Available online at www.sciencedirect.com

SCIENCE DIRECT®

Information and Software Technology 48 (2006) 901–914

INFORMATION AND SOFTWARE TECHNOLOGY

www.elsevier.com/locate/infsof

Extending the UML concepts to transform natural language queries with fuzzy semantics into SQL[☆]

Frank S.C. Tseng [a,*], Chun-Ling Chen [b]

[a] Department of Information Management, National Kaohsiung First University of Science and Technology, 1 University Road, YenChao, Kaohsiung 824, Taiwan, ROC

[b] Department of Computer Science and Information Engineering, National Chiao Tung University, HsinChu 300, Taiwan, ROC

Received 16 May 2004; received in revised form 20 November 2005; accepted 6 December 2005
Available online 30 January 2006

Abstract

Database applications tend toward getting more versatile and broader to comply with the expansion of various organizations. However, naïve users usually suffer from accessing data arbitrarily by using formal query languages. Therefore, we believe that accessing databases using natural language constructs will become a popular interface in the future. The concept of object-oriented modeling makes the real world to be well represented or expressed in some kinds of logical form. Since the class diagram in UML is used to model the static relationships of databases, in this paper, we intend to study how to extend the UML class diagram representations to capture natural language queries with fuzzy semantics. By referring to the conceptual schema throughout the class diagram representation, we propose a methodology to map natural language constructs into the corresponding class diagram and employ Structured Object Model (SOM) methodology to transform the natural language queries into SQL statements for query executions. Moreover, our approach can handle queries containing vague terms specified in fuzzy modifiers, like 'good' or 'bad'. By our approach, users obtain not only the query answers but also the corresponding degree of vagueness, which can be regarded as the same way we are thinking.

Keywords: Natural Language Query; UML; Class Diagram; Object-Oriented Modeling; Fuzzy Set Theory

1. Introduction

In today's highly challenging environments, knowledge is becoming an important organizational asset that enables sustainable competitive advantage. The concept of Knowledge Management (KM) [1] is that organizational users may make use of knowledge to be more effective and productive in their work. Indeed, knowledge is of limited organizational value if it is not properly shared [2]. That makes many organizations are developing information systems specifically to facilitate the sharing and integration of knowledge.

To share information and knowledge, it is inevitable to pose queries on database systems, as they are ubiquitous and popular in storing enterprise data for various applications. Although the rapid evolution of Internet enables people to share information everywhere at anytime by retrieving data from database systems, however, naïve users still suffer from issuing formal query statements arbitrarily, which may frustrate users and limit the information sharing process.

Since most of the human knowledge is recorded in linguistic form, systems that could understand natural languages could help to access various kinds of information. Therefore, one of the most natural ways to issue queries on databases is using natural languages. In particular, as we have shown in [23], many retrieval tasks posed by complex formal queries can be expressed by using only simple natural language statements.

E-R diagrams was introduced by Chen [7] in 1976 and had its widely usage to model the entities within a relational database schema. For mapping between natural language and database schema, Chen [6] has addressed 11 rules for translations between natural language constructs and E-R diagrams, and our prior research [23] proposed a methodology to map natural language constructs into relational algebra through E-R representations.

Isoda [11] has addressed that Object-Oriented (OO) analysis approach allows us to intuitively and naturally model the real

[☆] This research was partially supported by National Science Council, Taiwan, ROC, under Contract No. NSC 94-2416-H-327-009.

[*] Corresponding author. Tel.: +886 7 6011000; fax: +886 7 7659541.
E-mail addresses: imfrank@ccms.nkfust.edu.tw (F.S.C. Tseng), gis93804@cis.nctu.edu.tw (C.-L. Chen).

0950-5849/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2005.12.002

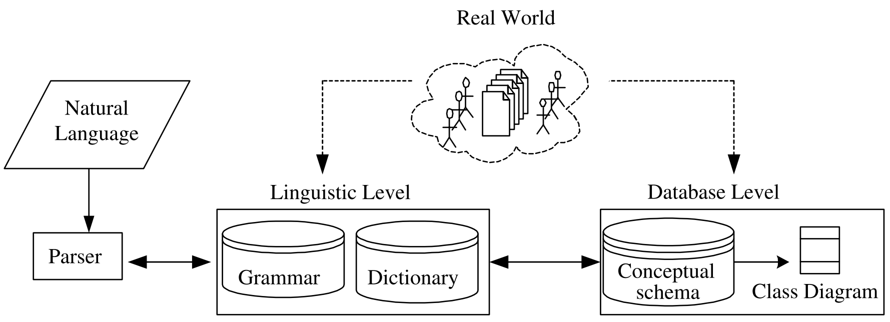


Fig. 1. The inter-relationship between the linguistic level and the database level.

world. The application of OO analysis can help us to identify how a class corresponds to an entity in the real world. Moreover, Moreno and van de Riet [15] have shown that conceptual modeling formalization of the basic constructs of English sentences can be mapped into OO conceptual model in a natural way.

In recent years, UML is becoming a widely accepted methodology for modeling and designing information systems. Due to its standardization, it is applied widely in the industrial and the academic societies. UML is a graphical notation for Object-Oriented system modeling to express requirement analysis and software design. The UML class diagram is an integrated part of UML and is used to design static data models. For the database schema that is designed by the UML class diagram, we have further extended the UML class diagram concepts to capture natural language semantics for database access in [24].

1.1. Objectives

In this paper, we present an approach to map the natural language queries with fuzzy semantics into SQL statements through the UML class diagram representations. Fuzzy queries allow users to express vague predicates, such as 'young' and 'good', to describe object of the real world more naturally. Additionally, fuzzy queries can possibly select a larger number of tuples in comparison to crisp ones. For example, even when crisp queries produce an empty result, the corresponding fuzzy queries can provide more possible answers for users. That makes queries with fuzzy terms more flexible than crisp queries. Based on [29–31], the theory of fuzzy set provides an application of the 'linguistic approach' for the modeling of natural language expressions.

Metais et al. [13] have pointed out that natural languages and the database conceptual schema can represent the conceptualization aspects of the real world. The inter-relationship between the linguistic level and the database level is shown in Fig. 1. In the linguistic level, natural language queries are analyzed by using the predefined grammar and an attached dictionary to reduce ambiguity and complexity. On the other side, in the database level, a schema can be regarded as the blueprint of the conceptual design of a database. Based on such observation, Owei et al. [18] proposed a concept-based query language that allows for the conceptual abstraction of database queries and exploits the rich semantics of semantic data models to ease and facilitate query formulation. In our work, for conceptual schemas organized by class diagrams, we will propose an approach for constructing a natural language interface.

The purpose of this paper is to explore the relationships between natural language constructs and the OO world. By referring to the database schema represented by the UML class diagram, our approach maps natural language constructs with fuzzy semantics into an extended class diagram representation, which will be further transformed into SQL statements for query executions according to the Structured Object Model (SOM) proposed in [9]. The approach handles natural language queries with fuzzy terms in combination with linguistic terms. That makes users issue both crisp and fuzzy query statements more conveniently.

1.2. Limitations

Many researchers believe that natural language interface (NLI) is an ideal means for user-system communication and have made a long effort to develop NLIs to database systems. However, as pointed out in [26], a frequent criticism concern of NLIs is that we cannot expect natural language interfaces to act appropriately for every input sentence. Therefore, we hope to limit our work in processing the following types of query sentences:

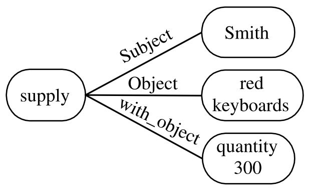
- Interrogative—e.g. "Does Smith supply monitors?"
- Imperative—e.g. "List all the suppliers."
- Declarative—e.g. "Smith supplies monitors." (Will be treated as a question.)

Moreover, users should be aware that the system might be unable to provide an answer if their expectations exceed the actual database capabilities, since the information stored in a database is just a precise world subset. If the system cannot make a decision to get answers due to some ambiguity, then users are asked to answer some questions to clarify the ambiguity.

2. Related work

2.1. Overview of natural language processing

Our prior research [23] has indicated that language-understanding process is commonly divided into three stages.



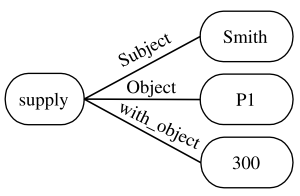


Fig. 2. (a) The semantic roles for "Smith supplies red keyboards with quantity 300." (b) Mapping semantic roles into specific objects.

First, the sentence is parsed according to the predefined grammar, then the semantic roles are built, and finally these semantic roles are mapped into the specific objects in the real world, which in turn may be represented as objects in databases.

The kernel skeleton of a natural language query is composed of a predicate and its accompanying arguments, where verbs and nouns are important features for capturing the main semantic roles within the sentence. In order to get started with the analysis of simple query sentences, we adopt a few grammar rules proposed in [9] for our analytic process.

For building the semantic roles, Winston [27] described a variety of constraints to help establish semantic roles in sentences. These semantic roles reveal how the nouns are related to the verb. Consider the sentence "$(\text{Smith})_{NP}$ $((\text{supplies})_{V4}$ $(\text{red keyboards})_{NP}$ $(\text{with } (\text{quantity } 300)_{NP})_{PP})_{VP}$." It is parsed into a noun phrase and a verb phrase. The verb phrase consists of a noun phrase and a prepositional phrase that consists of another noun phrase.

The semantic roles of the sentence are shown in Fig. 2(a). It indicates that the *verb* is 'supply', the *subject* is 'Smith', the *object* is 'red keyboards', and the *with_object* is 'quantity 300'. Finally, these semantic roles can be mapped into specific objects in the real world as Fig. 2(b) illustrates.

Our approach follows these three stages and focuses on the mapping from semantic roles to a class diagram schema. Thereafter, we apply SOM methodology to transform the corresponding class diagram into SQL statements for query execution.

2.2. Review of UML and Class Diagram

The Unified Modeling Language (UML) [3,17] is a notation that combines elements from the three major kinds of OO design: Rumbaugh's OMT modeling [19], Booch's OO Analysis and Design [4], and Jacobson's Objectory [12]. It provides the conceptual foundation for assembling a system out of components. The 4+1 views and nine diagrams can be selectively employed by analysts/designers within their own methodology to describe the architecture of a software-intensive system. Each view is a projection into the organization and structure of the system, focused on a particular aspect of that system.

Class diagram [3,16,17], one of the nine diagrams in UML, is used to show a set of classes and their relationships. Our approach mainly employs class diagram to provide a static view of application concepts, in terms of classes and their relationships including generalization and association. For example, the class diagram of the Suppliers-Parts-Projects database as shown in Fig. 3 can be shown in Fig. 4. The terms with *italic style* in Fig. 4 indicates the concepts about class diagrams. Our examples will be based on this database hereafter.

2.3. Traditional Database Model vs. Fuzzy Database Model

Traditional relational database systems have been proved on their usefulness in various domains. To extend the ability of dealing with vague and imprecise data, researchers expedite the development of fuzzy databases. In [25, 28, 33], researchers have pointed out that a fuzzy database can be defined as an enhanced relational database that stores *fuzzy attribute values* and *fuzzy truth values* [22] as shown in Table 1. We briefly explain these terms as follows.

- *Fuzzy attribute values*. Attribute values such as *status* in the Suppliers-Parts-Projects database have non-fuzzy values such as 20 in the relational database. However, to support fuzzy queries, such attribute values in fuzzy databases can be defined as fuzzy attribute values, such as 'good' or 'bad'.

Suppliers

sno	*sname*	*status*	*city*
S1	Smith	40	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adames	10	Taipei

Parts

pno	*pname*	*color*	*weight*
P1	keyboard	red	10
P2	case	blue	10
P3	monitor	white	25
P4	mouse	red	5
P5	mouse	blue	5
P6	keyboard	white	15

Projects

jno	*jname*
J1	AS100
J2	ML130
J3	SU150

Shipments

sno	*pno*	*jno*	*qty*
S1	P1	J1	300
S1	P2	J1	200
S1	P3	J1	400
S1	P4	J1	200
S2	P1	J2	100
S2	P2	J3	100
S2	P5	J3	300
S3	P1	J1	400
S4	P1	J1	200
S4	P2	J2	200
S5	P2	J2	300
S5	P4	J3	400

Fig. 3. The Suppliers-Parts-Projects database.

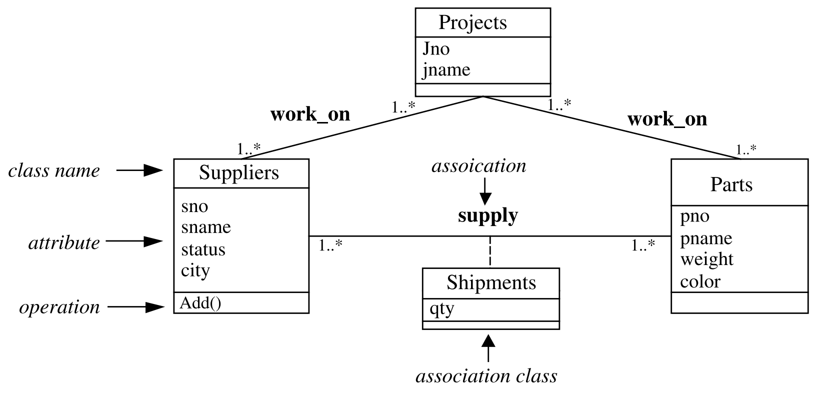


Fig. 4. A class diagram for the Suppliers-Parts-Projects database.

- *Fuzzy truth values*. Truth values of tuples are either 1 (true) or 0 (false) in traditional relational databases. However, they can be generalized into fuzzy truth values with associated possibilities defined in the fuzzy databases.

2.4. Fuzzy Sets, Membership Functions, Linguistic Variables and Fuzzy Terms

Fuzzy set theory [29] provides an auspicious representation for imprecise and vague data. It has been widely used in various fields, like decision-making, economics, and psychology. To discuss the concept of fuzzy database, we briefly review two basic definitions as follows.

Definition 2.1. A *fuzzy set A*, is defined in terms of a relevant universal set U by a membership function, denoted as follows

$$A = \{(u_i, \mu_A(u_i))|u_i \in U\},$$

where $\mu_A(u_i)$, $\mu_A(u_i)$: $U \to [0,1]$, indicates the grade of membership of u_i in A.

When U is a finite set and $U=\{u_1,u_2,\ldots,u_n\}$, then the fuzzy set A can be represented by

$$A = \sum_{i=1}^{n} \mu_A(u)/u_i = \mu_A(u_1)/u_1 + \mu_A(u_2)/u_2 + \cdots + \mu_A(u_n)/u_n,$$

where the symbol '+' means the 'union' operator, and the symbol '/' means a separator.

Definition 2.2. If A is an S fuzzy set of the universe of discourse U, then the membership function of $\mu_A(u)$ can be defined by:

$$\mu_A(u) = \begin{cases} 0, & u < a, \\ \left(\dfrac{u-a}{b-a}\right), & a \le u \le b, \\ 1, & u > b. \end{cases}$$

Fig. 5 shows the membership function curve of the S fuzzy set A.

Example 2.1. Consider the attribute *status* in the Suppliers-Parts-Projects database with an S fuzzy set of the universe of discourse [0,50]. Then the membership function of $\mu_{status}(u)$ can be defined as follows:

$$\mu_{status}(u) = \begin{cases} 0, & u < 0, \\ \dfrac{u-0}{50}, & 0 \le u \le 50, \\ 1, & u > 50. \end{cases}$$

Fig. 6 shows the membership function curve of the fuzzy set *status*.

Zadeh [32] defined a *linguistic variable* as a variable whose values are not numbers but words or sentences in natural language. The motivation for the use of words or sentences rather than numbers is that linguistic characterizations are, in general, less specific than numerical ones. The use of linguistic variables [20] allows a precise modeling of imprecise statements like 'old' and 'young'. Linguistic variables allow us to specify easy and natural specification of values for

Table 1
Fuzzy attribute values and fuzzy truth values for *status*

sno	*Status*	Fuzzy attribute values	Fuzzy truth values
S1	40	Good	0.8
S2	10	Bad	0.2
S3	30	Good, Middle	0.6
S4	20	Middle, Bad	0.4
S5	10	Bad	0.2

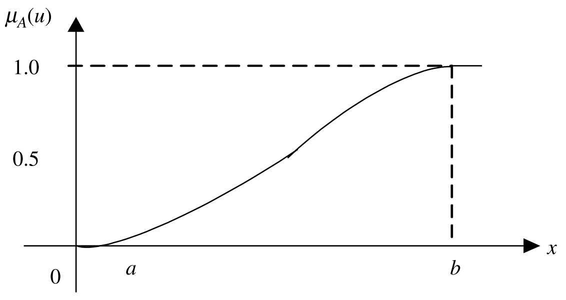


Fig. 5. The membership function curve of the S fuzzy set A.

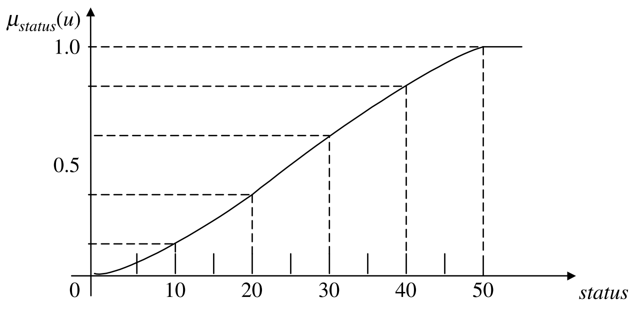


Fig. 6. The membership function curve of the fuzzy set *status*.

imprecise concepts. For instance, if the values of ‘age’ are linguistic (e.g. old, young, etc.) rather than numerical (e.g. 20,21,…), then ‘age’ is regarded as a *linguistic variable*. In this case, ‘old’ and ‘young’ are simply called *fuzzy terms*. For example, the attribute values of attribute *status* can be represented by the membership function $\mu_{status}(u)$ as shown in Table 2.

To present an example of $\mu_{status}(u)$, we define its corresponding fuzzy terms ‘good,’ ‘middle’ and ‘bad’ in Table 3.

2.5. Structured Object Model (SOM)

In [10], Higa and Owei proposed a conceptual data model driven programming tool for database query processes. They have developed the Structured Object Model (SOM) as an analysis, design and navigation tool for database applications. Consider the SOM diagram *S* shown in Fig. 7, where an added feature of double-headed arrows in Fig. 7 indicates direct logical connectivity between object pairs. Attached to these arrows are the attributes through which the objects are connected.

2.5.1. The Logical Adjacency Matrix and the Connectivity Matrix

Based on the SOM representation, the corresponding Logical Adjacency Matrix (LAM) can be derived to indicate the direct connectivity between pairs of objects. Fig. 8 is the logical adjacency matrix, L_S, of Fig. 7, where the adjacency between pairs of objects is not taken in the sense of the physical connections shown in the figure. A zero element in the matrix implies that there is no direct connection, where the two objects have no relationship, or that an object is trivially connected to itself.

Table 2
The mapping between the attribute values of *status* and μ_{status}

Status	0	10	20	30	40	50
μ_{status}	0	0.2	0.4	0.6	0.8	1

Table 3
The mapping between μ_{status} and its corresponding fuzzy terms

μ_{status}	[1,0.6]	[0.6,0.4]	[0.4,0]
Fuzzy term	Good	Middle	Bad

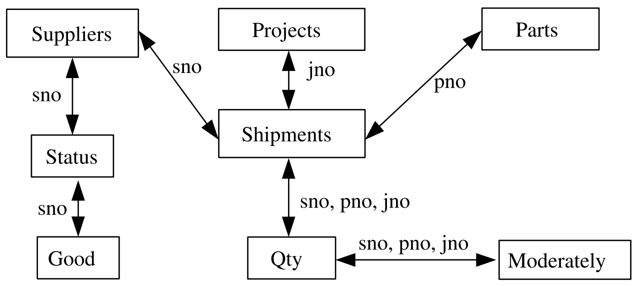


Fig. 7. The Example SOM Diagram *S*.

Based on the LAM, a Connectivity Matrix (CM), C_S, for the SOM diagram *S* can be determined as shown in Fig. 9. Matrix C_S captures the information on whether or not any two objects are connected, either directly or indirectly. An entry of C_S indicates a path length connecting the two corresponding objects.

2.5.2. The mapping of object paths to query statements

When users pose queries, the following steps can be employed to execute a query. First, the system extracts a sub-graph (or sub-tree), called a validation sub-tree, of the SOM diagram. Second, the validation sub-tree consists solely of the relevant object connecting the *source* (the object or set of objects, with their association valued-attributes, which are stated in the query, and to which the object and attributes of interest are related) and *target* (the object of ultimate destination). Then, the system maps the validated sub-tree to the connectivity matrix, to determine if a logical path exists between the source and target. Finally, if a logical path exists, the system internally maps the chosen logical path to an equivalent SQL-type statement, which is then executed.

We illustrate this process by referring the SOM diagram of Fig. 7. Suppose the query has been processed, such that we identify the source object as E, with primary attribute value ‘E1’, and the target object as D. The attribute of interest is the D-descriptor of object D. Then, the query session proceeds as outlined below.

User Specifies *Source*:
Object: E
Attribute: E-id = ‘E1’

User Specifies *Target*:
Object: D
Attribute: D-descriptor

By referring to the matrix C_S as shown in Fig. 9, the logical path is determined as:

E → D-E → D

The system then returns the validation sub-tree as shown in Fig. 10.

Finally, the system generates the SQL statement for query execution as depicted in Fig. 11(a). The generated SQL statement can be further converted into another SQL statement using inner joins (or natural joins) as shown in Fig. 11(b).

	Suppliers	Status	Good	Shipments	Parts	Projects	Qty	Moderately
Suppliers	0	1	0	1	0	0	1	0
Status	1	0	1	0	0	0	0	0
Good	0	1	0	0	0	0	0	0
Shipments	1	0	0	0	1	1	0	0
Parts	0	0	0	1	0	0	1	0
Projects	0	0	0	1	0	0	1	0
Qty	1	0	0	0	1	1	0	1
Moderatley	0	0	0	0	0	0	1	0

Fig. 8. LAM L_S for the SOM diagram S in Fig. 7.

3. The processing model and mapping process

The natural language processing model is shown in Fig. 12. In this model, data can be distinguished into 'crisp data' and 'fuzzy data':

1. *Crisp data*. Crisp data are stored in traditional databases. For example, the attribute values of 'Age' are precise numbers (e.g. 'Age' can be 20, 30 or 40).
2. *Fuzzy data*. Fuzzy data are represented as fuzzy terms that the words or phrases in queries are the name of variables. For example, if 'Age' is a linguistic variable, it may have fuzzy terms like 'young' or 'old'.

3.1. The processing model

Our processing model makes the following assumptions:

1. In addition to the original attributes, each class is augmented with an object identifier (OID), which is underlined in Fig. 3. Any reference to an OID is interpreted as a reference to the corresponding object. For example, in the Suppliers-Parts-Projects database, we have added the extra attributes *sno* and *pno* as the OIDs of **Suppliers** and **Parts**, respectively.
2. Each association will be represented as an association class, where an OID is composed of the OIDs of the involved classes. For example, the OID of the class **Shipments** for the association **supply** can be obtained by grouping *sno* and *pno* into (*sno*, *pno*).
3. All fuzzy classes that modify the same class *C*, which corresponds to a semantic role, have the same OID attribute with *C*. Besides, all FT classes have the same OID attribute with the fuzzy class in which they are attached. That is, an instance of a fuzzy class and the associated FT classes use the same OID to relate to the class instance they modify. For example, the OID of an instance, e.g. 'S2', in the fuzzy attribute *status* of the class **Suppliers** and the OID of that in the fuzzy class **Status** are the same.

In the processing model, there is a front–end parser/mapper to parse the English queries and map them into the underlying class diagram schema. The parsing and mapping process may refer to

1. The dictionary;
2. The underlying class diagram schema associated with the semantic-role frame (CD/SRF) as Fig. 2 illustrates;
3. The underlying class diagram schema associated with pre-defined membership functions (CD/MF). This database is used to allow users to express their queries with fuzzy predicates. It stores various membership functions and the description of the correspondence between fuzzy attribute values and fuzzy truth values.

After the parsing phase, the query is decomposed into:

1. *Semantic roles*. Each semantic role is composed of a headnoun and some modifiers. A headnoun is the main noun in a noun phrase. For example, 'the London suppliers' has a headnoun 'suppliers' and the other noun 'London' is a modifier, which modifies the headnoun.
2. *Verb*. The verb that relates the derived semantic roles.
3. *Fuzzy modifiers* and *fuzzy terms* used to describe objects. Each fuzzy modifier is a linguistic variable with fuzzy terms used as an adjective to describe an object represented by a semantic role. For instance, a query containing 'the suppliers with status good' has a fuzzy modifiers (or a linguistic variable) 'status' describing the headnoun 'suppliers' with fuzzy term 'good'.

	Suppliers	Status	Good	Shipments	Parts	Projects	Qty	Moderately
Suppliers	0	1	2	1	2	2	1	2
Status	1	0	1	2	3	3	2	3
Good	2	1	0	3	4	4	3	4
Shipments	1	2	3	0	1	1	2	3
Parts	2	3	4	1	0	2	1	2
Projects	2	3	4	1	2	0	1	2
Qty	1	2	3	3	1	1	0	1
Moderately	2	3	4	3	2	2	1	0

Fig. 9. Connectivity matrix C_S for the SOM diagram S in Fig. 7.

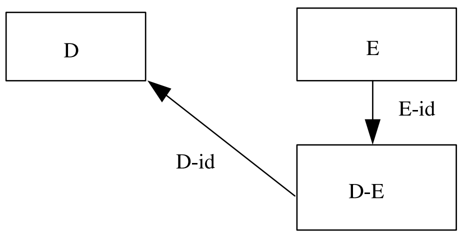


Fig. 10. The validation sub-tree for end-user.

4. *Fuzzy modifiers* and *fuzzy terms* used to describe relationships. In such cases, each fuzzy modifier is a linguistic variable with fuzzy terms used as an adverb to describe the verb itself. For instance, a query containing 'Smith *largely* supplies the monitors for AS100' has a fuzzy modifiers (or a linguistic variable) corresponding to the attribute *qty* describing the supplier 'Smith' with a fuzzy term 'largely'. For example, for attribute *qty*, the fuzzy attribute values and fuzzy truth values, together with their corresponding mappings between μ_{qty} can be assumed as Table 4 shows. Notice that, since a relationship involves semantic roles, we have to retain all the OIDs of the involved objects (i.e. *sno*, *pno*, and *jno*) into Table 4(c) to show the detailed relationship on which the *qty* is defined.

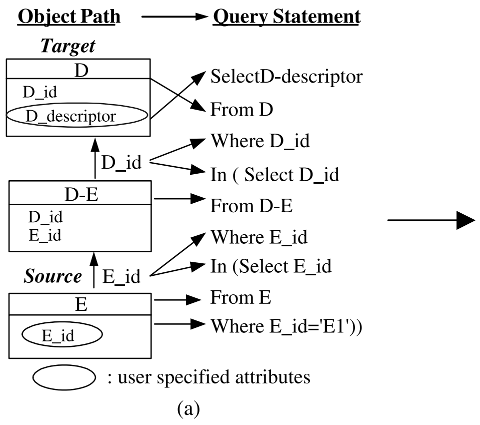
By referring to the dictionary (to be discussed in Section 3.1.1), each of the semantic roles is mapped into a class and its headnoun and modifiers are mapped into the corresponding attributes of those classes based on CD/SRF (discussed in Section 3.1.2). The verb that relates these semantic roles is mapped into the relationship among these semantic roles. Each of fuzzy modifiers is mapped into a fuzzy class and its adjective/adverb will be mapped into the corresponding fuzzy term (FT) classes of those classes based on CD/MF (discussed in Section 3.1.3).

In Table 5, we summarize the correspondence between English sentence structures and class diagram constructs, which simplifies the 11 translation rules addressed in [6]. Figs. 13 and 14 show the mapping mechanisms for association relationships and fuzzy queries, respectively.

3.1.1. The dictionary

The linguistic knowledge that enables the semantic roles of a verb to be mapped into the correct attributes of the corresponding relationship is stored in the dictionary. For the Suppliers-Parts-Projects database, the linguistic knowledge for a verb used in a natural language query can be shown in Table 6. For example, according to Fig. 4, the transitive verb 'supply' (and its synonyms) could be mapped into the association **supply**, which corresponds to the class **Shipments**, and its corresponding semantic roles can be mapped in the schema are a subject, a direct object, an indirect object, a with_object (which is regarded as an adverb phrase that modifies the verb), and a for_object (which relates to the indirect object). The subject, direct object, and indirect object associate the classes **Suppliers**, **Parts**, and **Project**, respectively, and are mapped into the corresponding OIDs.

Besides, there is a fuzzy dictionary used to store the linguistic variables that enable the fuzzy modifier to be mapped



```
Select  D-dscrptor
From (D INNER JOIN D-E  On D.D_id = D-E.D_id)
          INNER JOIN E      On D-E.E_id=E.E_id
where  E.E_id= 'E1'
```

(b)

Fig. 11. (a) Object Path-Query Statement Mapping. (b) SQL statement using INNER JOINS.

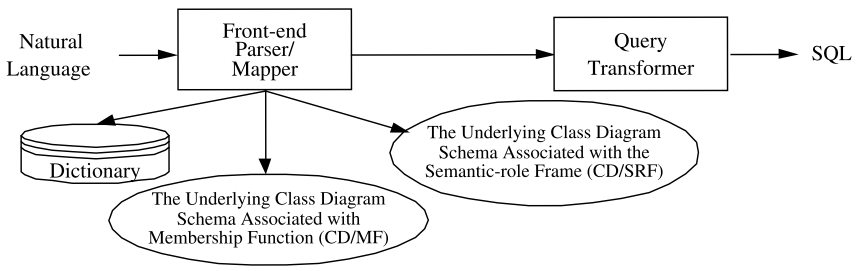

Natural Language
Front-end Parser/ Mapper
Query Transformer
SQL
Dictionary
The Underlying Class Diagram Schema Associated with the Semantic-role Frame (CD/SRF)
The Underlying Class Diagram Schema Associated with Membership Function (CD/MF)

Fig. 12. The processing model.

Table 4
The use of *qty* as an example to illustrate fuzzy modifiers and fuzzy terms for the verb *supply*

(a) The mapping between the attribute values of *qty* and μ_{qty}

qty	0	100	200	300	400	500
μ_{qty}	0	0.2	0.4	0.6	0.8	1

(b) The mapping between μ_{qty} and its corresponding fuzzy terms

μ_{qty}	[1,0.6]	[0.6,0.4]	[0.4,0]
Fuzzy term	Largely	Moderately	Rarely

(c) Fuzzy attribute values and fuzzy truth values for *qty*

sno	*pno*	*jno*	*qty*	Fuzzy attribute values	Fuzzy truth values
S1	P1	J1	300	Largely, moderately	0.6
S1	P2	J1	200	Moderately, rarely	0.4
S1	P3	J1	400	Largely	0.8
S1	P4	J1	200	Moderately, rarely	0.4
S2	P1	J2	100	Rarely	0.2
S2	P2	J3	100	Rarely	0.2
S2	P5	J3	300	Largely, moderately	0.6
S3	P1	J1	400	Largely	0.8
S4	P1	J1	200	Moderately, rarely	0.4
S4	P2	J2	200	Moderately, rarely	0.4
S5	P2	J2	300	Largely, moderately	0.6
S5	P4	J3	400	Largely	0.8

into the fuzzy classes of various membership functions and fuzzy term classes. For example, for the Suppliers-Parts-Projects database, we have shown in Table 7 that there may be two fuzzy modifiers 'status' and 'qty', which may occur in a natural language query for modifying the fuzzy classes 'Status' and 'Qty', respectively. The fuzzy classes 'Status', with the corresponding fuzzy term classes '**Good**', '**Middle**', and '**Bad**', and 'Qty', with the corresponding fuzzy term classes '**Largely**', '**Moderately**', '**Rarely**', are mapped to the adjective and adverb modifiers for suppliers and the verb *supply*, respectively.

3.1.2. The schema and the semantic-role frame represented in class diagram

For each class, attributes are identified as the *headnoun* and the *modifiers* that modify the headnoun. For example, the attributes corresponding to the headnouns of **Suppliers** and **Parts** are *sname* and *pname*, respectively. Such information will be encoded into the class diagram. For example, the schema and the semantic-role frame representing the Suppliers-Parts-Projects database are shown in Fig. 15.

3.1.3. The schema and membership function represented in class diagram

Each fuzzy modifier can be mapped into a membership function to define its corresponding fuzzy class. We assume the name of a fuzzy modifier and its corresponding fuzzy class is the same. For instance, a fuzzy modifier *status* in the class **Suppliers** of the Suppliers-Parts-Projects database can be mapped into the fuzzy class **Status**.

In our example, we regard 'status' as a linguistic variable and define its fuzzy term classes as **Good**, **Middle**, and **Bad**. The class diagram of fuzzy class **Status** and its fuzzy term classes are shown in Fig. 16 (please also refer to Fig. 6 and Table 2), where MFV represents the corresponding membership function values, and Fig. 17, respectively. Analogously, the class diagram of fuzzy class **Qty** and its fuzzy term classes can be derived accordingly.

3.2. Description of the mapping process

The natural language mapping process can be divided into the following stages:

1. *Mapping the only transitive verb into an association relationship*. Since a query sentence has only one transitive verb, by referring to the dictionary, we can map the transitive verb of a query sentence into the corresponding association and its semantic roles into the corresponding attributes. Besides, an adverb (or adverb phrase) that modifies the verb can be mapped into the corresponding attribute of the association class. For instance, the query "Does Smith *supply* keyboards with quantity 300 for ML130?" has the transitive verb *supply*, and by referring to the dictionary shown in Table 6, the semantic roles can be mapped into the corresponding attributes as depicted in Fig. 18. Notice that the query sentence may be posed in negative form instead of affirmative form. For example, "Doesn't Smith *supply* keyboards with quantity 300 for

Table 5
Correspondence between English sentence structures and class diagram constructs

English sentence structures	Class diagram constructs
Common noun	Class name/the attribute of class/fuzzy class name
Proper noun	Object name
Transitive verb	Association name/association class name
Adjective	The attribute of class/object/FT class name
Adverb	The attribute of association class/FT class name

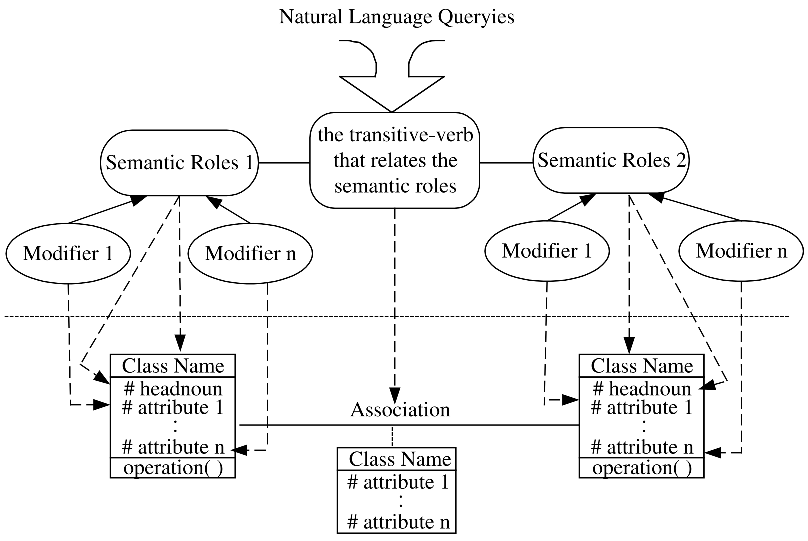


Fig. 13. The mapping mechanism of association relationship.

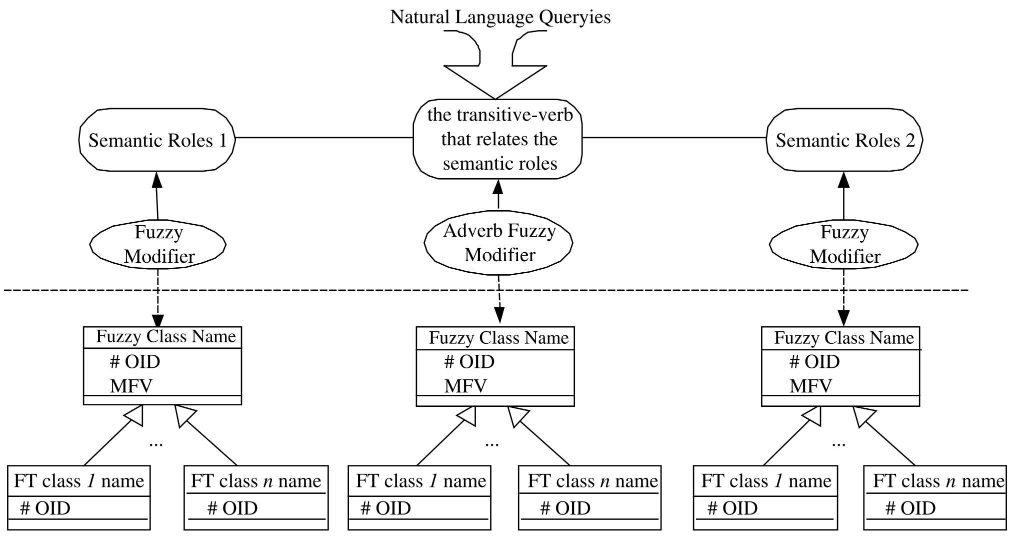


Fig. 14. The mapping mechanism of fuzzy modifiers.

ML130?" This detail about dealing with such situation can be found in [23], gentle readers are referred to it.

2. *Mapping noun phrases into classes and attributes*. As we have shown in the previous section, the semantic roles of a verb in a sentence can be mapped into those OIDs of the involved classes. These classes actually interpret the noun phrases of the sentence. However, the noun phrases may consist of more than one noun modifier (which can be adjectives or nouns) or relative clauses that modify the headnoun. For example, "the London supplier" and "the red parts" are two noun phrases and have the noun modifiers 'London' and 'red', respectively. These noun modifiers can be interpreted into attributes of the corresponding class. According to the dictionary, the interpretation of these noun modifiers described above can be mapped as Fig. 19 illustrate.

Table 6
The dictionary structure of Suppliers-Parts-Projects database

Verbs	Class	Semantic roles	Corresponding attribute(s)
Supply (and its synonyms)	**Shipments**	Subject	*sno* (the OID of **Suppliers**)
		Direct Object	*pno* (the OID of **Parts**)
		Indirect Object	*jno* (the OID of **Projects**)
		With_object	*qty*

Table 7
The fuzzy dictionary structure of Suppliers-Parts-Projects database

Fuzzy attribute	Corresponding fuzzy classes	Corresponding FT classes
status (for *subject* Suppliers)	**Status**	**Good** **Middle** **Bad**
qty (for *verb* supply)	**Qty**	**Largely** **Moderately** **Rarely**

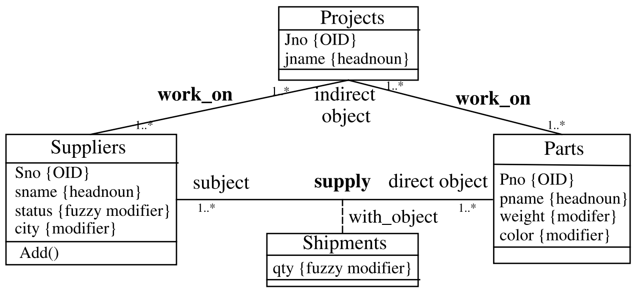


Fig. 15. The schema and the semantic-role frame of the Suppliers-Parts-Projects database.

3. The representation for the word '*All*': Consider the following examples:
   (1) List the suppliers who supply *all* red parts.
   (2) List the suppliers who supply red parts.

The answer of the first example is the suppliers who supply *all* the red parts, but the second one is the suppliers who supply *any* red parts. Therefore, the answer of (1) is always contained in that of (2). We use a shadow block on the class name to represent the class whose semantic role is preceded by 'all' (Fig. 20(a)). Otherwise, we use a blank block instead (Fig. 20(b)). Note that the semantic meaning of "List all the suppliers who supply all red parts." is equivalent to that of (1). The first 'all' has no effect on this query. We have already shown that the relational operator *divide* can be used to implement such queries. To make this paper concise, gentle readers are referred to [23,24].

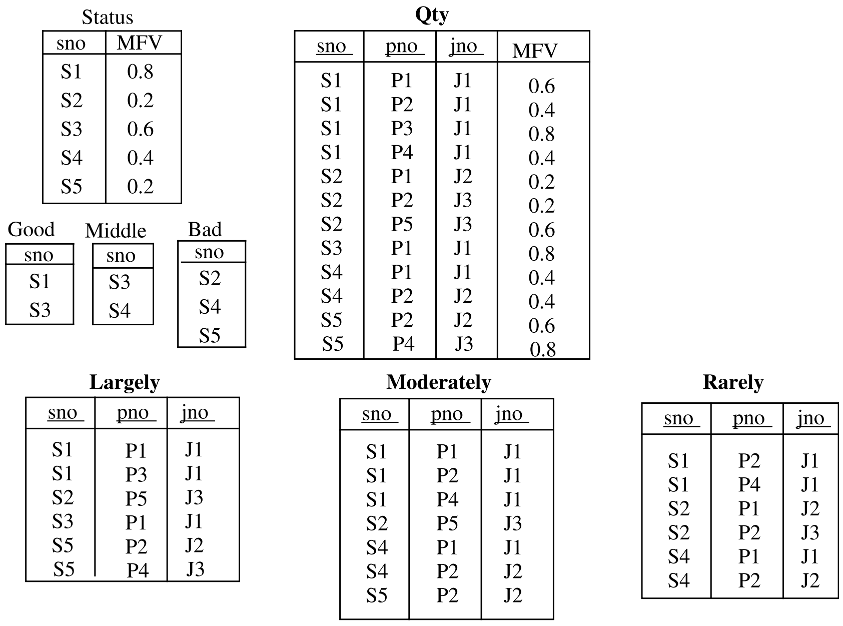
4. Mapping fuzzy noun phrases into fuzzy classes and fuzzy term classes. If a noun phrase consists of one fuzzy modifier (which can be an adjective or a noun), this fuzzy modifier can be mapped into its corresponding fuzzy class and fuzzy term classes. For example, a noun phrase "the suppliers with status good" (or "the good suppliers") has the fuzzy modifier 'status'. This fuzzy modifier can be interpreted into fuzzy class **Status**, and the adjective 'good' that modifies this fuzzy modifier can be mapped into fuzzy term class **Good**. According to the fuzzy dictionary, the interpretation of the modifier described above can be mapped as Fig. 21 shows.

Example 3.1. For the query $Q=$"List all the project names, for which a good supplier moderately supply red parts." The corresponding mapped class diagram of this query is shown in Fig. 22.

Note that, a recursive relationship may be regarded as a binary relationship relates the same entity, such that one copy of the entity corresponds to the subject, and the other to the object. Based on such realization, the mapping process can be applied accordingly.

4. Query transformer

In this section, we employ SOM to transform English queries into SQL statements through the corresponding mapped class diagrams. The database schema of Suppliers-



Status

sno	MFV
S1	0.8
S2	0.2
S3	0.6
S4	0.4
S5	0.2

Qty

sno	pno	jno	MFV
S1	P1	J1	0.6
S1	P2	J1	0.4
S1	P3	J1	0.8
S1	P4	J1	0.4
S2	P1	J2	0.2
S2	P2	J3	0.2
S2	P5	J3	0.6
S3	P1	J1	0.8
S4	P1	J1	0.4
S4	P2	J2	0.4
S5	P2	J2	0.6
S5	P4	J3	0.8

Good

sno
S1
S3

Middle

sno
S3
S4

Bad

sno
S2
S4
S5

Largely

sno	pno	jno
S1	P1	J1
S1	P3	J1
S2	P5	J3
S3	P1	J1
S5	P2	J2
S5	P4	J3

Moderately

sno	pno	jno
S1	P1	J1
S1	P2	J1
S1	P4	J1
S2	P5	J3
S4	P1	J1
S4	P2	J2
S5	P2	J2

Rarely

sno	pno	jno
S1	P2	J1
S1	P4	J1
S2	P1	J2
S2	P2	J3
S4	P1	J1
S4	P2	J2

Fig. 16. The fuzzy class and its fuzzy term classes.

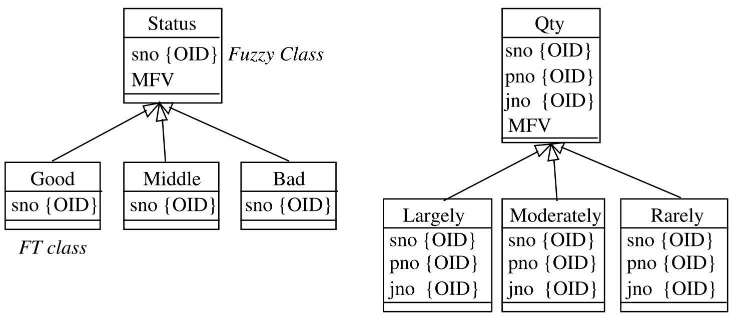


Fig. 17. The class diagram schema of the fuzzy class and its fuzzy term classes.

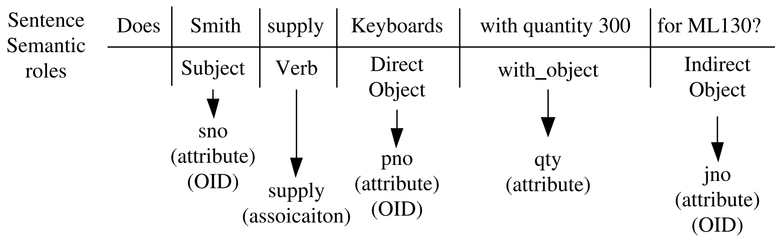


Fig. 18. The dictionary mapping for "Does Smith supply keyboards with quantity 300 for ML130?".

Parts-Projects can be mapped into the SOM schema as shown in Fig. 23. The root object is a dummy object, which represents the entire schema of the database.

We follow Example 3.1 to illustrate how to transform the corresponding class diagram into SQL statements. By referring to Fig. 23, Fig. 22 can be converted to an SOM diagram S as depicted in Fig. 7, which has the corresponding L_S and C_S as described in Figs. 8 and 9, respectively. That is, the following steps explain the query transformation process:

1. According to the class diagram obtained from a natural language query, the query transformer extracts a valid subgraph from the SOM diagram, such that the subgraph consists of the relevant objects connecting the *source* and *target*. For our example, Fig. 7 is a valid subgraph of Fig. 23.
2. The query transformer derives the connectivity matrix C_S from the validated subgraph to determine if a logical path exists between the source and target. If the path does not exist then it proceeds to Step 6. For example, Fig. 9 can represent the derived C_S of the validated subgraph, Fig. 7.
3. If a logical path exists, the query transformer maps the chosen logical path to equivalent SQL statements for execution. For instance, the source object of the query specified in Example 3.1 can be identified as **Shipments**, **Suppliers**, **Status**, **Good**, and **Parts** (with attribute value *color* = 'red'). Besides, the target object can be recognized as **Projects**. The attribute of interest is the attribute *jname* of **Projects**. The query session thus proceeds as below:

   User Specifies Source:

   Object: **Shipments**, **Qty**, **Moderately**, **Suppliers**, **Status**, **Good**, and **Parts**

Noun Phrase	Corresponding Class	Interpretation of the Noun Modifier
the London suppliers	**Suppliers**	City = 'London'
The red parts	**Parts**	Color = 'red'

Fig. 19. Mapping noun modifiers into attributes.

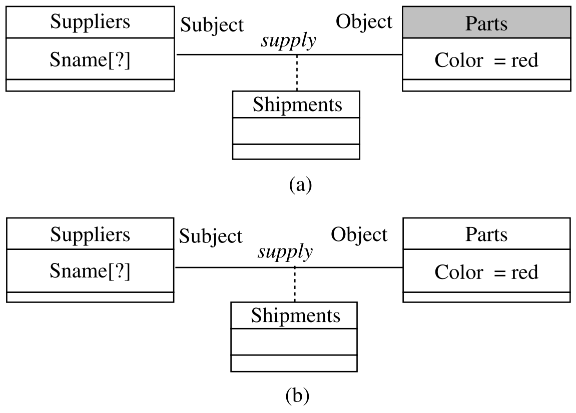


Fig. 20. (a) The logical form for "List the suppliers who supply all red parts." (b) The logical form for "List the suppliers who supply red parts". (For interpretation of the reference to colour in this legend, the reader is referred to the web version of this article.)

Noun Phrase	Corresponding Fuzzy Modifier	Corresponding Fuzzy Classes	Corresponding FT Classes
good status	status	**Status**	**Good**

Fig. 21. Mapping a fuzzy noun modifier into its fuzzy class and fuzzy term classes.

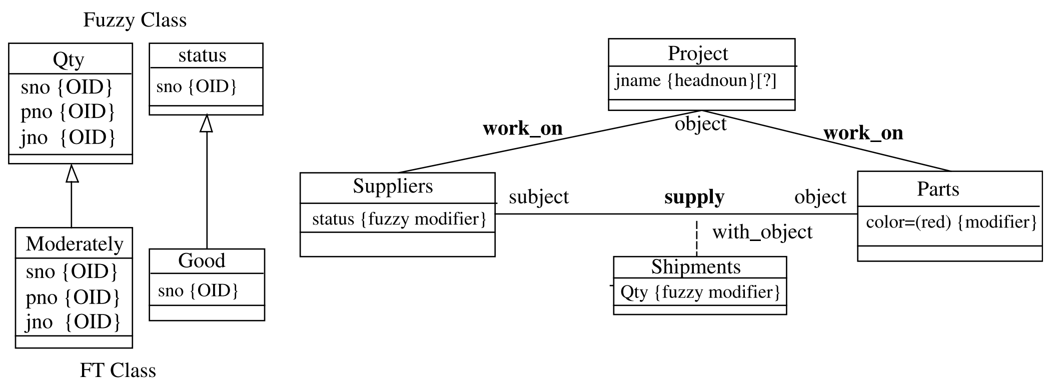


Fig. 22. Mapping “List all the project names, for which a good supplier moderately supply red parts” to class diagram.

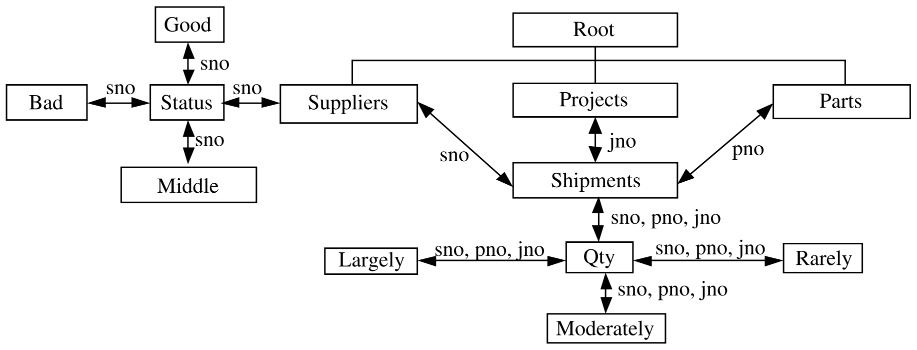


Fig. 23. The SOM diagram of the Suppliers-Parts-Projects database.

Attribute: *color* = ‘red’
User Specifies Target:
Object: **Projects**
Attribute: *jname*

Then, the query transformer returns the valid subgraph as Fig. 24 depicts.

By Fig. 9, the query transformer determines the logical path as:

$$\text{Projects} \leftarrow \text{Shipments} \leftarrow (\text{Qty} \leftarrow \text{Moderately}) \leftarrow (\text{Suppliers} \leftarrow \text{Status} \leftarrow \text{Good}) \leftarrow \text{Parts}$$

4. Based on the result derived in Step 3 and the approach proposed by [10], the query transformer can derive the complex query statement as Fig. 25 illustrates. However, according to the derived logical path, the SQL statement can also be transformed into a more simplified statement by inner joins (or natural joins) as specified in Fig. 26.
5. Then, the system presents the execution result to the end-user. For our example, the system will show ‘AS100’ (refer to Figs. 3 and 16) to the end-user.
6. If the system cannot make a decision to get answers, the question will be resolved by the following two principles. First, when two or more logical paths exist between the source and the target, users are then asked to answer some questions to clarify the ambiguity. Otherwise, if there is no logical path exists between the source and the target, an error message will be raised.

5. Conclusions and future directions

5.1. Conclusions

Natural language interfaces are promising for today’s database applications. This can be realized from some database vendors are eager to empower their products with the ability of issuing English queries for database access. For example, Microsoft SQL Server 2000 [14] is equipped with an English Query module to do this. However, there are still limitations of using the E-R diagram to be the logical form when mapping natural language constructs into database schemas. In this

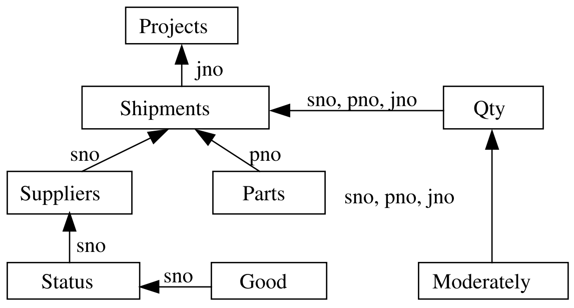


Fig. 24. The validation sub-graph of Example 3.1.

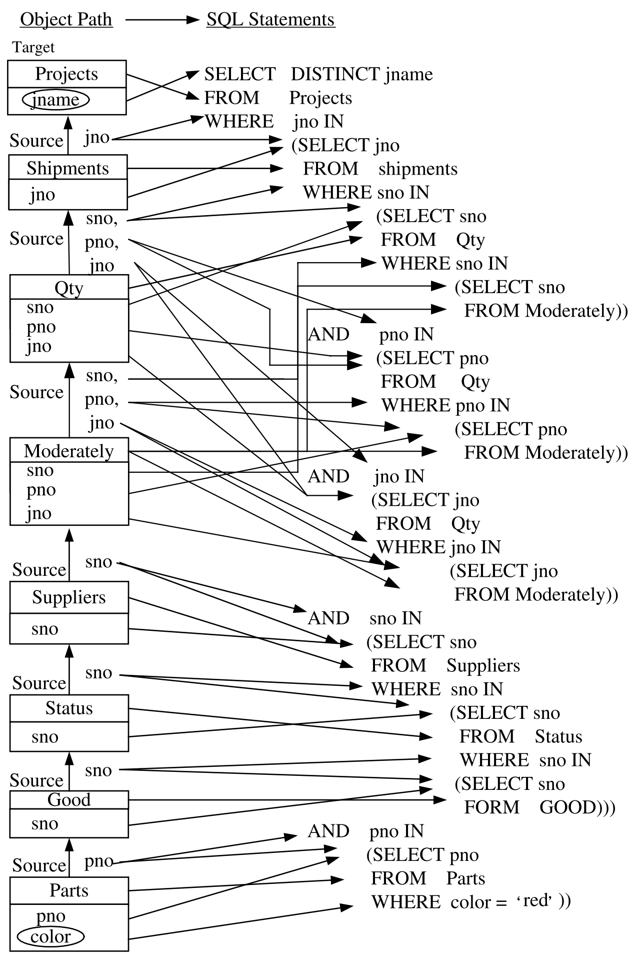


Fig. 25. The mapping process of Example 3.1.

paper, we extend the class diagram representation of UML to capture more natural language semantics for querying databases. We employ SOM methodology and describe a processing model for the query transformation process. In this processing model, when the target database is changed, we only have to change the dictionary, the CD/SRF, and the CD/MF in Fig. 12. The front–end parser/mapper and the query transformer remain unchanged.

Furthermore, we have studied the inter-relationship between natural language constructs and the class diagram conceptual schema. The basic parts of English sentences can be mapped into class diagram schemas in a natural way. Our approach allows users to intermingle fuzzy queries with crisp queries and provide a natural way to retrieve data from databases.

Finally, our approach is also applicable for a database modeled by E-R diagram or Extended E-R diagram, as they can be transformed into class diagram representation. If the underlying schema of a database was pre-existing but not structured by class diagram approach, then the schema can be restructured by class diagram approach through defining the design views by the UML approach to achieve our work.

Comparing to our previous work [23,24], the contribution of this paper is that we have further extended the previous work to accommodate more general cases for mapping queries containing fuzzy semantics into UML class diagrams. Specifically, our previous work presents a base foundation by formal and solid definitions of the whole approach. This paper further verifies the feasibility of our approach and validates the powerfulness of conceptual data modeling in the area of natural language query interface.

5.2. Future directions

We will extend our future research into the following aspects:

1. By further proposing more general and efficient structures for the dictionary, CD/MF, and CD/SRF for more flexible adaptation of our approach, we will implement a prototype system to valid our methodology.
2. Capture the semantics of natural language queries with fuzzy hedges: by combining present proposed fuzzy database frameworks, for example, Zadeh [31,32], Takahashi [22] and Bosc et al. [5], our work can be extended to process a natural language query involving a modifier like ‘almost’, ‘very’, or ‘nearly’. This combination is served as a step toward analyzing the use of modifiers, which are fuzzy in natural, to communicate with fuzzy databases.
3. Consider to output query results in XML formats: XML is rapidly becoming the standard scheme for storing and exchanging structured and semi-structured information among organizations. XML schemas provide a much richer set of structures, types and constraints for describing data and are therefore expected to be the most common method for defining and validating highly structured XML documents. Therefore, we will further stretch our work to transform

```
SELECT DISTINCT jname
FROM Projects P INNER JOIN Shipments SP ON P.jno = SP.jno
                INNER JOIN Qty Q ON SP.sno = Q.sno AND
                                    SP.pno = Q.pno AND
                                    SP.jno = Q.jno
                INNER JOIN Moderately M ON Q.sno = M.sno AND
                                           Q.pno = M.pno AND
                                           Q.jno = M.jno
                INNER JOIN Suppliers S ON SP.sno = S.sno
                INNER JOIN Status ST ON S.sno = ST.sno
                INNER JOIN Good G ON ST.sno = G.sno
                INNER JOIN Parts  PT ON SP.pno = PT.pno
WHERE PT.color ='red';
```

Fig. 26. A simplified SQL statement by using inner joins.

the query results into XML documents for sending or interchanging information across the Internet.

4. Extend our work to accommodate natural language queries with time-oriented applications. Loosely speaking, a temporal database is one that contains historical data instead of, or as well as, current data [21]. We may extend our logical form to represent natural language queries posed on a temporal database that does contain some temporal data but is not limited to temporal data only. From the viewpoint of temporal databases, we may further consider to accept queries expressed in 'time-stamped form' in natural language involving some prepositions like 'on', 'since', or 'during'. For instance, by revising the example appeared in [8], the query "Did supplier S1 supply parts since July 7, 2000?", in which the statement should have a possible interpretation of a 2-tuple containing the suppliers number 'S1' and the timestamp 'July 7, 2000', and then we should add some time tuples like 'since', 'during', or '(from, to)' in the original database. This combination is served as a step toward analyzing the nature of time itself to communicate with temporal databases.

Acknowledgements

The authors would like to thank the journal editor and anonymous referees whose valuable comments and suggestions helped to improve the quality of this paper substantially.

References

[1] M. Alavi, D.E. Leidner, Review: knowledge management and knowledge management systems: conceptual foundations and research issues, MIS Quarterly 25 (1) (2001) 107–136.

[2] M. Alavi, D. Leidner, Knowledge management systems: issues, challenges, and benefits, Communications of the Associations for Information Systems 1 (1999) (article 7).

[3] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Addison-Wesley, Reading, MA, 1999.

[4] G. Booch, Object Solutions: Managing the Object-Oriented Project, Benjamin/Cummings, Menlo Park, CA, 1996.

[5] P. Bosc, M. Galibourg, G. Hamon, Fuzzy querying with SQL: extensions and implementation aspects, Fuzzy Sets and Systems 28 (3) (1988) 333–349.

[6] P.P. Chen, English sentence structure and entity-relationship diagrams, Information Sciences 29 (2–3) (1983) 127–149.

[7] P.P. Chen, The entity-relationship mode—toward a unified view of data, ACM Transactions on Database Systems 1 (1) (1976) 19–36.

[8] C.J. Date, An Introduction to Database Systems, seventh ed., Addison-Wesley, Reading, MA, 2000.

[9] L.M.G. Feijs, Natural language and message sequence chart representation of use cases, Information and Software Technology 42 (9) (2000) 633–647.

[10] K. Higa, V. Owei, A data model driven database query tool, Proceedings of the Twenty-Fourth Annual Hawaii International Conference on System Sciences, vol. 3, 1991, pp. 53–62.

[11] S. Isoda, Object-oriented real-world modeling revisited, Journal of Systems and Software 59 (2) (2001) 153–162.

[12] Ivar Jacobson, Object-Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley, Reading, MA, 1992.

[13] E. Métais, Enhancing information systems management with natural language processing technique, Data and Knowledge Engineering 41 (2–3) (2002) 247–272.

[14] Microsoft SQL Server 2000 Web Site, http://www.microsoft.com/sql

[15] A.M. Moreno, R.P. van de Riet, Justification of the equivalence between linguistic and conceptual patterns for the object model, Proceedings of the International Workshop on Applications of Natural Language to Information Systems, Vancouver, June 27–29, 1997.

[16] R.J. Muller, Database Design for Smarties: Using UML for Data Modeling, Morgan-Kaufmann, Los Altos, CA, 1999.

[17] B. Oestereich, Developing Software with UML Object-Oriented Analysis and Design in Practice, Addison-Wesley, Reading, MA, 1999.

[18] V. Owei, S.B. Navathe, H.-S. Rhee, An abbreviated concept-based query language and its exploratory evaluation, Journal of Systems and Software 63 (1) (2002) 45–67.

[19] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, Object-Oriented Modeling and Design, Prentice-Hall, Englewood Cliffs, NJ, 1991.

[20] K.J. Schmucker, Fuzzy Sets, Natural Language Computations, and Risk Analysis, Computer Science Press, Rockville, MD, 1984.

[21] R.T. Snodgrass, Developing Time-Oriented Database Applications in SQL, Morgan-Kaufmann, San Francisco, CA, 1999.

[22] Y. Takahashi, Fuzzy database query languages and their relational completeness theorem, IEEE Transactions on Knowledge and Data Engineering 5 (1) (1993) 122–125.

[23] F.S.C. Tseng, A.L.P Chen, W.P. Yang, On mapping natural language constructs into relational algebra through E–R representation, Data and Knowledge Engineering 9 (1) (1992/93) 97–118.

[24] F.S.C. Tseng, C.L. Chen, Extending the UML concepts to capture natural language semantics for database access, Technical Report, National Kaohsiung First University of Science and Technology, 2005.

[25] M. Umano, Relational algebra in fuzzy database, IEICE Technical Report DE86-4, vol. 86, No. 192, 1986, pp. 1–8.

[26] P. Velardi, Natural language interfaces to databases: features and limitations, Proceedings of the Seventh International Conference on Entity-Relationship Approach, 1986.

[27] P.H. Winston, Artificial Intelligence, Addison-Wesley, Reading, MA, 1992.

[28] A. Yazici, D. Cibiceli, An access structure for similarity-based fuzzy databases, Information Sciences 115 (1–4) (1999) 137–163.

[29] L.A. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353.

[30] L.A. Zadeh, The concept of a linguistic variable and its application to approximate reasoning, Information Sciences 8 (3) (1975) 199–249.

[31] L.A. Zadeh, PRUF—a meaning representation language for natural language, International Journal of Man–Machine Studies 10 (1978) 395–460.

[32] L.A. Zadeh, The concept of a linguistic variable and its application to approximate reasoning (I), Information Sciences 8 (3) (1975) 199–249.

[33] M. Zemankova-Leech, A. Kandel, Fuzzy Relational Database—A Key to Expert System, Verlag TUV Rheinland, Cologne, Germany, 1984.