一种从关系数据库向 Flogic 本体转换的方法*)

曹泽文 张维明 邓 苏 钱 杰
(国防科技大学信息系统与管理学院 长沙 410073)

摘 要 目前提出的从关系数据库中获取本体的方法主要关注主键之间的相关性,导致抽取的语义信息不全。本文提出一种将关系数据库向 Flogic 表示的本体转换的方法,该方法通过对主键、数据、属性相关性的综合分析,可以抽取出关系数据库中隐含的语义信息,如继承关系及优化结构。

关键词 本体,关系数据库

From Relational Database to FLogic Ontology

CAO Ze-Wen ZHANG Wei-Ming DENG Su QIAN Jie
(College of Information System & Management, NUDT, Changsha 410073)

Abstract The existing approaches of extracting ontology from relational databases can only extract a small subset of semantics embedded within a relational database. In my opinion, the potential source of these problems lies in that the primary focus has been on analyzing key correlation. As an attempt to resolve the problems, we propose a novel approach, which is based on an analysis of key, data and attribute correlations, as well as their combination. Our approach allows the user to extract more semantics from the relational database, including inheritance and optimization structures.

Keywords Ontology, Relational database

1 引言

目前,本体在知识工程、语义 Web、系统建模、异构系统集成等领域得到了广泛的应用[1],越来越多的本体需要构建。但本体的手工构造仍是一项繁琐而辛苦的任务,并最终导致所谓的知识获取瓶颈。因此,从现有主流的关系数据库中采用工程化的方法,自动或半自动地获取本体具有重大的意义。

目前有一些学者已经提出了一些从关系数据库技术获取本体的方法[2~4],但是这些方法存在如下问题:(1)仅能抽取关系数据库中小部分语义信息;(2)要求用户干预,如进行语义标注等,因此自动化程度不高。我们认为,产生这些问题的根源是目前这些方法关注的焦点仅仅是主键之间的相关性,对数据、属性之间的相关性考虑较少。为了解决目前存在的问题,我们提出一种新的方法,通过分析主键、数据、属性以及它们的综合相关性,抽取数据库中的语义信息。为了说明,我们选用 FLogic 作为本体描述语言[5]。

2 关系数据库与本体

关系数据库所基于的数据模型是关系模型[6]。关系模型包括一个有限关系集 R 和一个有限属性集 A,一组主键、外键等。另外,还包括如下一些常用函数:

• 一个函数 $dom(A_i)$,它指定属性 A_i 的取值范围,其中 $A_i \in A$。

• 一个函数 $attr(R_i)$,它指定特定关系 R_i 的属性,其中 $R_i \in R; attr(R_i) \subseteq A$。

• 一个函数 $pkey(Ri)$,它指定一个给定关系 Ri 的主键,其中 $Ri \in R, pkey(Ri) \subseteq attr(Ri)$。

• 一个函数 $fkey(Ri)$,它指定一个给定关系 Ri 的外键,其中 $Ri \in R, fkey(Ri) \subseteq attr(Ri)$。

除了以上实体外,关系模型还包括一些约束,如 PRIMARY KEY, NOT NULL, UNIQUE 等以及一些依赖。所有这些称为关系模式(relation schema),用来描述数据的结构和关联。关系中的元组反映关系模式的值,构成了数据库的内容。

本体结构是一个 5 元组 $O:=\{C, R, H^C, rel, A^O\}$[7]。

• C 和 R 分别是概念标识符和关系标识符构成的集合,C 与 R 不相交。

• H^C 是 C 上的概念分类层次树:$H^C \subseteq C \times C$。$H^C(C_1, C_2)$ 意味着 C_1 是 C_2 的子概念。

• 函数 $rel: R \rightarrow C \times C$ 表示概念之间的非层次联系。$rel(R) = (C_1, C_2)$ 也可写成 $R(C_1, C_2)$。

• A^O 是定义在 C 或 R 上的公理,通常用逻辑程序设计语言表示。

基于以上本体结构,本体包含一个实例集,可以看成是概念的扩展。

本体和关系模型都是一种组织知识的模型,两者之间存在一定的语义相似性。在关系模型中实体以及实体之间的联系都是用关系来描述,所以关系模型中的一个关系可能对应着一个本体的概念或关系。如果数据库中的两个关系存在继承关系,其对应的两个本体概念或关系之间肯定存在层次关系。另外,数据库中的属性约束可以转化成本体中的公理,数据库中的元组可以转化成本体中实例。这种情况还只是关系数据库与本体之间最简单的语义关系。事实上,还存在更复

*)国家自然科学基金(60172012),武器装备预研基金资助项目(51421020904KG01)。曹泽文 副教授,博士生,主研方向为知识系统、决策支持系统等。

杂的语义关系，例如多个关系所描述的信息可以被集成为本体中的一个概念等。

根据上面的分析，可以看出关系数据库与本体之间存在一些语义相似性。因此，利用关系数据库与本体之间的语义相似性从关系数据库中获取本体应该是可行的。

但是，本体与关系数据库模式毕竟是不同的，两者的主要区别如下：在本体的类型系统中，没有基本类型的概念，所有事物都称为概念；本体描述概念时，没有区分概念的属性(attributes)、联系(relationships)，属性、联系都称为性质(property)；概念和性质都可以组织成分类层次。

3 关系的分类

关系数据库中的关系可以分成3类：基本关系、依赖关系、复合关系。

定义1(基本关系) 如果一个关系不依赖于关系数据库模式中的任何其它关系，则该关系属于基本关系。可以将关系模式中的基本关系构成的集合定义为 R_B。这样，基本关系的形式化定义如下：$\forall r\in R$，如果$\neg\exists r_1\in R$，满足 $pkey(r_1)\subset pkey(r)$，则 $r\in R_B$。

例1 下面关系 Department 没有外键，它是一个基本关系。关系 Employee 有属性 departmentID 是一个外键，但该属性不属于主键，因此 Employee 也是一个基本关系。

```
CREATE TABLE Department(
  departmentID INTEGER PRIMARY KEY)
CREATE TABLE Employee(
  employeeID INTEGER PRIMARY KEY,
  departmentID INTEGER REFERENCES Department)
```

定义2(依赖关系) 如果一个关系的主键依赖于另外一个关系的主键，则该关系属于依赖关系。可以将关系模式中的依赖关系构成的集合定义为 R_D。这样，依赖关系的形式化定义如下：

$\forall r\in R$，如果$\exists r_1, r_2, \cdots, r_n (R(n\geqslant 1)$，满足 $pkey(r_1)\subseteq pkey(r_2)\cdots\subseteq pkey(r_n)\subseteq pkey(r)$，则 $r\in R_D$。

例2 下面关系 Identification 是一个依赖关系，因为它的主键属性 employeeID 是另一个关系 Employee 的主键。

```
CREATE TABLE Employee(
  employeeID INTEGER PRIMARY KEY,
  departmentID INTEGER REFERENCES Department)
CREATE TABLE Identification(
  identificationID INTEGER,
  employeeID INTEGER REFERENCES Employee,
CONSTRAINT Identification_PK PRIMARY KEY
(identificationID, employeeID))
```

定义3(复合关系) 如果一个关系既不是基本关系，也不是依赖关系，则该关系属于复合关系。可以将关系模式中的复合关系构成的集合定义为 R_C。这样，复合关系的形式化定义如下：

$\forall r\in R$，如果 $r\notin R_B$，且 $r\notin R_D$，则 $r\in R_C$。

例3 下面关系 Assignment 是一个复合关系，因为它的主键属性 employeeID、projectID 是另外两个关系 Employee、Project 的主键属性的组合。

```
CREATE TABLE Employee(
  employeeID INTEGER PRIMARY KEY)
CREATE TABLE Project(
  projectID INTEGER PRIMARY KEY)
CREATE TABLE Assignment(
  employeeID INTEGER REFERENCES Employee,
  projectID INTEGER REFERENCES Project,
CONSTRAINT Assignment_PK PRIMARY KEY(em-
ployeeID, projectID))
```

因为依赖关系与复合关系的键都由一组属性构成，有时难以对它们进行区分，如例4所示。

例4 关系 Passport 看起来像一个复合关系，因为它的主键由另外两个关系的主键组合而成。但是由于 pkey(Employee)⊆pkey(Idengtification)⊆pkey(Passport)，根据依赖关系的定义，关系 Passport 是一个依赖关系。

```
CREATE TABLE Employee(
  employeeID INTEGER PRIMARY KEY)
CREATE TABLE Identification(
  identificationID INTEGER,
  employeeID INTEGER REFERENCES Employee,
CONSTRAINT Identification_PK PRIMARY KEY
(identificationID, employeeID))
CREATE TABLE Passport(
  employeeID INTEGER REFERENCES Employee,
  identificationID INTEGER REFERENCES Identifica-
tion,
CONSTRAINT Passport_PK PRIMARY KEY(em-
ployeeID, identifica-tionID))
```

4 从关系数据库到本体的转换

从关系数据库到本体的转换包括模式转换与数据移植。

4.1 模式转换

模式转换完成将关系数据库映射成本体结构的过程，具体包括：对主键、数据、属性的相关性的分析；抽取关系数据库的概念模式；将概念模式转换成本体结构。模式转换要求以满足3NF的关系数据库作为输入，处理过程分5步：(1)对关系进行分类；(2)映射关系(relations)；(3)映射属性；(4)映射关系间的联系；(5)映射约束。

4.1.1 映射关系(relations)

规则1 如果一个关系属于基本关系或依赖关系，则它可以直接转换成本体中的概念。

对于复合关系，转换过程比较复杂，根据它的结构的不同，可以转换成概念，有时也可转换成性质。

例如：表1的关系模式中，关系 Employee、Project 都属于基本关系，可以直接转换成概念，假设相应概念也为 Employee、Project，它们都是 Object 概念的子概念。关系 Assignment 是一个复合关系，但它仅仅用来描述两个关系之间的 m:n 联系，不能映射成本体中的概念，其中的主键属性都可以作为另一个属性所引用关系对应概念的性质。这样，得到的本体如表1所示，表中左边是待转换的关系模式，右边是转换得到的本体(下同)，因此可以得到规则2。

规则2 如果一个二元复合关系仅仅描述两个关系之间的 m:n 联系，则它不能映射成本体中的概念，其中的主键属性可以映射成另一个属性所引用关系对应概念的性质。

考虑另外一种情形：表2所示关系 Assignment 是一个复合关系，但它不仅描述两个关系之间的 m:n 联系，而且含有属性 Data，描述员工参加工程项目的开始时间。这时，我们

需要将关系 Assignment 映射成一个概念，相应本体结构如表　2 所示。

表 1　关系模式

CREATE TABLE Employee(employeeID INTEGER PRIMARY KEY) CREATE TABLE Project(projectID INTEGER PRIMARY KEY) CREATE TABLE Assignment(employeeID INTEGER REFERENCES Employee, projectID INTEGER REFERENCES Project, CONSTRAINT Assignment_PK PRIMARY KEY(employeeID, projectID))	Employee::Object. Project::Object. Employee[employeeID =>> INTEGER, projectID =>> Project]. Project[projectID =>> INTEGER, employeeID =>> Employee].

表 2

CREATE TABLE Assignment(employeeID INTEGER REFERENCES Employee, projectID INTEGER REFERENCES Project, startDate DATE, CONSTRAINT Assignment_PK PRIMARY KEY(employeeID, projectID))	Assignment::Object Assignment[employeeID =>> Employee, projectID =>> Project, startDate =>> DATE].

规则 3　如果一个二元复合关系不仅描述两个关系之间的 m:n 联系，而且含有其它属性，则它可以直接映射成本体中的概念。

规则 4　三元及三元以上复合关系，可以直接映射成本体中的一个概念，如表 4 所示。

表 4

CREATE TABLE Uses(employeeID INTEGER REFERENCES Employee, skillID INTEGER REFERENCES Skill, projectID INTEGER REFERENCES Project, CONSTRAINT Uses_PK PRIMARY KEY(employeeID, skillID, projec-tID))	Uses::Object. Uses[employeeID =>> Employee, skillID =>> Skill, projectID =>> Project].

4.1.2　映射属性(attributes)

规则 5　对于一个关系中除外键(含主键中的外键属性)外的所有属性，都可以转换成概念的性质，如表 5 所示。如果外键(含主键中的外键属性)仅仅用来表示关系之间的联系时，通常可以不做处理。对外键(含主键中的外键属性)的具体处理方法见下一节。

表 5

CREATE TABLE Employee(employeeID INTEGER PRIMARY KEY)	Employee[employeeID =>> INTEGER]

4.1.3　映射关系之间的联系(relationships)

在一个关系数据库模式中，联系通常用外键(含主键中的外键属性)表示。根据主键、数据、属性相关性的类型，外键(含主键中的外键属性)可以转换成概念、性质，甚至继承关系。

4.1.3.1　主键、数据集和属性集相关性(correlations)的定义

给定两个关系 $r_1, r_2, r_1 \in R, r_2 \in R$，假定 $K_1 = pkey(r_1)$，$K_2 = pkey(r_2)$，$r_1[K_1] = \prod_{k1} r1$，$r_2[K_2] = \prod_{k2} r2$，$A_1 = attr(r_1) - K_1$，$A_2 = attr(r_2) - K_2$。

定义 4(主键之间相关性)　主键相等($K_1 = K_2$)、主键包含($K_1 \subset K_2$)、主键重叠($K_1 \cap K_2 \neq \varnothing$，$K_1 - K_2 \neq \varnothing$，$K_2 - K_1 = \varnothing$)、主键不相交($K_1 \cap K_2 = \varnothing$)。

定义 5(数据集之间相关性)　数据集相等($r_1[K_1] = r_2[K_2]$)、数据集包含($r_1[K_1] \subset r_2[K_2]$)、数据集重叠($r_1[K_1] \cap r_2[K_2] \neq \varnothing$，$r_1[K_1] - r_2[K_2] \neq \varnothing$，$r_2[K_2] - r_1[K_1] \neq \varnothing$)、数据集不相交($r_1[K_1] \cap r_2[K_2] = \varnothing$)。

定义 6(属性集(非键属性)之间相关性)　属性集相等($A_1 = A_2$)、属性集包含($A_1 \subset A_2$)、属性集重叠($A_1 \cap A_2 \neq \varnothing$，$A_1 - A_2 \neq \varnothing$，$A_2 - A_1 \neq \varnothing$)、属性集不相交($A_1 \cap A_2 = \varnothing$)。

4.1.3.2　对键、数据集和属性集相关性进行分析

对于两个存在关联的两个关系 $r_1, r_2, r_1 \in R, r_2 \in R$，通过分析主键、数据集、属性集之间的相关性，可以得到相应的转换规则。为了清楚说明，我们通过给定的不同情形，分别说明转换方法与理由。

情形 1　主键不相交($K_1 \cap K_2 = \varnothing$)。表 6 中的两个关系，其主键不相交，可以直接将两个关系转换成概念，外键转换成性质。相应转换方法如表 6 所示。

情形 2　主键相等、数据集相等、属性集不相交。表 7 的两个关系，满足主键相等、数据集相等、属性集不相交，说明这两个关系是一个逻辑关系的垂直分割，应该将这两个关系转换成一个概念，合并其属性。相应转换规则如表 7 所示。

情形 3　主键相等、数据集不相交、属性集相等。表 8 中的两个关系，满足主键相等、数据集不相交、属性集相等，说明这两个关系是一个逻辑关系的水平分割，应该将这两个关系合并为一个关系，并转换成一个概念。相应转换规则如表 8 所示。

表 6

CREATE TABLE Project(projectID INTEGER PRIMARY KEY) CREATE TABLE Task(taskID INTEGER PRIMARY KEY, projectID INTEGER REFERENCES Project)	Project[projectID =>> INTEGER, taskID =>> Task]. Task[taskID =>> INTEGER, projectID =>> Project]

表 7

CREATE TABLE Project(projectID INTEGER PRIMARY KEY, budget FLOAT, dueDate DATE) CREATE TABLE SoftwareProject(projectID INTEGER PRIMARY KEY, language VARCHAR)	SoftwareProject[projectID =>> INTEGER, budget =>> FLOAT, dueDate =>> DATE, language =>> STRING].

表 8

CREATE TABLE Project(projectID INTEGER PRIMARY KEY, budget FLOAT, dueDate DATE, language VARCHAR) CREATE TABLE SoftwareProject(projectID INTEGER PRIMARY KEY, budget FLOAT, dueDate DATE, 1 anguage VARCHAR)	SoftwareProject[projectID =>> INTEGER, budget =>> FLOAT, dueDate =>> DATE, language =>> STRING].

情形 4 主键相等、数据集包含。表 9 中的两个关系，满足主键相等、数据集包含，即 SoftwareProject 的所有数据都包含在 Project 中，说明这两个关系的联系是单向继承，因此将这两个关系转换为两个概念，而且概念之间存在继承关系。相应转换规则如表 9 所示。

情形 5 主键相等、数据集重叠、属性集不相交。表 10 中的两个关系，满足主键相等、数据集重叠、属性集不相交，两个关系中存在许多公共数据，说明 SoftwareProject 与 HardwareProject 存在多元继承关系。因此，我们可以建立一个子概念，例如 SoftwareHardwareProject，SoftwareProject 与 HardwareProject 都是它的父概念。相应转换规则如表 10 所示。

表 9

CREATE TABLE Project(projectID INTEGER PRIMARY KEY, budget FLOAT, dueDate DATE) CREATE TABLE SoftwareProject(projectID INTEGER PRIMARY KEY REFERENCES Project, language VARCHAR)	Project[projectID =>> INTEGER, budget =>> FLOAT, dueDate =>> DATE]. SoftwareProject::Project[language =>> STRING].

表 10

CREATE TABLE HardwareProject(projectID INTEGER PRIMARY KEY, supplier VARCHAR) CREATE TABLE SoftwareProject(projectID INTEGER PRIMARY KEY, language VARCHAR)	HardwareProject::Project[projectID =>> INTEGER, supplier =>> STRING]. SoftwareProject::Project[projectID =>> INTEGER, language =>> STRING]. HardwareSoftwareProject::Object. HardwareSoftwareProject::HardwareProject. HardwareSoftwareProject::Soft wareProject.

情形 6 主键相等、数据集不相交、属性集重叠。表 11 中的两个关系，满足主键相等、数据集不相交、属性集重叠，两个关系中存在许多公共属性，即 SoftwareProject 与 HardwareProject 存在单值继承关系。因此，我们可以建立一个父概念，例如 Project，SoftwareProject 与 HardwareProject 都是它的子概念。相应转换规则如表 11 所示。

表 11

CREATE TABLE HardwareProject(projectID INTEGER PRIMARY KEY, budget FLOAT, dueDate DATE, supplier VARCHAR) CREATE TABLE SoftwareProject(projectID INTEGER PRIMARY KEY, budget FLOAT, dueDate DATE, language VARCHAR)	Project::Object. Project[projectID =>> INTEGER, budget =>> FLOAT, dueDate =>> DATE]. HardwareProject::Project[supplier =>> STRING]. SoftwareProject::Project[language =>> STRING].

情形 7 主键重叠、数据集重叠(或包含)。表 12 中的两个关系，满足主键重叠、数据集重叠(或包含)，即两个关系中存在许多公共属性和数据，说明这两个关系是钻石型继承关系。因此，我们可以建立一个子概念和一个超概念，假设分别为 HardwareSoftwareProject 和 Project，SoftwareProject 与 HardwareProject 都是 Project 的子概念，HardwareSoftwareProject 是 SoftwareProject 与 HardwareProject 的子概念。相应转换规则如表 12 所示。

情形 8 主键相等、数据集重叠、属性集重叠。表 13 中的两个关系，满足主键相等、数据集重叠、属性集重叠，因为 Project 关系中的 DepartmentID 信息都包含在 Employee 关系中，说明丢失了一个关于 DepartmentID 信息的关系。因此，我们可以建立一个新概念 Department，为它添加相应属性；同时，Project 与 Employee 不再直接发生联系，而是通过新概念的 DepartmentID 性质建立联系。相应转换规则如表 13 所示。

表 12

CREATE TABLE HardwareProject(projectID INTEGER PRIMARY KEY, budget FLOAT, dueDate DATE, supplier VARCHAR) CREATE TABLE SoftwareProject(projectID INTEGER PRIMARY KEY, budget FLOAT, dueDate DATE, language VARCHAR)	Project::Object. Project[projectID =>> INTEGER, budget =>> FLOAT, dueDate =>> DATE]. HardwareProject::Project[supplier =>> STRING]. SoftwareProject::Project[language =>> STRING]. HardwareSoftwareProject::Object. HardwareSoftwareProject::HardwareProject. HardwareSoftwareProject::SoftwareProject.

表 13

CREATE TABLE Employee(employeeID INTEGER, departmentID INTEGER, CONSTRAINT Employee_PK PRIMARY KEY(employeeID, departmen-tID)) CREATE TABLE Project(projectID INTEGER, departmentID INTEGER, CONSTRAINT Project_PK PRIMARY KEY(projectID, departmentID))	Department::Object. Department[departmentID =>> INTEGER, employeeID =>> Employee, projectID =>> Project]. Employee[employeeID =>> INTEGER, departmentID =>> Department]. Project[projectID =>> INTEGER, departmentID =>> Department].

4.1.4 映射约束

在关系数据库的 SQL 语言中，可以指定 PRIMARY KEY，UNIQUE，NOT NULL 等约束。我们可以根据这些约束的定义信息直接将它们转换成本体的公理。例如，根据约束 PRIMARY KEY 的定义：一个属性是主键，它必须是取值唯一(unique)并且必须有值(total)。一个属性取值唯一(unique)，要求关系中的元组在该属性的取值不会重复，并且也不允许元组在该属性上存在两个值。一个属性必须有值(total)，指关系的所有元组在该属性上有值。这样，可以得到相应公理，如表 14 所示。

4.2 数据移植

数据移植过程是将关系数据库中的数据移植到本体实例库的过程。具体来说，是将关系数据库的元组映射成本体的实例。这个过程分成两步：

- 创建本体实例，为每一个实例指定一个唯一的标识符；
- 建立本体实例之间的关系。

例如，表 15 中可以根据下面两个关系中的元组建立两个实例，分别为 d、e，两个本体实例之间存在引用关系。

表 14

CREATE TABLE Employee(EmployeeID INTEGER PRIMARY KEY)	PrimaryKey(Employee, employeeID)。 Forall C, P PrimaryKey(C, P) <-Unique(C, P) and Total(C, P)。 Forall C, P Unique(C, P) <-IdenticalValues(C, P) and SingleValue(C, P)。 Forall C, P IdenticalValues(C, P) <-Forall I1, I2, IP I1:C and I2:C and I1[P->>IP] and IC[P->>IP] and Equal(I1, I2)。 Forall C, P SingleValue(C, P) <- Forall IC, IP1, IP2 IC:C and IC[P->>IP1] and IC[P->>IP2] and Equal(IP1, IP2)。 Forall IC Exists IP IC:C and IC[P->>IP]。

表 15

INSERT INTO Department(departmentID) VALUES(1) INSERT INTO Employee(employeeID, departmentID) VALUES(2,1)	d:Department[departmentID ->> 1]. e:Employee[employeeID ->> 2, departmentD ->> d].

5 相关工作分析

目前已有许多学者研究从关系数据库中抽取实体关系和对象模型，但是这些工作不能直接用来研究从数据库中抽取本体。因为关系模式与本体之间存在许多不同之处，这种不同之处产生的原因有 3 点：

- 本体模型表达能力更丰富，例如 FLogic 允许将类的性质组织成层次结构，描述继承关系。
- 本体提供更多的语义信息，而在关系数据库中语义信息不仅包含在关系模式中，也可以包含在数据中。
- 本体中类与实例的界限比较模糊。例如，本体中的 meta-class，其实例是其自身，使得区分本体与本体实例比较困难。

目前 Stojanovic 等学者提出了一些从关系数据库中抽取本体的方法，下面将它们与本文方法进行比较。

Stojanovic 等提出的方法[2]：该方法与本文方法非常接近，但是该方法只分析了数据集相等或包含时的情况，因此它只能抽取出关系数据库中的部分语义关系。当数据集之间存在重叠关系或不相交关系时，通过本文的方法，能够发现关系数据库中隐藏的语义信息。

(下转第 206 页)

会话，165 个网页。

3.2 聚类结果

先在用户会话上应用 FCM 算法来考察信息熵的变化情况，从而与本文算法的信息熵做对照。权重指数取 2，迭代次数为 500，终止误差为 0.000001，表 1 为聚类数分别为 6～10 时得到的信息熵。

表 1 应用 FCM 算法不同类得到的信息熵

C	6	7	8	9	10
E	0.11204	0.12329	0.08994	0.10131	0.09409

再将本算法应用在用户会话上，表 2 为多次试算后理想初始参数取值。

表 2 初始参数设置

iniAbNum	N	Qd	Qs	Qn
30	10	0.036	0.1	0.2

免疫网络聚类算法区别于一般的基于划分的聚类算法，它不是将每一个对象归属到一个类别，而是经算法学习获得能代表类别特征的记忆抗体。这样，算法最终得到 11 个记忆抗体，按照同类别标记输出该类抗体，结果有 7 个类别，即聚类数为 7，信息熵为 0.0178197，远小于 FCM 算法聚类数为 6～10 的信息熵，证明本算法得到的同一聚类中的记忆抗体相似程度高，聚类效果好。

表 3 免疫网络算法聚类结果

类别	用户访问的 URL 类型
1	{main014 *. htm, guanlijigou/xiaoban/ *, yuanxishezhi/maofa/maoxing/ * }
2	{main013 *. htm, main02 *. htm, zhaoshengxinxi/ * }
3	{main05 *. htm, guanlijigou/kejiqiye/ *, yuanxishezhi/zcgl/index. asp, guanlijigou/kejiqiye/ *, guanlijigou/waishichu/ *, english/ * }
4	{main016 *. htm, xuexiaogaikuang/ *, guanlijigou/kejiqiye/zhidu/ * }
5	{main012. htm, guanlijigou/waishichu, guanlijigou/shiyuan/ * }
6	{main0 *. htm, tuanxuezuzhi/xstxs/ *, guanlijigou/chengshijianshe/ * }
7	{main0 *. htm, guanlijigou/chengshijianshe/ch/ *, yuanxishezhi/maofa/maoxing/new/ * }

表 3 为从用户会话中提取出的用户访问的模式，即用户访问的 URL 类型。表中只给出相同部分的链接，不同部分以 * 号代之，* 号也表示 0 到多个字符（实验中的 URL 则为每一个详细的网页地址）。

我们用 KNN 算法（K-Nearest Neighbor Algorithm）来验证聚类结果是否反映了用户会话的特征。从 56 个用户会话中随机选取 3 个会话作为检验抗体，在这 3 个会话向量中，根据其中为“1”的网页标识统计出它们分别代表了多次访问招生信息和偶尔访问学校概况、多次访问外事处和英语信息以及主要访问贸行学院和偶尔访问招生信息的 3 种用户浏览行为。分别计算 3 个检验抗体同 11 个记忆抗体的距离，距离最近的记忆抗体的类别作为检验抗体的类别。因为这 7 个类是由抗体不断学习原始用户会话的结构和分布特征抽象出的，相对于其它类，类别 2、类别 3 和类别 1 更能反映出这 3 个会话的特征，故将 3 个会话分别划归到这 3 个类中。

结束语 针对 Web 日志数据自身特点，提出免疫网络聚类算法。与一般的聚类算法不同，免疫网络聚类算法不需要具体确定用户会话所属的类别，而是利用免疫系统自身的自组织、自适应的特性，让带有类别标记的抗体学习抗原特性并提取其分布特征，使之尽可能地反映一个类的结构特点，用这些能代表一个类的较少的几个典型的抗体来概括出一个类的特征。通过在一个真实的网站日志数据集上进行实验，结果表明该算法是有效的。

参 考 文 献

1 董一鸿，庄越挺. 基于新型的竞争型神经网络的 Web 日志挖掘[J]. 计算机研究与发展，2003，40(5)：661～667
2 Krishnapuram R, Joshi A, Nasraoui O, et al. Low-Complexity Fuzzy Relational Clustering Algorithms for Web Mining. IEEE Trans. on Fuzzy Systems, 2001, 9(4): 595～607
3 宋擒豹，沈钧毅. Web 日志的高效多能挖掘算法[J]. 计算机研究与发展，2001，38(3)：328～333
4 刑东山，沈钧毅，宋擒豹. 从 Web 日志中挖掘用户浏览偏爱路径[J]. 计算机学报，2003，26(11)：1518～1523
5 王实，高文，李锦涛，等. 路径聚类：在 Web 站点中的知识发现[J]. 计算机研究与发展，2001，38(4)：482～486
6 De Castro L, Vou Zuben F J. AiNet: an artificial immune network for data analysis. In: Data Mining: A Heuristic Approach, H. A. Abbass, R. A. Sarker, C. S. Newton, eds. Idea Group Publishing, USA, Chapter XII, 2001. 231～259
7 李涛著. 计算机免疫学[M]. 北京：电子工业出版社，2004
8 Hang Xiaoshu, Dai Honghua. An immune network approach for Web document clustering [A]. In: Proc. of the IEEE/WIC/ACM International Conf. on Web Intelligence [C]. Beijing: China, 2004. 278～284
9 郭岩，白硕，于满泉. Web 使用信息挖掘综述[J]. 计算机科学，2005，32(1)：1～7

（上接第 153 页）

Kashyap 提出的方法[3]：该方法与 Stojanovic 等提出的方法类似，而且它要求用户进行语义标注工作，抽取语义信息时的自动化程度不高，该方法无法产生公理。

Dogan 和 Islamaj 提出的方法[4]：该方法提供了简单、自动的模式转换和数据映射过程，在该方法中一个关系映射成一个概念，其中的属性映射成概念的性质，一个关系的元组映射成本体的一个实例。该方法没有考虑继承关系，不能对关系数据库的结构进行优化，因此所建立的本体看起来很像关系。

结论 本文提出了一种新的从关系数据库中获取本体的方法。通过对主键、数据、属性的相关性的分析，允许用户从关系数据库中抽取更多的语义信息，包括继承关系，甚至可以对结构进行优化；减少用户干预，仅仅要求确认抽取的语义信息，为它们命名。以后可以将该方法应用于关系数据库向语义网的转换。

参 考 文 献

1 Studer R, Benjamins V R, Fensel D. Knowledge engineering, principles and methods. Data and Knowledge Engineering, 25(1-2): 161～197
2 Stojanovic L, Stojanovic N, Volz R. Migrating data-intensive Web Sites into the Semantic Web. In: Proc. of the 17th ACM symposium on applied computing. 2002
3 Kashyap V. Design and creation of ontologies for environmental information retrieval. In: Proc. of the 12th Workshop on Knowledge Acquisition, Modeling and Management. 1999
4 Dogan G, Islamaj R. Importing Relational Databases into the Semantic Web. http://www.mindswap.org/webai/2002/fall/Importing_20Relational_20Databases_20into_20the_20Semantic_20Web.html (2002)
5 F-Logic. http://www.informatik.uni-freiburg.de/~dbis/Publi-cations/95/flogic-jacm.html
6 Codd EF. A relational model of data for large shared data banks. CACM, 1970, 13(6)
7 Maedche A. Ontology learning for the Semantic Web. Boston: Kluwer Academic Publishers, 2002
8 Astrova I. Reverse Engineering of Relational Databases to Ontologies. In: Proceedings of the 19th ACM Symposium on Applied Computing (SAC), 2004