A Domain Independent Natural Language Interface to Databases Capable of Processing Complex Queries*

Rodolfo A. Pazos Rangel[1], Joaquín Pérez O.[1], Juan Javier González B.[2], Alexander Gelbukh[3], Grigori Sidorov[3], and Myriam J. Rodríguez M.[2]

[1] National Center for Investigation and Technological Development (CENIDET)
{pazos, jperez}@cenidet.edu.mx
[2] Technological Institute of Cd. Madero, México
{jjgonzalezbarbosa, myriam_rdz}@hotmail.com
[3] Center for Computing Research(CIC), National Polytechnic Institute(IPN), México
{gelbukh, sidorov}@cic.ipn.mx

Abstract. We present a method for creating natural language interfaces to databases (NLIDB) that allow for translating natural language queries into SQL. The method is domain independent, i.e., it avoids the tedious process of configuring the NLIDB for a given domain. We automatically generate the domain dictionary for query translation using semantic metadata of the database. Our semantic representation of a query is a graph including information from database metadata. The query is translated taking into account the parts of speech of its words (obtained with some linguistic processing). Specifically, unlike most existing NLIDBs, we take seriously auxiliary words (prepositions and conjunctions) as set theory operators, which allows for processing more complex queries. Experimental results (conducted on two Spanish databases from different domains) show that treatment of auxiliary words improves correctness of translation by 12.1%. With the developed NLIDB 82of queries were correctly translated (and thus answered). Reconfiguring the NLIDB from one domain to the other took only ten minutes.

1 Introduction

Access to information in a fast and reliable way is very important for modern society. Natural Language Interfaces to Databases (NLIDBs) permit users to formulate queries in natural language, providing access to information without requiring knowledge of programming or database query languages.

However, despite the achievements attained in this area, present day NLIDBs do not guarantee correct translation of natural language queries into database languages [1]. Moreover, queries are limited to the database domain configured by the database administrator. A Survey [9] on the importance of natural language processing (NLP) systems, conducted on 33 members of the "Pittsburg

* This research was supported in part by COSNET and RITOS2.

© Springer-Verlag Berlin Heidelberg 2005

Large User Group" professional society, mentions that:(1) NLIDBs are the most useful application for organizations among all NLP systems, (2) The five most desirable capacities of NLIDBs are: efficiency, domain independence, pronoun handling, understanding of elliptical entries (i.e., implied words), and processing of sentences with complex nouns, and (3) 50% of the best NLIDBs are those that offer domain independence.

This paper describes an approach that uses database semantic metadata to perform the analysis of nouns and auxiliary words (prepositions and conjunctions). This allows for translation of queries expressed in natural language (Spanish in our case) into SQL, with easy adaptation to different domains.

2 Related Work

Most existing NLIDBs do not carryout any real semantic processing of the user's input. They just look for keywords in the sentence [6] and focus their analysis on nouns and verbs, ignoring any auxiliary words in the query [7, 8].

In some NLIDBs, the semantic analyzer uses syntactic structure (obtained through a syntactic parser) to extract the meaning of the sentences [10]. However, no significant success is achieved yet in this direction. Semantic analysis of sentences is still a very complex task [11]: say, just determining the meaning of words is difficult due to their polysemy; for example, *file* is a tool and also a place for keeping documents.

In many NLIDBs that do use semantic analysis of a natural language query, it involves looking for keywords in the input sentence using a predefined pattern through multiple database mappings. Still, this approach is not sufficiently specific to give good results. In other systems, semantic analysis is based on probabilistic models [3, 4, 6]. Such systems rely on a corpus labeled with semantic information; however, there are no sufficiently large semantically marked corpora for use in NLIDBs.

Some NLIDBs use semantic graphs. However, the database relationships have to be defined by the user [6]. These approaches are subjective and require considerable manual effort. Such techniques have been applied to specific tasks in restricted semantic domains. They use a semantic representation, usually case frames [5].

Thus, no existing approach achieves good results in semantic analysis. That is why methods for its improvement are very important. In particular, auxiliary words (like prepositions and conjunctions) are not well-studied for tasks of processing natural language queries.

3 Assumptions

We assume that the database satisfies the following conditions, reasonable in well- designed databases: (C1) Relational, entity-relationship or a similar model is used; (C2) Each table has an explicit primary key; (C3) Each table column is explicitly associated to a domain (a named set of legitimate values for column values); (C4) Referential relationships between tables are explicitly expressed through foreign keys; (C5) Each table and column has a textual description; (C6) All tables are in second normal form; (C7) Information on conditions (C2) to (C5) can be extracted from the database metadata. Since descriptions of tables and columns are crucial for correct interpretation and translation of queries. Thus we additionally assume the following conditions: (C8) Descriptions of tables and columns are lexically and syntactically correct; (C9) Descriptions of tables and columns are short but meaningful phrases that consist of at least one noun and optionally of several meaningful words (nouns or adjectives; we do not take verbs into account), and optionally several auxiliary words (such as articles, prepositions, and conjunctions); (C10) The most meaningful word in the description of a table or a column is a noun; (C11) The description of each table is different from that of any other table or column; (C12) The description of each column is different from that of any other column of the same table (columns in different tables may have the same description); (C13) The description of each column that participates in a foreign key includes the description of the table referred to by the foreign key; (C14) The description of each column that participates in a primary key includes the description of the table, except for columns participating in a foreign key; (C15) The description of a column that does not participate in a primary or foreign key does not include the description of any table.

Though conditions (C8) to (C15) might seem restrictive, most of them would be required by humans to understand a database and correctly formulate SQL queries. Additionally, we make the following assumptions: (A1)Query sentences are lexically and syntactically correct (which can be checked by a syntactic analyzer);(A2)Queries are expressed in interrogative form.

We propose automatic creation of a domain dictionary from a synonym dictionary and metadata of the target database, using some linguistic processing. This technique performs the translation process independently of the NLIDB working data, thus avoiding reprogramming the NLIDB to port it to a database of different domain.

4 Generation of the Domain Dictionary

Dictionaries used by existing NLIDBs are created manually or semi-automatically [2]. We suggest automatic generation of the domain dictionary from a synonym dictionary and the database metadata with the help of some linguistic processing.

Synonym dictionary. In our case, we extracted a general synonym dictionary from a digital encyclopedia. It currently has 20,000 words with their synonyms and antonyms. This dictionary can be immediately used for most domains; however, it has an interface that permits to add more words.

Metadata dictionary. Database metadata can be used as a resource for interpretation of a query in a restricted domain [12]. The metadata dictionary stores database information such as number of tables, number of columns, and their location; additionally, for each table it stores the name of each column along with its data type and description.

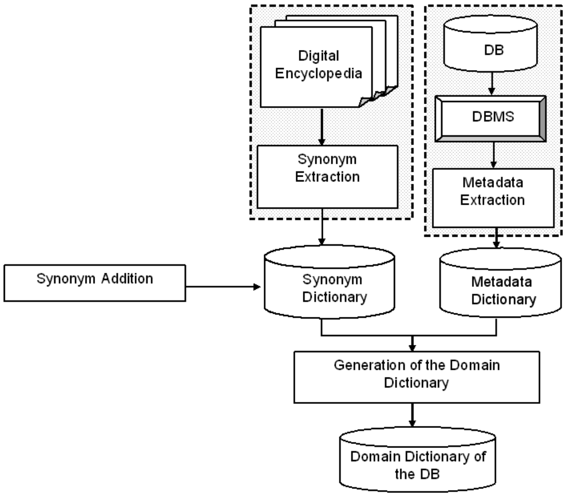

Fig. 1. Generation of the Domain Dictionary

Domain dictionary. For automatic generation of the domain dictionary, the description of each column from the metadata dictionary is processed to obtain the lemma and the part of speech (POS) of each word in the description (a POS tagger or a syntactic analyzer is used to resolve POS ambiguity). Then each noun is associated with columns and tables whose description includes this noun or its synonym. Notice that it is easier to provide meaningful descriptions for columns and tables (so that the interface can be configured automatically using this information) than to manually configure the interface dictionaries and modules for it to recognize and relate each column and table with some word in the domain dictionary.

5 Query Preprocessing

The preprocessing consists of analyzing each word of the natural language query to obtain its lexical, syntactic, and semantic information. Lexical information consists of the lemma of each word; the syntactic information consists of its part of speech (verb, noun, preposition, etc.). Semantic information is obtained from the domain dictionary in such a way that each noun is related to a set of columns and tables to which it may refer. Table 1 shows an example of a tagged query.

Table 1. Semantic information obtained after preprocesing

QUERY: *cuáles son los nombres y las direcciones de los empleados* (what are the names and addresses of the employees)

Word	Lema	Morphosyntactic information	Columns	Tables
cuáles(what)	cuál (which)	interrogative pronoun		
son (are)	ser (be)	verb, indicative, 3^{rd} person, plural		
los (the)	el (the)	plural, masculine, determinative		
nombres (names)	nombre (name)	plural, masculine, noun	Categories.CategoryName, Customers, CompanyName, Employees.FirstName, Orders.ShipName,...	
y (and)	y (and)	conjunction		
direcciones (addresses)	dirección (address)	plural, femenine, noun	Employee.Address, Orders.ShipAddress, Suppliers.Address, Customers.Address	
de (of)	de (of)	preposition		
los (the)	el (the)	plural, masculine, determinative		
empleados (employees)	empleado (employee)	plural, masculine,noun	Employee.EmployeeID, Orders.EmployeeID	Employee

6 Main Algorithm

The translation process is carried out in three phases: (1) identification of the `select` and `where` phrases; (2) identification of tables and columns, and (3) construction of the relational graph.

Phase 1: Identification of the select and where phrases. The query phrases that define the SQL `select` and `where` clauses are identified in order to pinpoint the columns (and tables) referred to by these phrases. Since these clauses always involve table columns, then, according to assumption (C9) above, the phrases are query subphrases that include at least one noun (and possibly prepositions, conjunction, articles, adjectives, etc.). Additionally, from assumption (A2) it follows that the phrase that defines the `select` clause always precedes the phrase that defines the `where` clause. In Spanish, the words that separate these phrases are: verbs, *cuyo* (whose), *que* (that), *con* (with) *de* (from, with), *donde* (where), *en* (in, on, at), *dentro de* (inside), *tal que* (such that), etc.

Phase 2: Identification of tables and columns. Usually each noun in the `select`/`where` phrases refers to several database columns or tables (see Table 1), which would yield several translations of the query. Therefore, in order to pinpoint the columns and tables referred to, it is usually necessary to analyze the preposition *de* (of) and conjunction *y* (and), since they almost always appear in `select/where` phrases expressed in Spanish [8]. Examination of prepositions and conjunctions permits, besides considering the meaning of individual nouns, to determine the precise meaning of a `select/where` phrase that involves nouns related by prepositions and conjunctions. For this, preposition *de* (of) and conjunction *y* (and) are represented by operations using set theory, because of the role they play in queries.

Preposition *de* (of) establishes a close relationship between a word and its complement [14], such that, if there exists a `select/where` phrase that includes two nouns p and q related by preposition *de* (of), then the phrase refers to the common elements (columns or tables) referred to by $p$and q. Formally, $S(p\ prep\_de\ q) = S(p) \cap (q)$, where $S(x)$ is the set of columns or tables referred to by phrase x. Conjunction y (and) expresses the notion of addition or accumulation [14], such that if there is a `select` phrase that involves two nouns p and q related by conjunction y (and), then the phrase refers to all the elements referred to by p and q. Formally, $S(p\ conj\_y\ q) = S(p) \cup (q)$. Conjunction y (and) in a `where` phrase is treated as a Boolean operation.

For example, consider the query: *cuáles son los nombres y direcciones de los empleados* (which are the names and addresses of the employees), see Table 1. Consider the `select` phrase *nombres y direcciones de los empleados* (names and addresses of the employees). According to the above explanation, to extract the meaning of the `select` phrase it is necessary to apply two set operations: a union, corresponding to the conjunction y (and), and an intersection, corresponding to the preposition *de* (of). A heuristics is applied to determine the order of the two operations. In this case the preposition *de* (of) applies to the two nouns (*names and addresses of the employees = names of the employees and addresses of the employees*), therefore, the intersection operation has precedence above the union.

The output of Phase 2 is the semantic interpretation of the `select` and `where` phrases (i.e. the columns and tables referred to by these phrases), which will be used in Phase 3 to translate them into the `select` and `where` clauses of the SQL statement. The process for determining the tables and columns is the following (we rely on conditions (C8) to (C15)):

1. If a major POS word (usually noun) in the `select` phrase refers only to a table (and not to another table or column) then the table is permanently marked and associated to this word. If it refers to several tables, then distinguishing major POS words are extracted from the table descriptions and looked for in the `select` phrase in order to find out which table(s) are referred to by the first major POS word. If only one table is found, it is permanently marked and associated with the first word; otherwise the tables found are temporarily marked and associated to the first word.
2. If a major POS word in the `select` phrase refers only to a column (and not to another table or column), then the column is permanently marked and associated to the word and the corresponding table is also permanently marked and associated to the word. Otherwise, if the major word refers to several columns, then the analysis of preposition *de* (of) and conjunction *y* (and) described above is carried out to determine which column(s) are

referred to by the first major POS word. If only one column is found, it is permanently marked and associated to the first word; otherwise the columns found are temporarily marked and associated to the first word.

3. If a major POS word in the `where` phrase refers only to a table (and not to another table or column), then the table is permanently marked and associated to this word. Otherwise, if the word refers to several tables, then distinguishing major POS words are extracted from the table descriptions and looked for in the `select` phrase to determine which table(s) are referred to by the first major POS word. If only one table is found, it is permanently marked and associated to the first word; if several tables are found but one of them has been permanently marked, it is associated to the first word; otherwise the tables found are temporarily marked and associated to the first word.
4. If a major POS word in the `where` phrase refers only to a column (and not to another table or column), then the column is permanently marked and associated to the word and the corresponding table is also permanently marked and associated to the word. Otherwise, if the word refers to several columns, then the analysis of preposition *de* (of) and conjunction *y* (and) described in the previous paragraphs is carried out in order to find out which column(s) are referred to by the first major POS word. If only one column is found, it is permanently marked and associated to the first word; if several columns are found but one of them has been permanently marked, it is associated to the first word; otherwise the columns found are temporarily marked and associated to the first word.

At the end of this process, if there are no temporarily marked columns and tables, then we can proceed with the analysis; otherwise the analysis is aborted.

Phase 3: Construction of the relational graph. The process for constructing the relational graph is as follows:

1. Considering condition (C1), a non-directed graph is constructed from the relational or entity-relationship model of the database. Each node represents a table and each arc represents a referential relationship between tables (from condition C4). We assume binary relationships (involve two tables); this is not a serious limitation since a relationship involving more than two tables $(T_1, T_2, ..., T_n)$ can always be substituted by an auxiliary table T_a with binary relationships with tables $T_1, T_2, ..., T_n$.
2. The nodes corresponding to the tables permanently marked in Phase 2 are marked. Afterwards, for each simple selection condition in the `where` phrase that involves one column of a table (for instance: *con órdenes para el barco Mercury* (with orders for Mercury ship)), the node corresponding to the table is labeled with its corresponding simple selection condition. Finally, each marked node is labeled with the columns (of the corresponding table) referred to in the `select` phrase.
3. For each simple selection condition in the `where` phrase that involves columns of two tables, the arc incident to the nodes representing the tables is marked;

if no such arc exists, it is added to the graph and marked. Each arc marked at this step is labeled with its corresponding simple selection condition.

4. If all the selection conditions are explicitly stated in the query phrase then the subgraph consisting of all marked nodes and arcs must be a connected graph. From this sub-graph it is easy to obtain the translation into an SQL expression.
5. A disconnected sub-graph means that there exist implicit selection conditions in the query or the query is incorrectly stated. In the first case, the NLIDB has to guess the implicit selection conditions. For this a heuristics is used which based on the following assumption: all the implicit selection conditions refer to natural joins that involve tables and columns participating in a referential relationship. Therefore, a connected sub-graph is constructed by adding unmarked arcs to the disconnected sub-graph so that the number of unmarked arcs added is minimal. From this sub-graph the translation into an SQL expression is straightforward. If no connected sub-graph can be constructed, then the query is reported as incorrect.

7 Experimental Results

There is no standard evaluation method for comparing and assessing NLIDBs. The most used criterion is the translation success; i.e., the semantic equivalence between the natural language query and the SQL statement [13]. Up to now most NLIDBs can satisfactorily translate queries involving several tables if they are explicitly mentioned in the query, or queries involving one table that is not mentioned explicitly. For the experiment, the Northwind and the Pubs databases of SQL Server 7.0 were used, and 50 users were asked to formulate queries in Spanish. The resulting corpus consists of 198 different queries for the Northwind database and 70 different queries for the Pubs database. For formulating their queries the users only were allowed to see the databases schemas (definitions). The queries were classified according to difficulty (which depends on the amount of implicit column and table information in the query and special functions) and were divided into six types: (1) explicit table and column information, (2) explicit table information and implicit column information, (3) implicit table information and explicit column information, (4) implicit table and column information, (5) special functions required, and (6) impossible or difficult to answer. Table 2 shows the results obtained for the Northwind database with all the queries, which involve one, two, or more tables; in this case the success rate was 84%. Similar experiments were conducted on the Pubs database. Table 3 shows the results considering all the queries; in this case the success rate is 80%, which is very similar to that for the Northwind database. It is worth mentionary that most of the queries to the Pubs database involve two or more tables.

Additional experiments were conducted in order to assess the impact of the analysis of prepositions and conjunctions (described in Phase 2 in Section 5) on the translation success. When this analysis was excluded from the translation process, the success rate was 72.6% for the Northwind database and 67.1% for the

Table 2. Results obtained for the Northwind database

Query Results	**Query Type**						**Total**	%	%
	1	**2**	**3**	**4**	**5**	**6**			
Answered correctly	31	57	19	49	0	0	156	79	**84**
Answered with additional information	0	0	5	5	0	0	10	5	
Incorrect answer	0	0	0	1	23	5	29	15	16
Unanswered	0	0	0	3	0	0	3	1	
Total	31	57	24	58	23	5	198	100	100

Table 3. Results obtained for the Pubs database

Query Results	**Query Type**						**Total**	%	%
	1	**2**	**3**	**4**	**5**	**6**			
Answered correctly	7	29	8	12	0	0	56	80	**80**
Answered with additional information	0	0	0	0	0	0	0	0	
Incorrect answer	0	0	0	1	10	1	12	17	20
Unanswered	0	0	0	0	1	2	3	20	
Total	7	29	8	13	11	2	70	100	100

Pubs database. When the analysis was enabled, the success rate increased 11.4% for the Northwind database and 12.9% for the Pubs database. Some examples of queries that could not be satisfactorily translated are the following: *cuál es la fecha del último envío* (what is the date of the last shipment) and *cuáles son los nombres de los empleados que nacieron en febrero* (what are the names of the employees born in February). The first query could not be answered because *last shipment* is not defined in the domain and the second because it needs a special function for extracting the month from a date in the `where` clause of an SQL statement.

8 Conclusions

The translation approach presented favors domain independence, since the NLIDB does not need to be manually configured with a set of keywords for carrying out specific actions. It is important to point out that configuring the NLIDB for another domain (from Northwind to Pubs) took only ten minutes. The tests conducted so far have shown that the proposed approach permits: (1) avoiding the wearisome process of configuring the NLIDB for a given domain and (2) obtaining good results in translating natural language queries into SQL statements. The databases used for the tests have been used by other NLIDBs [13], which sets the foundation for designing a metric to compare the results of our NLIDB versus others. Prepositions and conjunctions play a key role in extracting the meaning of a query in Spanish. Taking them into account as set operations (intersection and union) in-creases the success rate by 12.1%.

References

1. Popescu, A.M., Etzioni, O., Kautz, H.: Towards a Theory of Natural Language Interfaces to Databases. In: Proceedings of the 2003 International Conference on Intelligent User Interfaces. ACM Press, 2003.
2. Zarate, A., Pazos, R., Gelbukh, A., Padrón, I.: A Portable Natural Language Interface for Diverse Databases Using Ontologies. LNCS 2588, 2003.
3. Stallard, M.S., Bobrow, D., Schwartz, R.: A Fully Statistical Approach to Natural Language Interfaces. In: Proc. 34th Annual Meeting of the Association for Computational Linguistics. 1996. http://citeseer.nj.nec.com/miller96fully.html.
4. Minker, W.: Stochastically-Based Natural Language Understanding across Task and Languages. In: Proc of EuroSpeech97, Rodas, Greece. 1997. http://citeseer.nj.nec.com/ minker97stochasticallybased.html.
5. Moreno, L., Molina, A.: Preliminares y Tendencias en el Procesamiento del Lenguaje Natural. Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. http://www3.unileon.es/dp/dfh/Milka/MR99b.pdf.
6. Meng, F., Chu, W.W.: Database Query Formation from Natural Language Using Semantic Modeling and Statistical Keyword Meaning Disambiguation. Computer Science Department. University of California. www.cobase.cs.ucla.edu/tech-docs/ucla-990003.ps.
7. InBase-Online. English Queries to Personnel DB. Russian Research Institute of Artificial Intelligence. 2001. http://www.inbase.artint.ru/nl/kadry-eng.asp.
8. Montero, J.M.: Sistemas de Conversión Texto Voz. B.S. thesis. Universidad Politécnica de Madrid. http://lorien.die.upm.es/ juancho.
9. Sethi, V.: Natural Language Interfaces to Databases: MSI Impact, and Survey of their Use and Importance. 1986. University of Pittsburgh.
10. AVENTINUS - Advanced Information System for Multinational Drug Enforcement. http:// www.dcs.shef.ac.uk/nlp/funded/aventinus.html.
11. Sidorov, G.: Problemas Actuales de Lingüística Computacional. In: Revista Digital Universitaria, Vol. 2, No. 1. 2001. http://www.revista.unam.mx/vol.2/num1/art1.
12. Stratica, N., Kosseim, L., Desai, B.: NLIDB Templates for Semantics Parsing. In: Proceedings of Applications of Natural Language to Data Bases (NLDB 2003). pp 235-241. http//www.cs.concordia.ca/ kosseim/research.html.
13. ELF Software Co.: Results from the Head to Head Competition. 2001. http://elf-soft.com/ns/demos.htm.
14. Real Academia Española: Gramática Descriptiva de la Lengua Española. Espasa Calpe, 1999.