一种时态数据形式语言模型

苗德成[1,2] 奚建清[2]
(韶关学院数学与信息科学学院 韶关 512005)[1] (华南理工大学计算机科学与工程学院 广州 510640)[2]

摘 要 数据模型是数据库技术发展的主线,时态数据模型是时态数据库系统的核心与基础。针对时态数据模型的研究现状,初步探讨了时态数据模型的基本要素,建立了一种形式化时态数据模型;基于形式语言理论和形式语义学的指称语义方法,进一步建立了该时态数据模型的形式语言模型。应用时态数据形式语言模型定义了各类时态完整性约束的形式语义规则,深入分析了时态数据模型内在的时态语义联系,为时态数据模型的研究提供了一个便利、高效的形式化理论框架。

关键词 模型,形式语言,时态数据,完整性约束,形式语义
中图法分类号 TP301 **文献标识码** A

Formal Languages Model for Temporal Data

MIAO De-cheng[1,2] XI Jian-qing[2]
(School of Math and Information Science, Shaoguan University, Shaoguan 512005, China)[1]
(School of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, China)[2]

Abstract Data model is the main clue of trends in database technology, and tempoaral data model is the core and basis of temporal database system. This paper discussed preliminarily some basic elements of data model in accordance with the status quo of temporal data model, made a temporal data model formalized, and further made a formal languages model of this temporal data model based on formal languages theory and denotational semantics method of formal semantics. By the formal languages model this paper defined some formal semantics rules for all kinds of temporal integrity constraints, and deeply analyzed inherent temporal semantics relationships of temporal data model, which provides an efficient and convenient formalization theory framework for studying temporal data model.

Keywords Model, Formal languages, Temporal data, Integrity constraints, Formal semantics

1 引言

随着大量与时间相关的数据库应用需求不断增多,对ER模型进行时态扩展使其能准确地捕捉时间变化信息,已成为研究热点之一[1]。目前,时态数据库技术仍处于研究与发展阶段,学术界和工业界已提出几十种时态数据模型,但现有时态数据模型不够成熟,没有形成完整的国际标准[2]。大多数时态数据模型局限于数据库时态属性的研究,停留在数据处理层次,对时态逻辑信息表示、知识推理和完整性约束等方面的研究不够深入,主要体现在时态信息处理能力弱,时态数据计算体系不完备,时态数据依赖缺乏系统的时态规范化理论支持,因此现有时态数据模型完整性约束难以对时态数据模型进行有效规范化。

2 时态数据模型研究现状

表1归纳了国外一些具有代表性的时态数据模型的基本情况,12种时态数据模型的数据结构基本都是关系,现有时态数据模型是传统关系数据模型的扩展,将关系数据库作为特例。时态数据库支持3种时间类型:有效时间VT、事务时间TT与用户定义时间UDT[1],VT与TT是两个正交的时间维,时态数据模型至少支持其中一维,所有时态数据模型都支持UDT,表1中TempEER[3]、BCDM[2]与TimeER[5]支持TT。时态数据库有3种时态数据类型:时刻(instant)、时区(interval)与时长(period)[3]。其中,时长与时区均表示一段时间,但前者没有起点与终点时刻,表1只有BCDM同时支持VT的3种数据类型。时态数据库定义2种时间粒度:单粒度与多粒度[1],表1中5个时态数据模型支持多粒度,其余时态数据模型由于具体应用时间粒度不同而在数据库逻辑设计阶段延迟粒度的选择。时态数据操作扩展传统关系操作,在时态数据库中增加时态选择、时态投影、时态连接等时态操作,表1中大部分时态数据模型提供关系表达式、时态表达式及SQL表达式的VT操作,RAKE[3]只支持普通SQL操作,而MOTAR[3]与TERC+[3]扩展时态数据模型面向对象功能,所有操作均为函数,部分函数可嵌套,TempEER、BCDM

到稿日期:2011-05-19 返修日期:2011-09-18 本文受国家自然科学基金(61103038),广东省教育部产学研结合项目(2010B090400335),韶关学院科研项目(201020704)资助。

苗德成(1979—),男,博士生,讲师,主要研究方向为数据库与网络计算、形式语义学、软件系统形式化理论,E-mail:tony10860@126.com;奚建清(1962—),男,博士,教授,博士生导师,主要研究方向为数据库与网络计算、形式语义学、软件系统形式化理论。

与 TimeER 同时提供 VT 操作和 TT 操作。旧版本的向上兼容性[3]为新版本增强功能提供了平稳过渡，表 1 中 7 种时态数据模型具备向上兼容性。目前大部分时态数据模型查询语言扩展 SQL、Quel 等语言的时态数据查询功能，但时态查询其功能有限，效率较低，BCDM 的查询语言有 TempSQL、Tquel、TSQL2 等。表 1 中 4 种时态数据模型不支持时态约束，现有时态数据模型规范化程度最好的是 TERM[3] 与 TERC+，尽管两者均支持时态约束，但无法灵活地表达 UDT 变化属性之间固有的依赖约束。

国内学者汤庸将时态应用分为完全时态应用、嵌入式时态应用和混合型时态应用 3 种模式，提出一种时态知识/数据模型[4]及其框架原型，已在典型时态信息领域得到成功应用。郝忠孝提出的有限属性闭包、成员籍等时态函数依赖集的一些重要算法[1]对时态数据模型的设计具有普遍的意义。现有时态数据模型整体上在概念层以不同方式有效捕捉数据的时态特性，但每个时态数据模型侧重于不同应用领域建模，有较强倾向性而不具备普适性，难以满足时态数据库所有的时态特性要求。本文建立一种形式化时态数据模型及其形式语言模型，应用该形式语言模型定义各类时态完整性约束的形式语义规则，深入分析时态数据模型内在的时态语义联系，为时态数据模型的研究提供一个便利、高效的形式化理论框架。

表 1　时态数据模型研究现状

	数据结构	时间类型	VT 数据类型	多粒度	时态数据操作	向上兼容	查询语言	时态约束
TERM	关系	UDT、VT	时刻、时区	支持	VT 操作	否	无	支持
RAKE	关系	UDT、VT	时刻、时区	不支持	SQL 操作	是	无	不支持
MOTAR	关系、对象	UDT、VT	时刻	支持	函数操作	是	无	不支持
TEER	关系	UDT、VT	时长	不支持	VT 操作	否	有	支持
STEER	关系	UDT、VT	时长	不支持	VT 操作	否	有	支持
ERT	关系	UDT、VT	时区	不支持	VT 操作	是	有	支持
TER	关系	UDT、VT	时刻	不支持	VT 操作	否	无	不支持
TempEER	关系	UDT、VT、TT	时区	不支持	VT 操作、TT 操作	否	有	支持
TempRT	关系	UDT、VT	时区	不支持	VT 操作	是	无	不支持
TERC+	关系、对象	UDT、VT	时长	支持	关系操作、函数操作	是	无	支持
BCDM	关系	UDT、VT、TT	时刻、时区、时长	支持	VT 操作、TT 操作	是	有	支持
TimeER	关系	UDT、VT、TT	时刻、时长	支持	VT 操作、TT 操作	是	有	支持

3　时态数据形式语言模型

设 $S=\{s_1,s_2,\cdots,s_n\}$ 为有限集，$S_1\times S_2\times\cdots\times S_n=(s_1,s_2,\cdots,s_n)$ 为卡氏积，$s_i\in S_i$，$1\leqslant i\leqslant n$，记 $M(S)=\{\{s_1,s_2,\cdots,s_n\}\}$ 为多重集，$S^+=\langle s_1,s_2,\cdots,s_n\rangle$ 为 S 上非空有限表域，$S_1\uplus S_2$ 为不相交并，$\perp$ 为集合内未定义元素。

3.1　时间模型

本文采用离散线性时间模型，时域是一全序有限集，同构于自然数集 N 的有限子集，时域元素称为时间量子[2]，记为 c_i，其中 $i\in\mathbf{N}$。c_i 将时间基线划分为许多独立等值长度的时间片，$g=|c_i|$ 为时间粒度，反映时态数据库时刻最小值，由应用程序显式指定。引入两个时间变元 *now*[3] 与 *UC*[3]，前者有效值依赖于当前时间，后者指称 TT 中元组改变的时间。

记 VTE 为时态数据库中作用于实体的 VT，VTA 为作用于属性的 VT。设不同属性 VT 域为 $D^g_{VTA}=\{c^g_1,c^g_2,\cdots,c^g_k\}$，所有属性 VT 域为 $\mathcal{D}_{VTA}=\bigcup_g D^g_{VTA}$。设不同实体 VT 域为 $D^g_{VTE}=\{c^g_1,c^g_2,\cdots,c^g_{now}\}$，所有实体 VT 域为 $\mathcal{D}_{VTE}=\bigcup_g D^g_{VTE}$。设不同 TT 域为 $D^g_{TT}=\{c^g_1,c^g_2,\cdots,c^g_{now}\}\cup\{UC\}$，所有 *TT* 域为 $\mathcal{D}_{TT}=\bigcup_g D^g_{TT}$。

3.2　形式化时态数据模型

数据结构、数据操作与完整性约束是数据模型的 3 个基本要素，数据结构是数据模型的基础，是时态数据结构描述时态数据库系统静态特征的对象类型，是构成时态数据库的基本组成成份。数据操作描述数据库系统动态特性，时态数据操作定义时态数据库允许执行的操作集合，时态数据模型必须定义时态数据操作的确切含义、操作符号、操作规则。完整性约束是数据模型中数据及其联系的语义约束和依存规则，时态完整性约束限定时态数据库状态及其变化，保证时态数据库中数据的正确、有效、相容。

定义 1　本文建立的形式化时态数据模型 $\mathcal{TDM}$ 是一个三元组，$\mathcal{TDM}=(TDS,TDM,TIC)$，其中 *TDS* 为时态数据结构，*TDM* 为时态数据操作，*TIC* 为时态完整性约束，用巴克斯-诺尔范式 BNF 分别表示如下：

$TDS::=SCH|E_D|R_D|A_D|T_s|t$

SCH 为时态数据库模式，E_D 与 R_D 分别为实体型与联系型，A_D 为属性定义，T_s 为时集，t 为元组。

$TDM\supseteq\{\cup_T:E_D\times E_D\to E_D,-_T:E_D\times E_D\to E_D,\times_T:E_D\times\cdots\times E_D\to E_D,\sigma_T:E_D\to E_D,\pi_T:E_D\to\{t[A]|t\in E_D\}\}$

式中，$t[A]$ 为 t 在属性 A 上的分量，时态并、时态差、时态卡氏积、时态选择与时态投影是时态数据操作的 5 种基本运算，其它时态操作如时态交、时态连接与时态除等均可用这 5 种基本运算表达。

$TIC::=TIC_1;TIC_2|EIC|WEIC|RIC|IIC|UIC|PIC|TC$

$EIC::=SPK|TIK|TK$

$UIC::=IS\_A|HAS\_PART\_OF|GC$

$PIC::=$Snapshot Participation of E in R is(min,max)|Valid Time Participation of E in R is[min,max]|Participation of R to E is p_1,p_2

$p_1::=disjoint|overlapping$

$p_2::=total|partial$

$GC::=GP|GT$

$TC::=DEX|DEV$

时态完整性约束有实体完整性约束 EIC、弱实体完整性约束 WEIC、参照完整性约束 RIC、固有完整性约束 IIC、用户定义完整性约束 UIC、参与完整性约束 PIC 和变迁约束 TC 等 7 种类型。

3.3 时态数据形式语言模型

形式语言理论是用数学方法研究自然语言和程序设计语言的产生方式、一般性质和规则的理论，而形式语义学是形式语言理论的重要组成部分，以数学为工具，利用符号和公式精确定义和解释程序设计语言的语义。通过构造表达语义的数学对象，指称语义方法形式化描述程序设计语言的语义，其程序执行结果不因程序设计语言不同而变化。本文基于形式语言理论和形式语义学的指称语义方法，同时参考 TimeER 模型的文本表示法[5,6]，建立 $\mathcal{TDM}$ 的形式语言模型。

定义 2 $\mathcal{TDM}$ 的形式语言模型 $\mathcal{L}(\mathcal{TDM})$ 是一个四元组，$\mathcal{L}(\mathcal{TDM})=(SYN, SEM, AUF, SEF)$。其中，$SYN$ 与 SEM 分别为 $\mathcal{L}(\mathcal{TDM})$ 的语法域与语义域，AUF 为辅助函数，前缀为函数返回时态数据类型，辅助函数定义中的方括号表示可选项，SEF 为语义函数。

引入一些元变量符号：E、**Weak** E、R、A 分别为实体名、弱实体名、联系名、属性名，$B\in 2^A$ 为属性子集，粗体字为编译系统解析的分词。

SYN

$SCH::=E_D;R_D$

$E_D::=E_{D1};E_{D2}$ | **Entity Type** E **has** A_D | **Entity Type** E **with** T_s **has** A_D | **Weak Entity Type** E **has** A_D | **Weak Entity Type** E **with** T_s **has** A_D | **Entity Type** E_1 **IS_A** E_2 **has** A_D | **Entity Type** E_1 **IS_A** E_2 **with** T_s **has** A_D

$R_D::=R_{D1};R_{D2}$ | **Relationship Type** R **has** A_D | **Relationship Type** R **with** T_s **has** A_D

$A_D::=A_{D1};A_{D2}$ | **Attribute Type** A **is** d | **Attribute Type** A **is** d **with** T_s

$d::=$int | boolean | string

$T_s::=T_{s1};T_{s2}$ | (dim, ts, g)

$dim::=VTE|VTA|TT$

$ts::=$instant | period

$g::=$sec | min | hour | day | week | month | year

SEM

$D_S\cup\{\perp\}$：置换集[7]；$D_S^g\subseteq D_S$：实体 E 的置换集；$D_{VTA}=\mathcal{D}_{VTA}\cup\{\perp\}$：属性 VT 域集；$D_{VTE}=\mathcal{D}_{VTE}\cup\{\perp\}$：实体 VT 域集；$D_{TT}=\mathcal{D}_{TT}\cup\{\perp\}$：TT 域集；$\mathcal{D}[\![d]\!]$：基本数据类型域集；$PRED$：谓词集；$Role$：$E_D$：角色集；$D_{dim}^g$：时刻集。

AUF

二元辅助函数 $inSch$ 以实体名或弱实体名或联系名和数据库模式为参数，如果该实体或联系在此模式中定义，则返回真，否则为假。

boolean：$inSch(E, SCH)$ = boolean：$inSch(E, E_D)$

$$=\begin{cases}\text{ture}, & \text{if Entity Type } E[\text{with } T_s]\text{has } A_D\in E_D\\ \text{false}, & \text{otherwise}\end{cases}$$

boolean：$inSch$(**Weak** E, SCH) = boolean：$inSch$ (**Weak** E, E_D) $=$

$$\begin{cases}\text{ture}, & \text{if Weak Entity Type } E\\ & [\text{with } T_s]\text{ has } A_D\in E_D\\ \text{false}, & \text{otherwise}\end{cases}$$

boolean：$inSch(R, SCH)$ = boolean：$inSch(R, R_D)$ =

$$\begin{cases}\text{true}, & \text{if Relationship Type } R[\text{with } T_s]\text{ has } A_D\in R_D\\ \text{false}, & \text{otherwise}\end{cases}$$

一元辅助函数 $attOf$ 以实体类型声明或属性定义为参数，返回该实体类型的属性名。

$A_D:attOf$(**Entity Type** E[**with** T_s] **has** A_D) = $A_D:attOf(A_D)$

$A_D:attOf(A_{D1};A_{D2}) = A_D:attOf(A_{D1})\cup A_D:attOf(A_{D2})$

$A_D:attOf$(**Attribute Type** A **is** d[**with** T_s]) = $\{A\}$

一元辅助函数 $entPar$ 以联系名或实体名或弱实体名为参数，返回参与该联系的所有实体、弱实体名。$E_i\in R$ 表示实体 E_i 参与联系 R，单实体型或弱实体型内的联系结果为多重集。

$$E_D:entPar(R)=\begin{cases}\{E_1, E_2, \cdots, E_n\} & \text{if } E_i\in R,\quad 1\leqslant i\leqslant n\\ \perp, & \text{otherwise}\end{cases}$$

$E_D:entPar(R_1;R_2)=E_D:entPar(R_1)\cup E_D:entPar(R_2)$

$E_D:entPar(E)=\{\{E\}\}$

$E_D:entPar$(**Weak** E) = $\{\{E\}\}$

二元辅助函数 $belTo$ 以弱实体名和联系名为参数，返回该弱实体隶属的实体名。

$E_D:belTo$(**Weak** E, R) =

$$\begin{cases}\{entPar(R)-\text{Weak } E\}, & \text{if Weak } E\in R\\ \varnothing, & \text{otherwise}\end{cases}$$

一元辅助函数 $attList$ 以实体名或弱实体名为参数，返回该实体或弱实体的所有属性名；如果参数为子实体，则返回该子实体的属性名和其父实体的所有属性名；如果参数为弱实体，则只返回此弱实体的所有属性名。

$A_D:attList(E)$ =

$$\begin{cases}attOf(\text{Entity Type } E\cdots), & \text{if Entity Type } E\cdots\in E_D\\ attOf(\text{Entity Type } E \text{ IS\_A } E'\cdots)\cup A_D:attList(E'), & \\ \quad \text{if Entity Type } E\cdots\in E_D & \\ \perp, \quad \text{otherwise} & \end{cases}$$

$A_D:attList$(**Weak** E) = $attOf$(**Weak Entity Type** $E\cdots$)

一元辅助函数 $entNum$ 以联系 R 为参数，返回参与 R 的实体数目。R 为所有参与实体的自然连接，具有 n 个属性，$R.A$ 为 R 的属性集，$\pi_{TA_i}(R)$ 为对 R 的属性 A_i 的时态投影运算。

int：$entNum(R)$ =

$$\begin{cases}0, & \text{if } R.A=\varnothing\\ cnt(R-_T E), & \text{if } \cup_{1\leqslant i\leqslant n}\pi_{TA_i}(R)\neq E\\ cnt(R-_T E)+1, & \text{if } \cup_{1\leqslant i\leqslant n}\pi_{TA_i}(R)=E\end{cases}$$

SEF

对于指定的实体型、联系型或者属性定义，语义函数 $\mathcal{T}$ 返回具体时态支持的时域，其函数声明和语义方程定义如下：

$\mathcal{T}: T_s\to D_{VTE}\cup D_{VTA}\cup D_{TT}$

$\mathcal{T}[\![\text{with } T_{s1};T_{s2}]\!]=\mathcal{T}[\![T_{s1}]\!]\times\mathcal{T}[\![T_{s2}]\!]$ (1)

$\mathcal{T}[\![(dim, instant, g)]\!]=D_{dim}^g$ (2)

$\mathcal{T}[\![(dim, period, g)]\!]=2^{D_{dim}^g}$ (3)

语义函数 $\mathcal{A}$ 以属性定义为参数，返回具体属性的时态值，其函数声明和语义方程如下：

$\mathcal{A}: A_D\times d\times T_s\to\mathcal{D}[\![d]\!]\cup(\mathcal{T}[\![T_s]\!]\to\mathcal{D}[\![d]\!])$

$\mathcal{A}[\![A_{D1};A_{D2}]\!]=\mathcal{A}[\![A_{D1}]\!]\times\mathcal{A}[\![A_{D2}]\!]$ (4)

$\mathcal{A}[\![\text{Attribute Type } A \text{ is } d]\!]=\mathcal{D}[\![d]\!](d)$ (5)

$\mathcal{A}[\![\text{Attribute Type } A \text{ is } d \text{ with } T_s]\!]=\mathcal{T}[\![T_s]\!]\to\mathcal{D}[\![d]\!](d)$ (6)

语义函数 $\mathcal{I}$ 确定实体型的置换集，时态数据库自动生成实体在全系统唯一的置换属性[7]，以识别随时间变化的实体，其函数声明和语义方程定义如下：

$$\mathcal{I}: E \to D_S^E$$

$$\mathcal{I}[\![E]\!] = \begin{cases} D_S^E, & \text{if } E \in E_D \\ \perp, & \text{otherwise} \end{cases} \tag{7}$$

语义函数 ε 通过实体型定义识别实体实例，返回存储在时态数据库中该实体的元组集，dom 为定义域函数，实体语义方程中 a 为置换属性，弱实体与继承实体语义方程中 s 为置换运算[7]，sE_i 为实体 E_i 在时态数据库内唯一的置换属性，继承实体语义方程中 $E_2.T_s$ 为父实体 E_2 的时态特性。语义函数 ε 声明和语义方程定义如下：

$$\varepsilon: E_D \times T_s \times A_D \to D_S^E \times \mathcal{A}[\![A_D]\!] \cup (D_S^E \times \mathcal{T}[\![T_s]\!]) \times \mathcal{A}[\![A_D]\!]$$

$$\varepsilon[\![E_{D1}; E_{D2}]\!] = \varepsilon[\![E_{D1}]\!] \uplus \varepsilon[\![E_{D2}]\!] \tag{8}$$

$$\varepsilon[\![\textbf{Entity Type } E \textbf{ has } A_D]\!] = \{t \mid t \in S^+ \wedge dom(t) = \{a, attOf(A_D)\} \wedge t[a] \in \mathcal{I}[\![E]\!] \wedge_{A_i \in attOf(A_D)} t[A_i] \in \mathcal{A}[\![A_D]\!] \wedge (\forall t_i, t_j, i \neq j \Rightarrow t_i[a] \neq t_j[a])\} \tag{9}$$

$$\varepsilon[\![\textbf{Entity Type } E \textbf{ with } T_s \textbf{ has } A_D]\!] = \{t \mid t \in S^+ \wedge dom(t) = \{a, attOf(A_D)\} \wedge t[a] \in (\mathcal{T}[\![T_s]\!] \to \mathcal{I}[\![E]\!]) \wedge_{A_i \in attOf(A_D)} t[A_i] \in \mathcal{A}[\![A_D]\!] \wedge \forall c^t \in \mathcal{D}_{VTE}\, \forall c^t \in \mathcal{D}_{TT} (\forall t_i, t_j, i \neq j \Rightarrow t_i[a] \neq t_j[a])\} \tag{10}$$

$$\varepsilon[\![\textbf{Weak Entity Type } E \textbf{ has } A_D]\!] = \{t \mid t \in S^+ \wedge dom(t) = \{\cup_{E_i \in belTo(E,R)} sE_i, attOf(A_D)\} \wedge_{E_i \in belTo(E,R)} t[sE_i] \in \mathcal{I}[\![E_i]\!] \wedge_{A_i \in attOf(A_D)} t[A_i] \in \mathcal{A}[\![A_D]\!]\} \tag{11}$$

$$\varepsilon[\![\textbf{Weak Entity Type } E \textbf{ with } T_s \textbf{ has } A_D]\!] = \{t \mid t \in S^+ \wedge dom(t) = \{\cup_{E_i \in belTo(E,R)} sE_i, attOf(A_D)\} \wedge_{E_i \in belTo(E,R)} t[sE_i] \in \mathcal{T}[\![T_s]\!] \to \mathcal{I}[\![E_i]\!] \wedge_{A_i \in attOf(A_D)} t[A_i] \in \mathcal{A}[\![A_D]\!]\} \tag{12}$$

$$\varepsilon[\![\textbf{Entity Type } E_1 \textbf{ IS\_A } E_2 \textbf{ has } A_D]\!] = \{t \mid t \in S^+ \wedge dom(t) = \{sE_2, attList(E_2), attOf(A_D)\} \wedge t[sE_2] \in \mathcal{T}[\![E_2.T_s]\!] \to D_{S^2}^E \wedge_{A_i \in attList(E_2)} t[A_i] \in \mathcal{A}[\![A_D]\!] \wedge_{A_i \in attOf(A_D)} t[A_i] \in \mathcal{A}[\![A_D]\!] \tag{13}$$

$$\varepsilon[\![\textbf{Entity Type } E_1 \textbf{ IS\_A } E_2 \textbf{ with } T_s \textbf{ has } A_D]\!] = \{t \mid t \in S^+ \wedge dom(t) = \{sE_2, attList(E_2), attOf(A_D)\} \wedge t[sE_2] \in \mathcal{T}[\![E_2.T_s]\!] \times \mathcal{T}[\![T_s]\!] \to D_S^{E_2} \wedge_{A_i \in attList(E_2)} t[A_i] \in \mathcal{A}[\![A_D]\!] \wedge_{A_i \in attOf(A_D)} t[A_i] \in \mathcal{A}[\![A_D]\!]\} \tag{14}$$

4 时态完整性约束的形式语义

本文在时态数据形式语言模型 $\mathcal{L}(\mathcal{TDM})$ 理论框架内定义 $\mathcal{TDM}$ 各类时态完整性约束的形式语义规则，深入分析时态数据模型内在的时态语义联系。一元语义函数 $\mathcal{C}$ 定义时态数据库满足各种约束条件的谓词集，确保时态数据库逻辑设计的有效性，其函数声明和语义方程定义如下：

$$\mathcal{C}: TIC \to PRED$$

$$\mathcal{C}[\![TIC_1; TIC_2]\!] = \mathcal{C}[\![TIC_1]\!] \wedge \mathcal{C}[\![TIC_2]\!] \tag{15}$$

$$\mathcal{C}[\![B \textbf{ is Simple Primary Key of } E]\!] = inSch(E, SCH) \wedge ((B \subseteq attOf(\textbf{Entity Type } E \textbf{ has } A_D) \wedge \forall t_i, t_j \in \varepsilon[\![\textbf{Entity Type } E \cdots]\!]((t_i[B] = t_j[B] \Rightarrow t_i = t_j \wedge AT(d\,g)))) \vee (B \subseteq attOf(\textbf{Entity Type } E \textbf{ with } T_s \textbf{ has } A_D) \wedge \forall t_i, t_j \in \varepsilon[\![\textbf{Entity Type } E \cdots]\!]((\mathcal{T}[\![T_s]\!] \to t_i[B] = \mathcal{T}[\![T_s]\!] \to t_j[B] \Rightarrow \mathcal{T}[\![T_s]\!] \to t_i = \mathcal{T}[\![T_s]\!] \to t_j) \wedge AT(d\,g))) \tag{16}$$

$$\mathcal{C}[\![B \textbf{ is TimeInvariant Key of } E]\!] = inSch(E, SCH) \wedge ((B \subseteq attOf(\textbf{Entity Type } E \textbf{ has } A_D) \wedge \forall t_i, t_j \in \varepsilon[\![\textbf{Entity Type } E \cdots]\!]((t_i[B] = t_j[B] \Rightarrow t_i = t_j \wedge IN(d\,g)))) \vee (B \subseteq attOf(\textbf{Entity Type } E \textbf{ with } T_s \textbf{ has } A_D) \wedge \forall t_i, t_j \in \varepsilon[\![\textbf{Entity Type } E \cdots]\!]((\mathcal{T}[\![T_s]\!] \to t_i[B] = \mathcal{T}[\![T_s]\!] \to t_j[B] \Rightarrow \mathcal{T}[\![T_s]\!] \to t_i = \mathcal{T}[\![T_s]\!] \to t_j) \wedge IN(dg)))) \tag{17}$$

$$\mathcal{C}[\![B \textbf{ is Temporal Key of } E]\!] = inSch(E, SCH) \wedge ((B \subseteq attOf(\textbf{Entity Type } E \textbf{ has } A_D) \wedge \forall t_i, t_j \in \varepsilon[\![\textbf{Entity Type } E \cdots]\!]((t_i[B] = t_j[B] \Rightarrow t_i = t_j \wedge [E.VTE_s, E.VTE_e]))) \vee (B \subseteq attOf(\textbf{Entity Type } E \textbf{ with } T_s \textbf{ has } A_D) \wedge \forall t_i, t_j \in \varepsilon[\![\textbf{Entity Type } E \cdots]\!]((\mathcal{T}[\![T_s]\!] \to t_i[B] = \mathcal{T}[\![T_s]\!] \to t_j[B] \Rightarrow \mathcal{T}[\![T_s]\!] \to t_i = \mathcal{T}[\![T_s]\!] \to t_j) \wedge [E.VTE_s, E.VTE_e]))) \tag{18}$$

图 1 扩展 TimeER 图[5]表示法，为 company 时态数据库实例的 EER 图。图 1 有职工、部门等实体，其中家人为弱实体，部门属性办公地点为多重值，实体右上角标识其支持的时间类型，属性右边标识其支持的时间类型。用带圆圈的 o 表示重叠 IS_A 约束，带圆圈的 d 表示不相交 IS_A 约束。时态数据库有 3 类实体完整性约束，简单主码约束 SPK 在时态数据库任一快照中唯一确定一个实体，$AT \in PRED$, $AT(d\,g) \in D_{dm}^{\mathcal{E}}$；时间不变式[7]码 TIK 是一类 SPK，在实体存在的一段 VT 内不随时间改变，如图 1 规定职工号 50 年内唯一，50 年后同一职工号可赋予另一职工，$IN \in PRED$, $IN(d\,g) \in 2^{D_{dm}^{\mathcal{E}}}$；时态码[8] TK 是一类 TIK，在实体整个 VT 内唯一，如图 1 中职工的身份证号，其中 $[E.VTE_s, E.VTE_e]$ 表示实体 E 在整个时态数据库中的 VT。

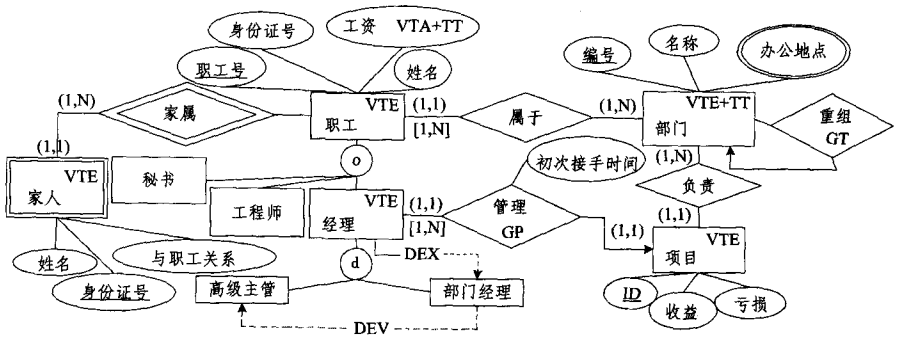

图 1 company 时态数据库实例 EER 图

$$\mathcal{C}[\![B \textbf{ is Partial Key of Weak } E]\!] = inSch(\textbf{Weak } E, SCH) \wedge ((B \subseteq attOf(\textbf{WeakEntity Type } E \textbf{ has } A_D) \wedge \forall t_i, t_j \in \varepsilon[\![\textbf{Weak Entity Type } E \cdots]\!](\cup_{E^1 \in belTo(E,R)} t_i[sE^1] = \cup_{E^1 \in belTo(E,R)} t_j[sE^1]) \wedge (t_i[B] = t_j[B]) \Rightarrow t_i = t_j) \vee$$

$(B \subseteq attOf(\textbf{WeakEntity Type } E \textbf{ with } T_s \textbf{ has } A_D) \wedge \forall t_i, t_j \in \varepsilon[\![\textbf{Weak Entity Type } E \cdots]\!](\mathcal{T}[\![T_s]\!] \rightarrow \bigcup_{E^1 \in belTo(E,R)} t_i[sE^1] = \mathcal{T}[\![T_s]\!] \rightarrow \bigcup_{E^1 \in belTo(E,R)} t_j[sE^1]) \wedge (\mathcal{T}[\![T_s]\!] \rightarrow t_i[B] = \mathcal{T}[\![T_s]\!] \rightarrow t_j[B]) \Rightarrow \mathcal{T}[\![T_s]\!] \rightarrow t_i = \mathcal{T}[\![T_s]\!] \rightarrow t_j)) \wedge [E.VTE_s, E.VTE_e] \subseteq [\bigcup_{E^1 \in belTo(E,R)} E^1.VTE_s, \bigcup_{E^1 \in belTo(E,R)} E^1.VTE_e]))$ (19)

弱实体完整性约束 WEIC 中，弱实体的码用 Partial Key 表示，弱实体 E 的存在必须以属主实体 E^1 的 VT 为前题，即 $[E.VTE_s, E.VTE_e] \subseteq [\bigcup_{E^1 \in belTo(E,R)} E^1.VTE_s, \bigcup_{E^1 \in belTo(E,R)} E^1.VTE_e]$。

$\mathcal{C}[\![B \textbf{ is Foreign Key of } E]\!] = inSch(E, SCH) \wedge \exists S \in E_D (inSch(S, SCH) \wedge B \in \{S.SPK, S.TIK, S.TK\}) \wedge ((B \subseteq attOf(\textbf{Entity Type } E \textbf{ has } A_D) \wedge \forall t \in \varepsilon[\![\textbf{Entity Type } E \cdots]\!]((t[B] \textbf{ is Null}) \vee (t[B] = t[S.K], K \in \{SPK, TIK, TK\}))) \vee (B \subseteq attOf(\textbf{Entity Type } E \textbf{ with } T_s \textbf{ has } A_D) \wedge \forall t \in \varepsilon[\![\textbf{Entity Type } E \cdots]\!]((\mathcal{T}[\![T_s]\!] \rightarrow t[B] \textbf{ is Null}) \vee (\mathcal{T}[\![T_s]\!] \rightarrow t[B] = \mathcal{T}[\![T_s]\!] \rightarrow t[S.K]))))$ (20)

时态数据库参照完整性约束 RIC 定义外码与码之间的引用规则，$S.SPK$，$S.TIK$，$S.TK$ 分别表示参照表 S 的简单主码、时间不变式码与时态码。

$\mathcal{C}[\![B \textbf{ is Inherent Attribute of } E]\!] = inSch(E, SCH) \wedge ((B \subseteq attOf(\textbf{Entity Type } E \textbf{ with } T_s \textbf{ has } A_D) \wedge ([B.VTA_s, B.VTA_e] \subseteq [E.VTE_s, E.VTE_e]) \wedge ([B.TT_s, B.TT_e] \subseteq [E.VTE_s, E.VTE_e]) \wedge ([E.TT_s, E.TT_e] \subseteq [E.VTE_s, E.VTE_e]))$ (21)

固有完整性约束 IIC 是时态数据库特有的约束类型，实体属性的 VT 必须是实体 VT 的子集；同时，时态数据库执行事务活动时，属性与实体的 TT 也必须是实体 VT 的子集。

$\mathcal{C}[\![\textbf{Entity Type } E_1 \textbf{ IS\_A } E_2 \textbf{ has } A_D]\!] = \wedge_{i=1,2} inSch(E_i, SCH) \wedge ([E_1.VTE_s, E_1.VTE_e] \subseteq [E_2.VTE_s, E_2.VTE_e]) \wedge (attList(E_2) \subseteq A_D)$ (22)

$\mathcal{C}[\![\textbf{Entity Type } E_1 \textbf{ IS\_A } E_2 \textbf{ with } T_s \textbf{ has } A_D]\!] = \wedge_{i=1,2} inSch(E_i, SCH) \wedge ([E_1.VTE_s, E_1.VTE_e] \subseteq [E_2.VTE_s, E_2.VTE_e]) \wedge (\mathcal{T}[\![T_s]\!] \rightarrow attList(E_2) \subseteq \mathcal{T}[\![T_s]\!] \rightarrow A_D)$ (23)

$\mathcal{C}[\![\textbf{Entity Type } E \textbf{ HAS\_PART\_OF } E_1, E_2, \cdots, E_n \textbf{ has } A_D]\!] = inSch(E, SCH) \wedge_{1 \leqslant i \leqslant n} inSch(E_i, SCH) \wedge (E.VTE = \bigcup_{1 \leqslant i \leqslant n} E_i.VTE) \wedge (\bigcup_{1 \leqslant i \leqslant n} attList(E_i) \subseteq A_D)$ (24)

$\mathcal{C}[\![\textbf{Entity Type } E \textbf{ HAS\_PART\_OF } E_1, E_2, \cdots, E_n \textbf{ with } T_s \textbf{ has } A_D]\!] = inSch(E, SCH) \wedge_{1 \leqslant i \leqslant n} inSch(E_i, SCH) \wedge (E.VTE = \bigcup_{1 \leqslant i \leqslant n} E_i.VTE) \wedge (\bigcup_{1 \leqslant i \leqslant n} (\mathcal{T}[\![T_s]\!] \rightarrow attList(E_i)) \subseteq \mathcal{T}[\![T_s]\!] \rightarrow A_D)$ (25)

用户自定义 IS_A[3,9] 约束中，子实体 VT 的是父实体 VT 的子集，并从父实体继承所有属性及其时态特性，子实体不可修改或删除继承属性及时态特性，但可增加新属性。HAS_PART_OF[3] 规则约束局部实体继承全局实体的时态特性，并将全局实体视为由所有局部实体构成的一个单一实体，所有局部实体属性的时态特性同步更新。

$\mathcal{C}[\![\textbf{Entity Type } E_1 \textbf{ Generation Production } E_2 \textbf{ with } T_s]\!] = \wedge_{i=1,2} inSch(E_i, SCH) \wedge (\mathcal{T}[\![T_s+1]\!] \rightarrow E_1 = \mathcal{T}[\![T_s]\!] \rightarrow E_1)$ (26)

$\mathcal{C}[\![\textbf{Entity Type } E_1 \textbf{ Generation Transformation } E_2 \textbf{ with } T_s]\!] = \wedge_{i=1,2} inSch(E_i, SCH) \wedge (\mathcal{T}[\![T_s+1]\!] \rightarrow E_2 \neq \mathcal{T}[\![T_s]\!] \rightarrow E_1) \wedge (\mathcal{T}[\![T_s+1]\!] \rightarrow E_1 \textbf{ is Null})$ (27)

用户自定义二元生成规则 GC 约束从源实体产生目标实体，根据源实体是否保留可分为生成积 GP 约束和生成转换 GT 约束两类，前者保留源实体，如图 1 所示，经理通过“管理”联系生成目标项目实体，而源实体经理仍在时态数据库中保留；后者不保留源实体，重组后的部门与源部门不同，$\mathcal{T}[\![T_s+1]\!]$ 表示 $\mathcal{T}[\![T_s]\!]$ 的下一时刻，在图 1 中用 GP 与 GT 表示联系的类型，用箭头指向目标实体。

$\mathcal{C}[\![\textbf{Snapshot Participation of } E \textbf{ in } R \textbf{ is}(\min, \max)]\!] = inSch(E, SCH) \wedge inSch(R, SCH) \wedge (E \in entPar(R) \wedge (\min \leqslant entNum(R) \leqslant \max) \wedge AT(d\,g))$ (28)

$\mathcal{C}[\![\textbf{Valid Time Participation of } E \textbf{ in } R \textbf{ is}[\min, \max]]\!] = inSch(E, SCH) \wedge inSch(R, SCH) \wedge (E \in entPar(R) \wedge (\min \leqslant entNum(R) \leqslant \max) \wedge [E.VTE_s, E.VTE_e])$ (29)

$\mathcal{C}[\![\textbf{Participation of } R \textbf{ to } E \textbf{ is } \text{disjoint, total}]\!] = inSch(R, SCH) \wedge inSch(E, SCH) \wedge E \in entPar(R) \wedge \exists E_i \in entPar(R)\, inSch(E_i, SCH) \wedge \forall c^t \in \mathcal{D}_{VTE}\, \forall c^t \in \mathcal{D}_{TT} (\forall t \in \bigcup_{E_i \in entPar(R)} \varepsilon[\![\textbf{Entity Type } E_i \textbf{ IS\_A } E \cdots]\!] \exists t^1 \in \varepsilon[\![\textbf{Entity Type } E \cdots]\!](t[sE_i] = t^1[sE]) \wedge \forall t \in \varepsilon[\![\textbf{Entity Type } E \cdots]\!] \exists t^1 \in \bigcup_{E_i \in entPar(R)} \varepsilon[\![\textbf{Entity Type } E_i \textbf{ IS\_A } E \cdots]\!](t[sE] = t^1[sE_i]) \wedge \forall E_i, E_j \in entPar(R) \rightarrow \exists t_1 \in \varepsilon[\![\textbf{Entity Type } E_i \textbf{ IS\_A } E \cdots]\!] t_2 \in \varepsilon[\![\textbf{Entity Type } E_j \textbf{ IS\_A } E \cdots]\!](i \neq j \wedge t_1[sE_i] = t_2[sE_j]))$ (30)

$\mathcal{C}[\![\textbf{Participation of } R \textbf{ to } E \text{ is disjoint, partial}]\!] = inSch(R, SCH) \wedge inSch(E, SCH) \wedge E \in entPar(R) \wedge \exists E_i \in entPar(R)\, inSch(E_i, SCH) \wedge \forall c^t \in \mathcal{D}_{VTE}\, \forall c^t \in \mathcal{D}_{TT} (\forall t \in \bigcup_{E_i \in entPar(R)} \varepsilon[\![\textbf{Entity Type } E_i \textbf{ IS\_A } E \cdots]\!] \exists t^1 \in \varepsilon[\![\textbf{Entity Type } E \cdots]\!](t[sE_i] = t^1[sE]) \wedge \forall E_i, E_j \in entPar(R) \rightarrow \exists t_1 \in \varepsilon[\![\textbf{Entity Type } E_i \textbf{ IS\_A } E \cdots]\!] t_2 \in \varepsilon[\![\textbf{Entity Type } E_j \textbf{ IS\_A } E \cdots]\!](i \neq j \wedge t_1[sE_i] = t_2[sE_j]))$ (31)

$\mathcal{C}[\![\textbf{Participation of } R \textbf{ to } E \textbf{ is } \text{overlapping, total}]\!] = inSch(R, SCH) \wedge inSch(E, SCH) \wedge E \in entPar(R) \wedge \exists E_i \in entPar(R)\, inSch(E_i, SCH) \wedge \forall c^t \in \mathcal{D}_{VTE}\, \forall c^t \in \mathcal{D}_{TT} (\forall t \in \bigcup_{E_i \in entPar(R)} \varepsilon[\![\textbf{Entity Type } E_i \textbf{ IS\_A } E \cdots]\!] \exists t^1 \in \varepsilon[\![\textbf{Entity Type } E \cdots]\!](t[sE_i] = t^1[sE]) \wedge \forall t \in \varepsilon[\![\textbf{Entity Type } E \cdots]\!] \exists t^1 \in \bigcup_{E_i \in entPar(R)} \varepsilon[\![\textbf{Entity Type } E_i \textbf{ IS\_A } E \cdots]\!](t[sE] = t^1[sE_i]))$ (32)

$\mathcal{C}[\![\textbf{Participation of } R \textbf{ to } E \textbf{ is } \text{overlapping, partial}]\!] = inSch(R, SCH) \wedge inSch(E, SCH) \wedge E \in entPar(R) \wedge \exists E_i \in entPar(R)\, inSch(E_i, SCH) \wedge \forall c^t \in \mathcal{D}_{VTE}\, \forall c^t \in \mathcal{D}_{TT} (\forall t \in \bigcup_{E_i \in entPar(R)} \varepsilon[\![\textbf{Entity Type } E_i \textbf{ IS\_A } E \cdots]\!] \exists t^1 \in \varepsilon[\![\textbf{Entity Type } E \cdots]\!](t[sE_i] = t^1[sE]))$ (33)

时态数据库参与完整性约束 PIC 有 3 类：快照参与、VT 参与和类型参与。快照参与是时态数据库某一时刻实体的参与度，图 1 在实体与联系的连线上用圆括号表示，如 (a, b)；VT 参与是实体整个 VT 内的参与度，图 1 在实体与联系的连线上用方括号表示，如 $[c, d]$，时态数据库中实体参与度满

（下转第 204 页）

表5 求核与属性约简结果

数据集	核属性	属性约简	运行时间(s)
Hepatitis	1	1,16,6,3,5	1.6991
Heart	——	8,5,4	4.8546
Ionosphere	——	18,24,4	8.1760
Credit	2	2,3,8,6	12.7890
Soybean	17,7,16,12	17,7,16,12,1,22,6,15,8,35,4,9	11.8171

结束语 在不完备决策表的属性约简中，通常定义的差别矩阵是以存储条件属性集为基础的。而本文从不一致对象的角度，给出了一种对象矩阵及其对应的属性约简的定义。证明了该属性约简与基于正区域的属性约简是一致的，并给出一个启发式的属性约简算法。

相对于不完备信息系统中的相容关系，差异关系是从对象的个性出发。针对不完备决策表的属性约简，研究对象矩阵的约简与基于差异关系的约简的关系，是我们下一步的工作。

参 考 文 献

[1] Pawlak Z. Rough Sets [J]. International Journal of Computer and Information Science, 1982, 11(5): 341-356

[2] Pawlak Z. Rough Sets [M]. London, UK: Kluwer Academic Publishers, 1991

[3] Hu X H, Cercone N. Learning in relational databases: a rough set approach[J]. Computational Intelligence, 1995, 11(2): 323-337

[4] 叶东毅，陈昭炯. 一个新的差别矩阵及其求核方法[J]. 电子学报，2002，30(7)：1086-1088

[5] Kryszkiewicz M. Rough set approach to incomplete information systems[J]. Information Sciences, 1998, 112(1): 39-49

[6] Huang Bing, He Xin, Zhou Xian-zhong. Rough Computational Methods Based on Tolerance Matrix [J]. Acta Automatica Sinica, 2004, 30(3): 364-370

[7] 冯朝一，梁家荣，黄柳萍，等. 基于集合覆盖的不完备信息系统属性约简方法[J]. 计算机应用，2006，26(11)：2664-2666

[8] 赵亚鹏，丁以中. 一种不完备信息系统的属性约简算法研究[J]. 计算机应用研究，2008，17(7)：46-48

[9] 周海岩. 采用布尔矩阵不完备信息系统的属性约简[J]. 计算机工程与应用，2010，46(1)：119-121

(上接第 176 页)

足 $c \geqslant a, d \geqslant b$；$p_1$ 是子实体子类化程度[7]，即不相交子类化与重叠子类化；p_2 是子实体参与程度，即全参与和部分参与，min，max 是整型常量。

$$\mathscr{C}[\![\textbf{Entity Type } E_1 \textbf{ Dynamic Extension } E_2 \textbf{ with } T_s]\!] = \wedge_{i=1,2} inSch(E_i, SCH) \wedge \exists ro_1, ro_2 \in Role(\mathscr{T}[\![T_s+1]\!] \rightarrow ro_2 : E_2 \neq \mathscr{T}[\![T_s+1]\!] \rightarrow ro_1 : E_1) \wedge (\mathscr{T}[\![T_s+1]\!] \rightarrow ro_2 : E_2 \in \mathscr{T}[\![T_s]\!] \rightarrow ro_1 : E_1) \quad (34)$$

$$\mathscr{C}[\![\textbf{Entity Type } E_1 \textbf{ Dynamic Evolution } E_2 \textbf{ with } T_s]\!] = \wedge_{i=1,2} inSch(E_i, SCH) \wedge \exists ro_1, ro_2 \in Role((\mathscr{T}[\![T_s+1]\!] \rightarrow ro_2 : E_2 \neq \mathscr{T}[\![T_s+1]\!] \rightarrow ro_1 : E_1) \wedge (\mathscr{T}[\![T_s+1]\!] \rightarrow ro_2 : E_2 \notin \mathscr{T}_s[\![T+1]\!] \rightarrow ro_1 : E_1)) \quad (35)$$

用户自定义变迁规则[10]约束源实体到目标实体的演变，根据演变前、后实体角色是否相容可分为动态扩展和动态进化两类，前者角色相容，如图 1 所示的职工晋升为经理后其角色仍属于职工，用带箭头的虚线表示并标以 DEX；后者角色不相容，从部门经理晋升为高级主管，其角色不再相容，在图 1 中标以 DEV。动态进化不能作用于父实体与子实体，两个不相交实体间的动态扩展即为动态进化。

结束语 基于形式语言理论和形式语义学的指称语义方法，同时参考 TimeER 模型的文本表示法，建立一种形式化时态数据模型 $\mathcal{TDM}$ 及其形式语言模型 $\mathcal{L}(\mathcal{TDM})$。在 $\mathcal{L}(\mathcal{TDM})$ 框架内综合分析时态完整性约束常见的 7 种类型：实体完整性约束、弱实体完整性约束、参照完整性约束、固有完整性约束、用户定义完整性约束、参与完整性约束和变迁约束，并定义了各类时态完整性约束的形式语义规则，深入分析了时态数据模型内在的时态语义联系，为时态数据模型的研究提供了一个便利、高效的形式化理论框架。$\mathcal{L}(\mathcal{TDM})$ 模型各语言成份的独立性与极小性，由 $\mathcal{L}(\mathcal{TDM})$ 构成复杂形式系统的完备性与协调性等元理论性质的分析与论证是下一步研究工作的主要内容，局部性约束和聚集约束等特殊时态完整性约束类型的语义分析也是下一步工作的研究内容。

参 考 文 献

[1] 郝忠孝. 时态数据库设计理论[M]. 北京：科学出版社，2009：5-38

[2] 汤庸，叶小平，汤娜. 时态信息处理技术及应用[M]. 北京：清华大学出版社，2010：16-85

[3] Gregersen H, Jensen C S. Temporal entity-relationship models—a survey[J]. IEEE Transactions on Knowledge and Data Engineering, 1999, 11(3): 464-497

[4] 汤庸，汤娜，毛承洁，等. 时态知识/数据模型研究及应用[J]. 中山大学学报：自然科学版，2004，43(6)：62-66

[5] Gregersen H. The formal semantics of the TimeER model[C]// Proceedings of the Third Asia-Pacific Conference on Conceptual Modelling. Hobart, Australia: Australia Computer Society, 2006: 35-44

[6] Gregersen H, Jensen C S. Conceptual modeling of time-varying information[R]. TR-35. Aalborg, Denmark: TimeCenter Publication, 1998

[7] Combi C, Degani S, Jensen C S. Capturing temporal constraints in temporal ER models[J]. Lecture Notes in Computer Science, 2008, 5231: 397-411

[8] McBrien P. Temporal constraints in non-temporal data modelling languages[J]. Lecture Notes in Computer Science, 2008, 5231: 412-425

[9] Hoang Q, Nguyen T V. Extraction of TimeER model from a relational database[J]. Lecture Notes in Computer Science, 2011, 6591: 57-66

[10] Artale A, Franconi E. Foundations of temporal conceptual data models[J]. Lecture Notes in Computer Science, 2009, 5600: 10-35