基于Ontology的数据库自然语言查询接口的研究

李 虎[1] 田金文[1] 王缓缓[2] 石 勇[1,3]
(华中科技大学图像识别与人工智能研究所 武汉 430074)[1]
(三峡大学经济与管理学院 宜昌 443002)[2] (武汉通信指挥学院 武汉 430070)[3]

摘 要 提出了一种基于Ontology的关系数据库自然语言查询接口的系统模型及设计框架。采用WordNet作为基本数据库并在WordNet之上定义领域词库,可以提高语法分析的识别率;同时利用Ontlogly知识表达能力存储关系数据库概念模型,并对概论模型的内容进行扩充;另外对Ontology和WordNet的同义词集进行关联,可以提高语义的识别率。用户的输入查询语句通过语法分析、语义分析生成中间表达式语言DRS,然后通过模板技术转换成SQL,通过DBMS执行SQL并返回结果。实验证明,这种方案不但实用可行,而且通过逐步完善Ontology知识库的定义,可以大大提高查询的命中率;另外通过WordNet和Ontology定义领域词库和领域知识,提高了系统的可移植性。最后,所提供的方法可以很容易地移植到其他领域。

关键词 本体,数据库自然语言查询接口,篇章表述结构,数据库管理系统,SQL,OWL
中图法分类号 TP391.1 **文献标识码** A

Ontology-based Natural Language Interface to Relational Databases

LI Hu[1] TIAN Jin-wen[1] WANG Huan-huan[2] SHI Yong[1,3]
(Institute of Pattern Recognition & Artificial Intelligence, Huazhong University of Science & Technology, Wuhan 430074, China)[1]
(School of Economic & Management, Three Gorgeses University, Yichang 443002, China)[2]
(Wuhan Communication Command Academy, Wuhan 430070, China)[3]

Abstract This paper proposed an approach to creating an ontology-based natural language interface to relational databases which allows end-users to access and query information in database with a natural language. In order to strengthen the reliability and the user-friendliness of the system, our approach integrates WordNet as the base lexicon and ontology as the knowledge base of the semantic interpreter. The user inputs will be first translated into a syntax tree by syntax analysis and parsing, then it will be translated into an intermediate expression language DRS by the ontology-based semantic interpreter. The SQL generator transfers DRS expressions into SQL sentences by a template technique. The DBMS executes SQL sentences and returns the final result. This paper presented the framework based on ontology techniques to implement portable NLIDBs that makes it easier to migrate from one domain to another. The prototype system and our experiments show that our approach has generated good results with excellent performance on common queries.

Keywords Ontology, NLIDBs, DRS, DBMS, SQL, OWL

1 引言

数据库自然语言接口(NLIDBs)是AI领域智能接口技术的一个研究课题。NLIDBs允许最终用户通过自然语言向数据库系统发送请求,然后得到查询结果。与其他查询方式相比,NLIDBs用户只需知道领域内的概念,无需知道数据库系统本身的结构和数据库的操作语言;用户不需要或者只需要简单培训就能直接查询数据库。

从20世纪60年代末至今,在NLIDBs研究领域已经有大量的文献介绍它的原理及实现方案。Androutsopoulos[1]介绍了4种主要的NLIDBs架构方案:第一种是基于模式匹配的架构,它也是第一个应用到NLIDBs的技术,典型案例有SAVVY[2]等;第二种是基于中间描述语言的架构,通过中间描述语言转换成SQL,典型案例有MASQUE/SQL[3]等;第三种是基于语法的框架,自然语言查询语句通过句法分析得到语法树,通过语法树转换成SQL,典型案例有LUNAR[4]等;最后一种是基于语义语法的框架,自然语言查询语句通过语义分析得到语义树,然后转换成SQL,此方案与语法分析的不同在于用语义范畴替代语法概念,典型案例有PLANES[5]等。

本文在前人研究的基础上提出了基于Ontology的数据库自然语言访问接口系统模型和设计框架。本模型具备以下特点:

1) 采用Ontology组织NLIDBs的知识库,包括关系数据

到稿日期:2009-07-20 返修日期:2009-09-27

李 虎(1978—),男,博士生,主要研究方向为自然语言处理、模式识别等,E-mail:2002_ahu@126.com;田金文(1961—),男,教授,博士生导师,主要研究方向为遥感图像处理、小波分析、图像压缩、计算机视觉和分形几何等;王缓缓(1978—),女,博士生,讲师,主要研究方向为管理系统模拟;石 勇(1978—),男,博士生,讲师,主要研究方向为人工智能、信息安全等。

库的概念模型和领域知识业务模型。Ontology 是从哲学范畴借用的一个概念，其目标是捕获相关的领域知识，提供对该领域知识的常规理解，确定该领域内共同认可的词汇，并从不同层次的形式化模式上给出这些词汇和词汇之间相互关系的明确定义。OWL[6]语言是由 W3C 组织定制用来定义和实例化 Ontology 以及开发 Ontology 应用的标准语言。本文最大限度地利用 Ontology 来消除自然语言的歧义，提高用户自然语言查询语句的命中率和准确率，从而提高 NLIDBs 的可信度。通过采用 Ontology 技术，可以有效分离 NLIDBS 的执行算法和领域知识，从而可以提高 NLIDBs 系统的可移植性。

2）采用 WordNet[7]组织领域词库。WordNet 是美国普林斯顿大学的心理学家、语言学家和计算机工程师基于可分离性假设、模式假设和广泛性假设联合设计的一种基于认知语言学的英语词网，是按照单词的意义组成的一个“单词的网络”。本文按照 WordNet 的组织结构来定义领域内词库，并利用 WordNet 的同义、反义等关系与 WordNet 进行关联。这样，我们的 NLIDBs 系统就能够利用 WordNet 的同义词集关系来识别用户输入的非领域词库的单词，然后再映射到领域词库中的单词，从而提高自然语言查询的命中率和准确率。

3）采用篇章表述结构（DRS[8]）作为 NLIDBs 的中间描述语言。篇章表述理论（DRS[8]）是动态的自然语言涵意的形式语义结构，由篇章语义的构造算法和语义的正确性验证两部分组成，它是在 MG（Montague Grammar）的基础上建立起来的，而又克服了 MG 的局限性的语义理论；本文采用 DRS 作为 NLIDBs 的中间描述语言，可以很好地表述语义。

4）采用模板技术实现从 DRS 到 SQL 的转换。模板技术在现代软件开发中占用重要地位，可以很好地实现内容与形式或样式的分离。在本文中，样式由 SQL 语法决定，内容由 DRS 提供。

2 系统模型及总体设计

NLIDBs 系统模型及框架（如图 1 所示）由两个模块组成：语言处理模块和数据库处理模块。这两个模块通过中间表示语言 DRS 关联起来。DRS 基于 Ontology 在较高层次表示查询语句的语义，在此处自然语言查询语句转换成 DRS 逻辑查询语句，然后转换成 SQL 查询语句。采用此框架的好处是从逻辑上和物理上分离了语言处理部分和数据库处理部分，为此系统移植到其他系统提供了概念上的保障。

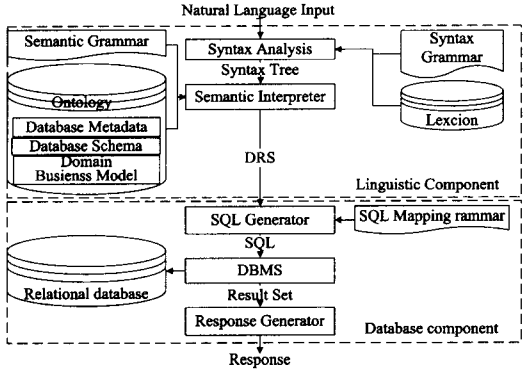

图 1 NLIDBs 系统结构

语言处理模块主要负责语言本身的处理，包括切词、单词变体处理、句法分析和语义解析等。此模块提供友好的用户接口，用户通过此接口输入查询语句；句法解析器根据句法语法和领域词库解析查询语句，生成语法树；然后语义解释器根据 Ontology 知识库和语义语法解释语法树，生成 DRS。DRS 表达式是语言处理模块的最终输出，DRS 应包含后面数据库操作部分的所有语义信息。

数据库处理模块负责和数据库管理系统进行交互。此模块接收语言处理模块的输出 DRS，通过模板技术转换成 SQL，然后调用数据库管理系统执行 SQL 语句，得到查询记录集，最后格式化记录集返回给用户。这样就完成了一个自然语言查询过程。

定义 1 NLIDBs 系统模型 Σ，用一个六元组表示：

$\Sigma=\{S,SG,L,Ont,DCG,SQLT\}$；

NLIDBs 处理过程定义为下面的公式：

$ST=SP(S,SG,L)$；

$DRS=SI(ST,Ont,DCG)$；

$SQL=SQLG(DRS,SQLT)$；

$R=DBMS(SQL)$。

其中，ST 表示语法树，SP 表示语法解析器，S 表示输入的自然语言查询句子，L 表示词库，DRS 表示查询句子的篇章表述结构，SI 表示语义解析器，Ont 表示 Ontology 知识库，DCG 表示语义语法，SQL 表示数据库查询语句，SQLG 表示 SQL 生成器，SQLT 表示 SQL 模板，DBMS 表示数据库管理系统，R 表示自然语言查询结果。

3 语言处理模块

3.1 选用句法解析器

句法分析是语言处理模块的子模块，其主要功能是根据词库（包括通用词库和领域词库）和句法语法对自然语言查询句子进行句法分析，得到语法树。目前可以选用的英语句法分析工具较多，本文选用基于概率统计的 Stanford Parser[14]为句法解析器。选择此句法分析工具的理由如下：

1）Stanford Parser 集成多个解析算法于一身，具备相对完善的自然语言解析工具，其中包括 PCFG 解析器、词库 PCFG 解析器和内容依赖解析器；

2）Stanford Parser 提供对中文句子进行解析的功能，方便迁移到中文数据库系统；

3）它是免费开源软件，可以自由下载。

3.2 构造领域词库

词库由 3 部分组成：实义词（名词、动词、形容词、副词），预定义功能词（冠词、限定词、介词、代词）和变体词（所有非规则的变体词）。目前有很多类型的词库资源可以选择，比如 WordNet，VerbNet，Brown Corpus，Oxford Dictionary 等。可以基于上面的词库作为通用词库，根据具体的应用领域需求，通过工具扩展领域词库。

本文选用 WordNet 为通用词库，并在 WordNet 基础上扩展领域词库。之所以选用 WordNet，是因为它是一个面向语义的英语词库，单词按照一定的语义组成一个语义网络。本文按照 WordNet 的组织结构定义领域内词库，并利用 WordNet 的同义、反义等关系与 WordNet 进行关联。这样，本 NLIDBs 系统就能够利用 WordNet 的同义词集关系来识别用户输入的非领域词库的单词，然后映射到领域词库中的单词，从而提高自然语言查询的命中率和准确率。比如，查询

语句“Who is the writer of the book ‘Gone with the Wine’?”和“Who is the author of the book ‘Gone with the Wine’?”应该得到相同的查询结果。我们可以定义 writer 和 author 为同义词，用同一个 Synset Id 表示，在语法分析时通过简单的同义映射得到同义 DRS 输出，从而实现查询结果相同。

在 WordNet 的 Prolog 版本中，语义关系通过一组 Synset Id 表示。每个语义采用独立的文件维护，目前有 21 个词库文件。在定义领域词库时，并不是所有的关系我们都需要，因此可以根据领域要求进行裁减。本文仅仅定义 5 个关系，分别是 domain_s. pl，domain_g. pl，domain_ant. pl，domain_fr. pl，domain_hyp. pl。另外定义一个关联文件把 WordNet 词库和领域内词库关联起来。具体词库定义如表 1 所列。

表 1　领域词库表

WordNet	领域	谓词结构	描述
wn_s. pl	d_s. pl	s(s_id,w_num,word,ss_type,sense_number,tag_count)	同义词关系
wn_g. pl	d_g. pl	g(s_id,gloss)	词义解释
wn_hyp. pl	d_hyp. pl	hyp(s_id,s_id)	上下文关系
wn_ant. pl	d_ant. pl	ant(s_id,w_num,s_id,w_num)	反义关系
wn_fr. pl	d_fr. pl	fr(s_id,f_num,w_num)	动词框架
wn_exc. pl	d_exc. pl	exc(word,morph,ss_type)	单词变体
—	d_syn. pl	syn(domain_s_id,wn_s_id)	关联领域单词与 WordNet 单词

3.3　构造本体知识库

3.3.1　关系数据库 E-R 模型

关系数据库是建立在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据。E-R 模型是运用真实世界中的事物与关系的概念来解释数据库中抽象的数据框架。在关系数据库中，用实体或实体之间的关系来组织数据，通过增加、删除、修改、查询来操作关系数据库。

定义 2　关系数据库概念视图可以通过下面 E-R 模型来表示：

ER=(E,R)

E=(N,A,PK,U)

A=(N,DT,D,R)

R=(N,E1,E2,FK)

此处 E 表示实体，R 表示关系，A 表示属性，DT 表示属性值类型，D 表示领域，PK 表示实体的主键，FK 表示外键，U 表示约束值，N 表示对应的实体、属性或关系的名称。

3.3.2　抽取 E-R 数据库模型到 OWL

OWL 语言是由 W3C 组织定制，用来定义和实例化 Ontology，并开发 Ontology 各种应用的标准语言。OWL 包含一系列的标注、公理和事实及标注记录作者身份等其他信息；OWL 主要内容通过 classes，properties 和 individuals 的方式存储在公理和事实中。OWL 目前有 3 个子版本：OWL-Lite，OWL-DL 和 OWL-Full。本文采用 OWL-DL 来描述数据库 E-R 模型。

定义 3　对给定的关系数据库 E-R 模型，根据以下抽取规则可以获取关系数据库的 Ontology 模型。

Rule 1：定义 OWL 模型。一个 E-R 模型表示一个完整 Ontology 模型，我们定义一个独立的 OWL 文件表示此模型。

Model(ER) → Ontology(owl)

Rule 2：抽取实体。实体表示领域内某个具体的事实，抽取实体并填充到 OWL 的 owl:Class 中。

Model(E) → Ontology(owl:Class)

Rule 3：抽取属性。属性在 E-R 模型中表示实体的特性，抽取属性并填充到 OWL 的 owl:DatatypeProperty 中，其中属性的数据类型对应 rdfs:domain，取值范围对应 rdfs:range。

Model(A) → Ontology(owl:DatatypeProperty,rdfs:domain,rdfs:range)

Rule 4：抽取关系。关系在 E-R 模型中表示实体之间的关系，抽取关系并填充到 OWL 的 owl:ObjectProperty 中。

Model(R) → Ontology(owl:ObjectProperty)

Rule 5：抽取业务数据。数据库中存储的部分业务数据会在查询语句中出现，抽取这些数据填充到 OWL 的 Individuals。

Data(A) → Ontology(Individuals)

按照上面的 E-R 模型到 Ontology 模型的转换步骤，可以构造出一个基本的 Ontology 模型。此模型已经具备满足用户查询的基本信息，用户可以通过构造一个具备完整信息内容的查询语句完成查询任务。比如一个图书出版系统，定义了 book 实体，有属性字段 author 和 title，若用户想查询某本书的作者是谁，则查询语句为“Who is the author of the book whose title is ‘Gone With the Wind’?”包含了后续语义分析所要的所有信息。事实上，由于自然语言的特性，用户的查询请求一般不是这么直观，不能通过直接匹配的方式找到查询所需的信息。比如上面同样的查询需求，可以采用查询语句“Who is the author of the book ‘Gone With the Wind’?”，很明显这句查询语句包含了实体 book、实体的属性字段 author 和值“Gone With the Wind”，但并没有指定查询条件字段 title。如果要成功得到查询结果，必须在值“Gone With the Wind”和 title 之间建立关系。

所以，为了扩大 NLIDBs 接收查询语句的范围，提高查询语句的命中率，需对 Ontology 的知识库进行扩展，添加相关的数据信息。

定义 4　扩展 Ontology 知识库可以有效地提高系统接收查询语句的范围，提高查询语句的命中率和正确率。本文提出按照下面规则来扩展 Ontology 知识库。

Rule 1：定义同义关系。由于我们采用并扩展 WordNet 作为词库，因此可以通过定义 OWL 的 dc:identifier 关联 WordNet 的名词 Synset Id。

Rule 2：定义字段关联动作。可以通过定义 OWL 的 dc:identifier 关联 WordNet 的动词 Synset Id。

Rule 3：定义实体默认关联的字段。比如实体 book 的默认关联字段 title，由‘Gone with the Wine’修饰 book 则可以推理得出‘Gone with the Wine’是 title 的值。添加实体的属性实现 owl:DatatypeProperty。

Rule 4：扩展字段属性。此属性的取值反问是[p,l,t,r,a]，其中 p 表示人，l 表示位置，t 表示时间信息，r 表示原因，a 表示物品。

定义 5　通过定义 3 和定义 4，可以定义 Ontology 的元素：

ontClass(C,[DP],[NSyn],[VSyn],DefaultDP)

ontDatatypeProperty(DP,C,[NSyn],[VSyn],Ext,R)

ontObjectProperty(OP,[NSyn],[VSyn])

ontIndividuals(D,DP)

3.4 语义分析

语义分析是把语法树对应的句子内容转换成篇章表述结构 DRS。DRS 作为语义处理的中间语言，可以根据需要转换成其他的知识表示方式，比如描述逻辑(DL)、一阶谓词逻辑(FOL)等。本文目标是把 DRS 转换成 SQL，然后传递给 DBMS 进行查询处理。

3.4.1 查询句子类型

NLIDBs 接口主要回答用户的查询问题，每一个问题对应一个或者多个查询子句。查询子句可以划分为 4 种类型。

类型 1：YesNoQuery，此查询的回答是 yes 或者 no，比如：

Is the book wrote by Hemingway?

类型 2：WhQuery，此查询的回答含有具体的内容，即所谓的“wh-words”，包括{what，why，where，who，when} 5 种子类型，分别表示{什么，为什么，哪里，谁，什么时候}，比如：

Who is the author of ‘The Old Man and Sea’?

类型 3：CommandQuery，此查询语句通常为动词短语，查询结果同 WhQuery，比如：

Get the books write by Hemingway.

类型 4：HybirdQuery，此查询表示一个句子中包含多个查询语句，比如：

Who is the author of ‘The Old Man and Sea’ and what other books did he write?

3.4.2 定义 DRS 结构

DRS 用一个扁平标注结构表示一个逻辑原子。例如，图书管理系统中的 book A 实例，用 book(A)表示，在此处采用 table(A，book)的形式表示一个 DRS 结构，这样可以采用少量定义的谓词结构表示大量的扁平结构的谓词，降低谓词的数量，同时方便后续的处理。

SQL 语句是非过程语言，其基本的查询语法结构如下：SELECT c1，c2，… FROM from_clause [WHERE where_exp]。

定义 6 根据 Ontology 结构及 SQL 结构，可以定义对应的 DRS 元素及相关特性：

table(Ref，Name)

querycolumn(Ref，Name)

wherecolumn(Ref，Name，Value，Type)

whererelation(Ref1，and/or，Ref2)

其中，table 表示查询表，querycolumn 表示查询字段，wherecolumn 表示查询条件，whererelation 表示查询条件连接。

3.4.3 语义解释算法

要成功地执行查询并得到结构，用户输入的自然语义查询语句必须包含 3 类信息：查询对应的表、查询属性字段、查询条件属性字段及值。在某些案例中，用户输入的查询语句中省略了某些信息，完善用户的查询信息也是语义解析的一个难点。比如查询语句“Who wrote ‘Gone with the Wine’?”省略了查询的表信息 book。本文语义解析算法采用 Prolog 实现，下面采用 Prolog 子句表示语义解析的算法。语义解释算法可以通过以下步骤实现：

Step 1 系统初始化，加载初始化文件 nlidb. pl，此文件引用 domain 词库文件和 Ontology 知识库文件 consult(nlidb. pl)。

Step 2 输入语法分析生成的 token、连词子句及查询类型 assert(token(Name，POS，ModifyToken))，assert(conjunction (Token1， Token2， Conjunction))， assert (querytype (Name，Type，Subtype))。

Step 3 从 Ontology 知识库中匹配 table，匹配的规则如下：

1) token 的 Name 与 owl:Class 的 Name 匹配：

```
table(X):-token(X,_,_),
    ontClass(X,DP,NSyn,VSyn,DefaultDP)。
```

2)token 的 Name 与 owl:Class 对应的 Synset 词匹配：

```
table(X):-token(X,_,_),
    (ontClass(X,DP,NSyn,VSyn,DefaultDP);
word(NSyn);word(VSyn))。
```

Step 4 从 Ontology 知识库中匹配 querycolumn，规则如下：

1) YesNoQuery 查询期望结果或是查询记录是否存在，querycolumn(‘*’):-querytype(_，yesno，_)。

2) WhQuery 匹配规则根据具体的问句相关，Who 查询与 owl:Class 对应的 owl:DatatypeProperty 的扩展属性 p 匹配，表示人称；Where 查询与 owl:Class 对应的 owl:DatatypeProperty 的扩展属性 l 匹配，表示位置；when 查询与 owl:Class 对应的 owl:DatatypeProperty 的扩展属性 t 匹配，表示时间；what 查询与 owl:Class 对应的 owl:DatatypeProperty 的扩展属性 a 匹配，则表示事物；why 查询与 owl:Class 对应的 owl:DatatypeProperty 的扩展属性 r 匹配，表示原因。如果每个对应的查询存在一个 owl:DatatypeProperty 匹配，则表示找到查询字段；如果有多个 owl:DatatypeProperty 存在，则需要根据其他信息区分哪个是查询字段。

```
querycolumn(X):-token(X,_,_),
    querytype(X,wh,who),
    ontDatatypeProperty(X,C,NSyn,VSyn,p,R)。
querycolumn(X):-token(X,_,_),
    querytype(X,wh,why),
    ontDatatypeProperty(X,C,NSyn,VSyn,r,R)。
querycolumn(X):-token(X,_,_),
    querytype(X,wh,where),
    ontDatatypeProperty(X,C,NSyn,VSyn,l,R)。
querycolumn(X):-token(X,_,_),
    querytype(X,wh,when),
    ontDatatypeProperty(X,C,NSyn,VSyn,t,R)。
querycolumn(X):-token(X,_,_),
    querytype(X,wh,what),
    ontDatatypeProperty(X,C,NSyn,VSyn,a,R)。
```

3) CommandQuery 主要是动词短语组成，动词后面对应的名词如果是 owl:Class，则 owl:Class 对应的默认字段就是 querycolumn，否则动词后面的名词就是 querycolumn：

```
querycolumn(Y):-token(X,_,Y),
    querytype(Y,command,_),
    ontDatatypeProperty(Y,C,NSyn,VSyn,Ext,R)。
```

4) 若上面的匹配都失败，则不带条件修饰的 token 是 querycolumn；

```
querycolumn(X):-token(X,_,_),
```

ontDatatypeProperty(X,C,NSyn,VSyn,Ext,R)。

Step 5 从 Ontology 知识库中匹配剩余 token 的 wherecolumn:

1) 带修饰的 token,则此 token 为 where 条件字段,其修饰为此字段的值:

wherecolumn(X,Y,R):-token(X,_,Y),
ontDatatypeProperty(X,C,NSyn,VSyn,Ext,R)。

2) 修饰 table 的 token,其 where 条件字段是 table 默认字段:

wherecolumn(DefDP,Y,R):-token(X,_,Y),
table(X),
ontClass(X,DP,NSyn,VSyn,DefDP),
ontDatatypeProperty(DefDP,C,NSyn,VSyn,Ext,R)。

3) 表示值的 token,通过值找到对应的字段:

wherecolumn(X,Y,R):-token(_,_,Y),
ontIndividuals(Y,X),
ontDatatypeProperty(X,C,NSyn,VSyn,Ext,R)。

Step 6 根据连词 and/or 匹配 whererelation:

1) token 是 and,则连接修饰的两个 wherecolumn:

whererelation (X,Y,and):-wherecolumn(X,_,_),
wherecolumn (Y,_,_),conjunction(X,Y,and)。

2) token 是 or,则连接修饰的两个 wherecolumn:

whererelation (X,Y,or):-wherecolumn(X,_,_),
wherecolumn(Y,_,_),conjunction(X,Y,or)。

Step 7 把上面解释生成的谓词转换成 DRS:

drs(Table,Querycolumn,[Wherecol],[Whererelat])。

4 数据库处理组件

DRS 表达式已经具备完整的语义信息,包含了 SQL 语句所需的查询条件。现在仅仅需要做的是把 DRS 按照 SQL 语法规则进行映射,生成 SQL 语句,然后传递给 DBMS 进行查询处理,并得到结果。

从 DRS 到 SQL 的转换可以通过转换模板实现。在转换模板中按照 SQL 语法定义转换规则,DRS 表达式数据及语义信息直接映射到模板文件中的变量,从而产生 SQL 语句。整个转换过程的关键是模板的选定和在模板中转换规则的定义。现在有很多模板技术可以选用,比如 Xslt, Velocity, Freemarker 等。本文采用 Velocity 实现 DRS 到 SQL 的映射,Velocity 的模板定义如下:

```
##映射 DRS 到 SQL
SELECT
##查询字段
#if( ${drs.querycolumns.size()} == 0) *
#else #$i=0;
#foreach( $querycolumn in ${drs.querycolumns}
$qeurycolum.name    $i=i+1;
#if( $i ! = ${drs.querycolumns.size()}),
#end #end #end
##查询表
FROM #{drs.table.name}
##查询条件及条件之间的关联
WHERE
#if( ${drs.whererelations.size()} > 0)
#foreach( $wr in ${drs.whererelations}
$drs.wherecolumns.remove(wr.wherecolumn1);
$drs.wherecolumns.remove(wr.wherecolumn2);
$ wr.wherecolumn1.name =
#if ( $wr.wherecolumn1.type == "Number")
    $ wr.wherecolumn1.value
#else '$ wr.wherecolumn1.value' #end
$wr.key $wr.wherecolumn2.name =
#if ( $wr.wherecolumn2.type == "Number")
    $wr.wherecolumn2.value
#else '$wr.wherecolumn2.value' #end #end
#else
#foreach( $wc in ${drs.wherecolumns}
$wc.name =
#if ( $wc.type == "Number")
    $wr.wherecolumn1.value
#else '$wc.value' #end #end
#end
```

DBMS 是非常成熟并且被广泛应用的项目。此处我们可以采用一些标准技术,比如 ODBC,JDBC 等提供一个统一的关系数据库访问技术。这样,本文提出的方法就不依赖于具体的某种数据库实现,从而进一步可以提高系统的可移植性。

在本文中只是返回数据库查询的结果记录,并格式化成一个表格的形式输出。如果要做成对话形式,则必须由自然语言生成器产生自然语言句子输出,这是本文后续要做的工作,不在本文中陈述。

结束语 COT 方法首先需要创建档案模板。本文提出的基于 Ontology 的 NLIDBs 系统模型和设计框架,采用 WordNet 为基础词库,在其之上定义领域词库,采用 OWL Ontology 定义关系数据库 E-R 模型及领域业务模型,提高了查询语句的命中率和正确率,同时也是一种具有高可靠性和高可移植性的关系数据库自然语言访问接口方案。

我们采用 SQLServer 自带的 Pubs 数据库,收集了 72 个查询自然语言查询句子进行试验,结果如图 2 所示。

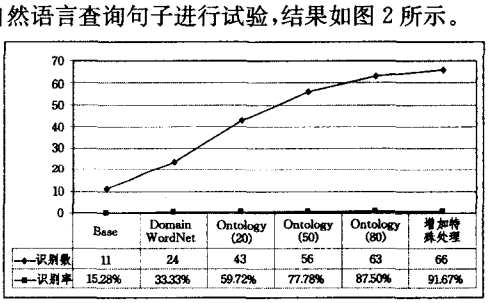

图 2 SQLServre Pubs 数据库测试数据分析图

试验证明此模型基本满足单表查询。在 Ontology 知识库仅有数据库概念模型的情况下,单表有效查询率仅为 15%;结合基于 WordNet 的领域词库以后,单表查询有效率提高至 33%;随着逐步完善 Ontology 知识库,单表有效查询率可以提升到 88%。有些复杂查询子句目前还不在本文提供的方法中,比如:"Who is the author wrote the most number of books?"需要结合一些数据库的表达式功能,属于本文后续研究工作的范围,但是在本文中可以对此查询语句增加一些特殊处理,达到能够识别的目的,这样单表有效查询率可以提升到 92%。

显然,本文提出的模型只是基于 Ontology 知识库的研究

工作的初步结果。为了进一步提高这种模型和方法的可用性，还有一些难点需要解决，比如解决多表关联查询问题、多表嵌套查询问题、根据上下文查询问题和带有表达式语句的复杂查询等问题。

参 考 文 献

[1] Androutsopoulos I, Ritchie G, Thanisch P. Natural language interfaces to databases-an introduction [J]. Journal of Language Engineering, 1995, 1(1): 29-81

[2] Johnson T. Natural Language Computing: The Commercial Applications[M]. London: Ovum Ltd., 1985

[3] Androutsopoulos I. Interfacing a Natural Language Front-End to a Relational Database [D]. Department of Artificial Intelligence, University of Edinburgh, 1993

[4] Woods W A, Kaplan R M, Webber B N. The Lunar Sciences Natural Language Information System: Final Report[R]. BBN Report 2378. Cambridge, Massachusetts: Bolt Beranek and Newman Inc., 1972

[5] Waltz D L. An English Language Question Answering System for a Large Relational Database [J]. Communications of the ACM, 1978, 21(7): 526-539

[6] OWL Web Ontology Language Reference [EB/OL]. http://www.w3.org/TR/owl-ref

[7] Fellbaum C. WordNet: An Electronic Lexical Database [M]. Cambridge, MA: MIT Press, 1998

[8] van Eijck J, Kamp H. Representing Discourse in Context [D]. Elsevier, Amsterdam: Handbook of Logic and Language, 1997: 179-237

[9] Grinberg D, Lafferty J, Sleator D. A robust parsing algorithm for link grammars [R]. CMU-CS-95-125. Proc. Fourth Int. Workshop on Parsing Technologies. Prague, September 1995

[10] The Stanford Parser: A statistical parser[EB/OL]. http://nlp.stanford.edu/software/lex-parser.shtml

[11] Popescu A, Etzioni O, Kautz H. Towards a theory of natural language interfaces to databases [C]// Proceedings of the 8th International Conference on Intelligent User Interfaces. Miami, Florida, USA: ACM Press, January 2003: 327-327

[12] Copestake A J. Natural language interfaces to databases [J]. Knowledge Engineering Review, 1990, 5(4): 225-249

[13] Janus J M. The Semantics-based Natural Language Interface to Relational Databases [M]. Cooperative Interfaces to Information Systems, New York: Springer-Verlag, 1986: 143-187

[14] Kaplan S J. Designing a Portable Natural Language Database Query System [J]. ACM Trans. on Database Systems, 1984, 9(1): 1-19

[15] Trinkunas J, Vasilecas O. Building Ontologies from Relational Databases Using Reverse Engineering Methods [C]// ACM International Conference Proceeding Series. 2007: 285

(上接第 190 页)

展策略，在前、后两个方向上对序列模式进行组合，同时进行剪枝优化和闭合检验。实验结果表明，CloCSP 能够有效挖掘闭合组合序列模式。

参 考 文 献

[1] Marsan L, Sagot M-F. Algorithms for extracting structured Motifs using a suffix tree with an application to promoter and regulatory site consensus identification[J]. J. Computational Biology, 2000, 7(3/4): 345-362

[2] Wang Jianyong, Han Jiawei, Pei Jian. CLOSET+: searching for the best strategies for mining frequent closed itemsets [C]// Proceedings of KDD'2003. 2003: 236-245

[3] Zaki M J, Hsiao C-J. Efficient Algorithms for Mining Closed Itemsets and Their Lattice Structure [J]. IEEE Trans. Knowl. Data Eng., 2005: 462-478

[4] Yan X, Han J, Afshar R. CloSpan: Mining Closed Sequential Patterns in Large Databases [C]//Proc. SIAM Int'l Conf. Data Mining (SDM'03). May 2003: 166-177

[5] Wang Jianyong, Han Jiawei, Li Chun. Frequent Closed Sequence Mining without Candidate Maintenance [J]. IEEE Transaction on Knowledge and Data Engineering, 2007, 19(3): 1042-1056

[6] Carvalho A M, Freitas A T, Oliveira A L, et al. An Efficient Algorithm for the Identification of Structured Motifs in DNA Promoter Sequences [J]. IEEE/ACM Trans. Comput. Biology Bioinform, 2006, 3(2): 126-140

[7] Fassetti F, Greco G, Terracina G. Mining Loosely Structured Motifs from Biological Data [J]. IEEE Trans. Knowl. Data Eng., 2008, 20(11): 1472-1489

[8] Wijaya E, Rajaraman K, Yiu Siu-ming, et al. Detection of generic spaced motifs using submotif pattern mining [J]. Bioinformatics, 2007, 23(12): 1476-1485

[9] Werner T. Models for Prediction and Recognition of Eukaryotic Promoters[J]. Mammalian Genome, 1999, 10(2): 168-175

[10] Tu Z, Li S, Mao C. The Changing Tails of a Novel Short Interspersed Element in Aedes Aegypti: Genomic Evidence for Slippage Retrotransposition and the Relationship between 3' Tandem Repeats and the Poly(da) Tail[J]. Genetics, 2004, 168(4): 2037-2047

[11] Pavesi G, Mauri G, Pesole G. In Silico Representation and Discovery of Transcription Factor Binding Sites[J]. Briefings in Bioinformatics, 2004, 5: 217-236

[12] Bieganski P, Riedl J, Carlis J V, et al. Generalized Suffix Trees for Biological Sequence Data: Applications and Implementation [C]//Proceedings of HICSS (5)'1994. 1994: 35-44