。因为条件函数依赖允许绑定属性的特定取值，所以数据库上成立的条件函数依赖数目可能比较庞大，需要有效的技术实现基于条件函数依赖的数据不一致性检测。下面主要通过数据操作语言检测破坏条件函数依赖的不一致数据。

4.1 基于条件函数依赖检测不一致数据

对于集合 Σ 中的任意条件函数依赖 $\Phi=(R: X \rightarrow A, T_p)$，通过下列查询检测实例 I 是否破坏 Φ[19]：

(1)当 Φ 的右部属性取值为常数时，直接查询 I 中是否存在元组 t，与模式元组 t_p 的左部匹配但右部却不匹配，这样的元组破坏 Φ；

(2)当 Φ 的右部属性取值为'_'时，查询 I 中是否存在对于 X 属性取值相等且与 Φ 模式元组左部匹配，但右部却不相等的元组，即如果 I 中存在多条 Φ 的相关元组具有相同的左部属性取值但右部属性取值却不同，那么 A 的不同取值数必然大于 1，这些元组同样破坏了 Φ。

4.2 增量式检测

数据库更新，如插入新的记录或删除已有记录，会引起 I 发生变化，从而使得数据的一致性发生变化，此时需要判断更新后数据集的一致性。

虽然采用 4.1 节的方法可以检测更新后的实例是否一致，但由于可能会重复大量冗余检测从而导致不必要的计算开销。因此采用下面的增量式方法检测更新导致的不一致[19]。

4.2.1 记录不一致信息

对关系 R 进行扩展，对于每个条件函数依赖 Φ，增设记录每条元组破坏条件函数依赖 Φ 情况的标记字段，取值为布尔变量。如果数据元组不满足 Φ，那么将该元组的字段设置为 1，否则为 0。

4.2.2 处理删除更新

当从数据库删除某条记录 t 时，更新及一致性检测分为两步[19]：首先删除 t，然后检测实例中 t 匹配的条件依赖$(R: X \rightarrow A, T_p)$相关数据的一致性是否发生变化，如果更新后属性 A 取值均相同，这时将字段置为 0。

4.2.3 处理插入更新

当向数据库插入新的记录 t 时，首先需要检测该记录是否破坏了条件函数依赖，其次需要检测 t 的插入是否引起已有数据一致性的变化。此时数据库更新及一致性检测分为 3 步[19]：首先将元组 t 插入关系，初始化标记字段均为 0；其次检测是否存在条件函数依赖 Φ，使得 t 与 Φ 的左部属性取值匹配但却不满足其模式元组右部属性的取值，如果存在则设置标记字段为 1；最后检测 t 与数据库中已有元组间是否存在不一致，即检测数据库中是否存在 t' 与 t 对于 Φ 左部属性取值相等且与 Φ 匹配，但相应 Φ 的右部属性取值却不同，如果存在则将 t' 与 t 的标记字段均设置为 1。

上述操作为条件依赖理论在数据清洗中的应用提供了实现方法。

结束语 条件函数依赖通过对数据一致性的形式化建模，定义了应用相关的正确数据的模式，可以检测数据元组粒度的不一致性。条件函数依赖体现了特定应用数据间的内在语义关联，通过应用相关的语义约束检测实例层次上的不一致数据。一致性检测和求解极小覆盖等技术有助于提高大规模数据清洗的效率，为自动实现数据不一致性检测、减少清洗过程的人工干预提供了基础。

条件函数依赖及其应用研究正在深入展开[29-31]，如为了表达关系属性取值存在多种情况(disjunction)或不等于某些取值(inequality)等更丰富的语义，提出了扩展的条件函数依赖(eCFDs, extended Conditional Functional Dependencies)，而且表达能力的增加并未导致复杂度的增加。通过系统、深入的研究，条件函数依赖逐渐发展成为完整的理论体系，将在数据清洗及提高数据质量的研究中发挥重要作用。

参 考 文 献

[1] Lenzerini M. Data Integration: A Theoretical Perspective[C]// pods'02. 2002

[2] 郭志懋，周傲英. 数据质量和数据清洗研究综述[J]. 软件学报，2002，13(11)：2076-2082

[3] Rahm E, Do H H. Data cleaning: problems and current approaches[J]. IEEE Data Engineering Bulletin, 2000, 23(4): 3-13

[4] Winkler W E. Advanced methods for record linkage[M]. Statistical Research Division, U. S. Bureau of the Census, 1994

[5] Hernandez M A, Stolfo S. Real-world data is dirty: Data cleansing and the merge/purge problem[J]. Data Min. Knowl. Discov., 1998, 2(1): 9-37

[6] Galhardas H, Florescu D, Shasha D, et al. AJAX: An extensible data cleaning tool[C]// Proceedings of the International Conference on Management of Data (SIGMOD). 2000

[7] Monge A E. Matching algorithms within a duplicate detection system[J]. IEEE Data Eng. Bull., 2000, 23(4): 14-20

[8] Raman V A H J M. Potter's wheel: An interactive data cleaning system[C]// Proceedings of the International Conference on Very Large Databases (VLDB). 2001

[9] Silberschatz A, Korth H F. Database System Concepts. McGraw-Hill, 1986

[10] Ullman J D. Principles of Database Systems[M]. Computer Science Press, 1982

[11] Arenas M, Bertossi L E, Chomicki J. Consistent query answers in inconsistent databases[J]. Theory Pract. Logic Program, 2003, 3(4/5): 393-424

(下转第 145 页)

与此同时，需要看到的是，混合模型需要两种模型作为基础。因此，在模型应用时，必须同时以满足两种模型的应用条件为前提。例如，由于一般中小型电子商务购物网站很难获得用户的偏好信息，也不易得到用户对商品的评价，这就不能很好地应用本混合模型。这也是本模型以后需要继续完善和改进的地方。

参 考 文 献

[1] Ben Schafer J, Konstan J, Riedl J. Recommender systems in e-commerce[C]//Proc. of the 1st ACM Conf. on Electronic Commerce. New York: ACM press, 1999: 158-166

[2] Riedl J, Dourish P. Introduction to the Special Section on Recommender Systems[J]. ACM Transactions on Computer-Human Interaction, 2005, 12: 371-373

[3] Ricci F, Werthner H. Introduction to the special issue: Recommender systems[J]. International Journal of Electronic Commerce, 2006, 1(2): 5-9

[4] Balabanovic M, Shoham Y. Content-based, collaborative recommendation[J]. Communications of the ACM, 1997, 40(3): 66-72

[5] Terveen L, Hill W. Beyond recommender systems: Helping people help each other[C]//Carroll J, ed. HCI in The New Millennium. New York: Addison-Wesley Publishing Co., 2001: 1-21

[6] Han Peng, Xie Bo, Yang Fan. A Scalable P 2 P Recommender System Based on Distributed Collaborative Filtering[J]. Expert Systems with Applications, 2004, 27 (2): 203-210

[7] Min Sung-hwan, Han Ingoo. Detection of the Customer Time-variant Pattern for Improving Recommender Systems[J]. Expert Systems with Applications, 2005, 28 (2): 189-199

[8] 许海玲，吴潇，李晓东，等. 互联网推荐系统比较[J]. 软件学报，2009，20(2)：350-362

[9] Lazcorreta E, Botella F, Fernández-Caballero A. Towards personalized recommendation by two-step modified Apriori data mining algorithm[J]. Expert Systems with Applications, 2008, 35(3): 1422-1429

[10] Cao Yu-kun, Li Yun-feng. An Intelligent Fuzzy2Based Recommendation System for Consumer Electronic Products[J]. Expert Systems with Applications, 2007, 33 (1): 230-240

[11] Liu Duen-ren, Lai Chin-hui, Huang Chiu-wen. Document Recommendation for Knowledge Sharing in Personal Folder Environments[J]. Journal of Systems and Software, 2008, 81(8): 1377-1388

[12] 余力，刘鲁. 电子商务个性化推荐研究[J]. 计算机集成制造系统，2004，10(10)：1306-1313

[13] Yu Li, Liu Lu, LI Xue-feng. A hybrid collaborative filtering method for multiple-interests and multiple-content recommendation in E-Commerce[J]. Expert Systems with Applications, 2005, 28 (1): 67-77

[14] 黎星星，黄小琴，朱庆生. 电子商务推荐系统研究[J]. 计算机工程与科学，2004，26(5)：7-10

[15] Basu C, Hirsh H, Cohen W. et al. Recommendation as classification: Using social and content-based information in recommendation[C]//Proceedings of the 1998 Workshop on Recommender Systems. 1998: 43-52

[16] Ansari A, Essegaier S, Kohli R. Internet Recommendation Systems[J]. Journal of Marketing Research, 2000, 37: 363-375

[17] http://www.grouplens.org/node/73

[18] 周建硕. 供应链信息共享与利用的相关模式[J]. 重庆工学院学报：自然科学版，2009，23(2)：118-121

(上接第 118 页)

[12] Franconi E, Palma A L, Leone N, et al. Census data repair: a challenging application of disjunctive logic programming[C]// Proceedings of the Artificial Intelligence on Logic for Programming (LPAR). 2001

[13] Bravo L A B L. Logic programs for consistently querying data integration systems[C]//Proceedings of the International Joint Conference on Artificial Intelligence. 2003

[14] Cali A, Lembo D, Rosati R. On the decidability and complexity of query answering over inconsistent and incomplete databases [C]//Proceedings of the Symposium on Principles of Database Systems (PODS). 2003

[15] Cali A, Lembo D, Rosati R. Query rewriting and answering under constraints in data integration systems[C]//Proceedings of the International Joint Conference on Artificial Intelligence. 2003

[16] Chomicki J A M. Minimal-change integrity maintenance using tuple deletions[J]. Inform. Comput., 2005, 197(1/2): 90-121

[17] Greco G, Greco S, Zumpano E. A logical framework for querying and repairing inconsistent databases[J]. IEEE Trans. Knowl. Data Engin., 2003, 15(6): 1389-1408

[18] Wijsen J. Database repairing using updates[J]. ACM Trans. Datab. Syst., 2005, 30(3): 722-768

[19] Fan Wenfei, Jia Xibei. Anastasios Kementsietsidis, Conditional Functional Dependencies for Capturing Data Inconsistencies[J]. ACM Transactions on Database Systems (TODS), 2008, 33(2)

[20] Fan W. Dependencies Revisited for Improving Data Quality[C]//ACM Symposium on Principles of Database Systems (PODS) (invited). 2008

[21] Fan Wenfei. A Revival of Integrity Constraints for Data Cleaning[C]// The 34th International Conference on Very Large Data Bases (VLDB). tutorial, 2008

[22] Fan W. Extending Constraints with Conditions for Data Cleaning[C]// IEEE 8th Int'l Conf. on Computer and Information Technology (invited). 2008

[23] Bohannon P, et al. Conditional Functional Dependencies for Data Cleaning[C]// The 23rd International Conference on Database Engineering (ICDE) (the best paper award). 2007

[24] Fan Wenfei, Hu Yanli, Liu Jie, et al. Propagating Functional Dependencies with Conditions[C]// The 34th International Conference on Very Large Data Bases (VLDB). 2008

[25] 叶舟，王东. 基于规则引擎的数据清洗[J]. 计算机工程，2006，32 (23)：52-54

[26] 郭志懋，俞荣华，田增平，周傲英. 一个可扩展的数据清洗系统[J]. 计算机工程，2003，29(3)：95-96

[27] 谈子敬，施伯乐. 函数依赖和规范化在关系和 XML 间的传播[J]. 软件学报，2005，16(4)：533-539

[28] Fan Wenfei, Hu Yanli, Liu Jie, et al. Propagating Functional Dependencies with Conditions[C]// VLDB. Auckland, New Zealand, 2008

[29] Golab L, Korn F, Srivastava D, et al. On Generating Near-Optimal Tableaux for Conditional Functional Dependencies[C]// The 34th International Conference on Very Large Data Bases (VLDB). 2008

[30] Fan Wenfei, Jia Xibei. Semandaq: A Data Quality System Based on Conditional Functional Dependencies[C]//The 34th International Conference on Very Large Data Bases (VLDB), demo, 2008

[31] Bravo L, et al. Increasing the Expressivity of Conditional Functional Dependencies without Extra Complexity[C]// The 24th International Conference on Database Engineering (ICDE). 2008