Piotr PECHMANN
WEST POMERANIAN UNIVERSITY OF TECHNOLOGY, SZCZECIN, FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY,
ul. Żołnierska 49, 71-210 Szczecin

PNL/JDO – Polish Natural Language Interface to JDO Compliant Databases

Ph.D. Piotr PECHMANN

Graduated from the Faculty of Computer Science and Information Technology, Szczecin University of Technology in 1996. In 2006 obtained Ph.D. degree in the field of computer science (specialty: artificial intelligence). Main research interests: natural language user interfaces, intelligent agent systems. Employed as Associate Professor at the Faculty of Computer Science and Information Technology, West Pomeranian University of Technology, Szczecin.

e-mail: ppechmann@wi.zut.edu.pl

Abstract

Database systems are nowadays used in almost all kinds of professional activities. However, standard means for obtaining data from such systems are either too difficult to learn by average user (SQL) or offer limited capabilities (form-based interfaces). HCI experts agree that one of the best solutions to that problem are natural language interfaces. The paper presents one of the solutions of that kind: Polish natural language interface to object-oriented databases compliant with JDO standard.

Keywords: natural language interfaces, object-oriented databases.

PNL/JDO – bazujący na naturalnym języku polskim sprzęg użytkownika do baz danych w standardzie JDO

Streszczenie

Systemy bazodanowe stosowane są obecnie w niemal wszystkich obszarach ludzkiej aktywności zawodowej. Standardowe środki dostępu do zgromadzanych w nich informacji są jednak albo trudne do opanowania przez przeciętnego użytkownika (np. język SQL) albo oferują ograniczone możliwości (np. formularze). Za jedne z lepszych rozwiązań uważane są sprzęgi bazujące na zapytaniach w języku naturalnym. W publikacji przedstawiono propozycję tego typu sprzęgu dla języka polskiego i obiektowych baz danych w standardzie JDO. Podstawę rozwiązania stanowi autorski model A-O-I będący uogólnionym modelem procesu analizy i interpretacji komunikatów (poleceń, zapytań lub odpowiedzi) w języku naturalnym kierowanych przez użytkownika do systemu informatycznego. Sprzęg realizowany według modelu A-O-I jest trójwarstwowy i składa się z: a) analizatora składniowo-semantycznego, b) opisu rozpoznanego znaczenia komunikatu w postaci obiektowej reprezentacji semantyki (ORS) oraz c) interpretera ORS. W prezentowanym rozwiązaniu podstawowym zadaniem realizowanym przez interpreter ORS jest generacja odpowiedniego zapytania w języku JDOQL na podstawie rozpoznanego znaczenia zapytania w języku polskim. W kolejnych punktach artykułu omówione zostały: ogólny model sprzęgu i jego główne składowe, prototyp systemu implementującego ten model oraz wyniki testów skuteczności i wydajności proponowanego rozwiązania. W podsumowaniu wskazane zostały kierunki dalszych badań.

Słowa kluczowe: interfejsy użytkownika bazujące na języku naturalnym, obiektowe bazy danych.

1. Introduction

Databases belong to the most important types of computer systems used in various areas of our civilization. It would be difficult to point to an area of life where it would not be necessary to gather large amounts of data. One of the most important tasks that IT experts face is to provide effective means of accessing the data stored in the databases.

This mission takes on special significance particularly today, when many people who need to communicate with the database systems have little or no computer experience. In practice, SQL language, which is commonly used in communication with databases, remains a province of a narrow group of specialists, i.e. IT experts. It turns out, however, that many of common users also find it difficult to operate the form-based interfaces which are created especially for their use. Efficient and effective use of these interfaces is even more difficult for “an average user”.

Clearly, from the point of view of most users, the best solution would be equipping database systems with ability of natural language communication. Users know their natural language very well, moreover, natural languages have great power of expression by which almost any problem can be described.

Because of those reasons, many projects aimed at developing effective ways of communication with databases using queries in a natural language have been conducted out all over the world for many years [1]. Most of those projects were led for English language (e.g. [3, 5, 10]) but one can also find solutions developed for other natural languages, for example Spanish [6] or German [4].

Unfortunately, as of today only two such projects were carried out for the Polish language [2, 7]. What is more, both of them dealt with communication with relational databases. Thus no solution were available for the Polish language for communication with object-oriented databases. It was the need of such a solution that inspire the author of this paper to undertake research aimed at finding one.

As of today there exists no commonly accepted standard for object-oriented databases. The JDO standard has been chosen for the project presented in this paper as it seems to be one of the best established standards. The query language used for that standard is JDOQL [11].

2. The PNL/JDO Model

The proposed solution is based on the A-O-I model, which was developed by the author of this paper [7]. It is a generalized model of the process of analysis and interpretation of messages (orders, queries or answers) in a natural language addressed by the user to the computer system. In this model three layers have been distinguished. The acronym A-O-I was formed from the first letters of the names of the corresponding layers:

a) The first layer is *semantic analyzer*. Its function is to perform the syntactic and semantic analysis of natural language sentences [7], which are the user’s messages, and generating description of the meaning of those messages.
b) The second layer is a description of the meaning of a message recognized by the system; an object-oriented model of description of the sentence’s meaning, called *object-oriented representation of semantics* (ORS) has been developed especially for the A-O-I [7, 8].
c) The final, third layer in the A-O-I model is an *ORS interpreter*. It analyzes an ORS and (based on the results of this analysis) it generates messages and control instructions expressed in the internal language of a computer system.

In the presented model, the basic function performed by the ORS interpreter is generating the query in the JDOQL language that corresponds to the recognized meaning of the user’s query in Polish. The diagram depicting the structure of a system implementing the model is shown in Figure 1.

The function performed by the *user interface module* is conducting the process of communication with the user. It includes a) receiving a query in Polish and sending it to *semantic analyzer module* in the text form, and b) presenting the results obtained on the base of the execution a query.

As the result of the syntactic-semantic analysis process of a received query in the semantic analyzer module the meaning of that query is identified. The description of that meaning is then sent to the *query processing module* in the ORS form.

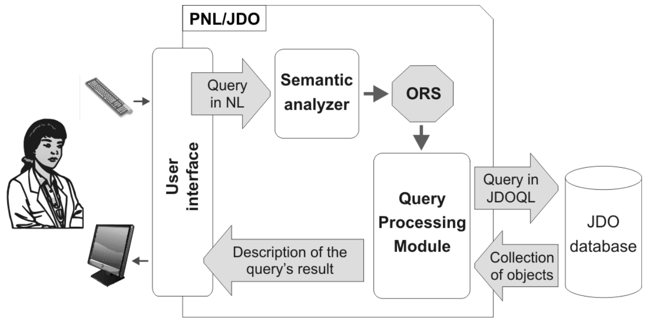

Fig. 1. Diagram of the system implementing the PNL/JDO model
Rys. 1. Schemat systemu implementującego model PNL/JDO

2.1. Query processing module

Query processing module (QPM) is the last, third layer in the PNL/JDO model. It corresponds to the ORS interpreter in the general A-O-I model. The structure of the QPM is depicted in diagram shown in Fig. 2.

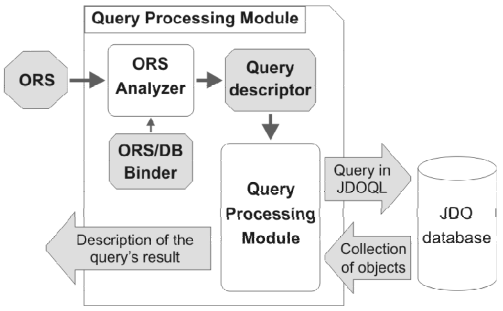

Fig. 2. Diagram of the query processing module
Rys. 2. Schemat modułu przetwarzania zapytań

The following three layers were distinguished in the model of the query processing module:
a) ORS analyzer,
b) query descriptor,
c) query processor.

ORS analyzer

The function of the *ORS analyzer* (ORS-A) is recognizing to which objects and their attributes from the database refers the query given by the user. The ORS-A module performs such recognition using information provided in the *ORS/DB binder*, which is a list of structures called *binding descriptors* (BD). Each BD bounds particular class and/or its attributes with corresponding ORS fragments represented by one or more *object representation descriptors* (ORD) [7]. For example, the BDs shown in Fig. 3 determine such bindings for class `Patient` and its attributes `LastName` and `Age`.

```
<BindingDescr for="pp.dbEntities.Patient">
 <ORDList>
  <ORD value="otPacjent"/>
 </ORDList>
</BindingDescr>
...
<BindingDescr for="pp.dbEntities.
                      Patient[LastName]">
 <ORDList>
  <ORD value="otPatient[satLastName]"/>
  <ORD value="otPatient:otLastName[satLastName]"/>
  <ORD value="otInformation:otPatient"/>
 </ORDList>
</BindingDescr>
...
<BindingDescr for="pp.dbEntities.Patient[Age]">
 <ORDList>
  <ORD value="otAge:otPatient"/>
  <ORD value="otInformation:otPatient"/>
 </ORDList>
</BindingDescr>
```

Fig. 3. Examples of ORS/DB binders
Rys. 3. Przykłady deskryptorów wiązań ORS/DB

Details of the recognition process performed by ORS-A can be found in [9].

Query descriptor

The set of pieces of information defining the details of a query is saved by the ORS analyzer in a structure called *query descriptor* (QD). Such descriptor contains:
- the name of the class of which objects the query refers to,
- the information about conditions imposed on the value of one or two attributes of the searched objects; the condition is defined by a triple: ⟨ *attribute_name*, *operator*, *attribute_value* ⟩,
- the name of the attribute the query refers to,
- the information regarding the order of result preferred by the user in case the result contains a collection of objects.

An example of query descriptor can be found in Fig. 4. Part (a) of that figure shows the ORS representation of the meaning generated by the semantic analyzer for the query *Podaj wiek pacjenta Piotra Pechmanna* (*Give me the age of patient Piotr Pechmann*). QD generated by the ORS-A module based on that ORS and information defined in the ORS/DB binder is shown in the part (b) of the figure.

a)

$$\underline{R} = [\ !\ |\ \mathit{rtGive}.\langle\ \mathrm{AP}\{\ \},\ \mathrm{ARG}\{\ [\ \mathrm{otAge} : \mathrm{otPatient}.\{\mathrm{satLastName},\ \mathrm{satFirstName}\}\ ]_{\mathrm{srtObject}}\ \},\ \varnothing\ \rangle\ ]$$

b)

```
<QueryDescriptor>
  <Class name="pp.dbEntities.Patient"/>
  <ConstraintAttribute_1 name="LastName"
                         operator="="
                         value="Pechmann"/>
  <ConstraintAttribute_2 name="FirstName"
                         operator="="
                         value="Piotr"/>
  <SearchedAttribute name="Age"/>
  <ResultsOrder value="ascending"/>
</QueryDescriptor>
```

Fig. 4. Example of a query descriptor (b) corresponding to an ORS describing the meaning of a natural language query (a)
Rys. 4. Przykład deskrypta zapytania (b) odpowiadającego ORS opisującemu znaczenie zapytania w języku naturalnym (a)

Query processor

The last element of the QPM is the *query processor*. It performs two tasks based on the information contained in the query descriptor:
1) it generates an appropriate query in JDOQL language and sends it to the database system;
2) it: a) receives a collection of objects returned by the database and retrieves from that objects the values of those attributes which the user's query refers to, b) generates an answer, and c) transmits it to the user interface module.

The processes of generating JDOQL queries and constructing answers have been described in details in [9].

2.2. Prototype of the system

The solution presented here has been experimentally verified. It was implemented in a prototype of a system handling the process of communication with an object-oriented database using queries in Polish. The prototype was created in Java. ObjectDB server [12] was used as database server. Tests were conducted on a computer with Pentium 4 1,4 GHz processor and 256 MB RAM working under Windows XP system. Test database contained example information related to medicine.

The tests confirmed effectiveness of the developed algorithms. The JDOQL equivalents were correctly generated for simple queries such as *Czy pacjent Pechmann był chory na grypę?* (*Has patient Pechmann suffered from flu?*) as well as for more complex queries like *Podaj szczegóły ostatniego pobytu w sanatorium pacjenta Pechmanna* (*Show the details of patient Pechmann's last sanatorium stay*). Data filtering from the collection of objects returned by the database system was also performed correctly.

The prototype efficiency turned out to be promising as well. Communication with the system running on the platform described above took place in real time: the complete processing of each of the tested queries never exceeded one second. The execution time of semantic analysis process and generating the appropriate JDOQL query oscillated between 180 ms for simple queries such as *Podaj adres pacjenta Pechmanna* (*Show patient Pechmann's address*) up to 540 ms for queries such as *Podaj szczegóły ostatniego pobytu w sanatorium pacjenta Pechmanna* (*Show the details of patient Pechmann's last sanatorium stay*).

3. Further research

Further development of the proposed solution is related to increasing the scope of sentence schemes which can be used to construct natural language queries directed to the system. Current models of processes of syntactic-semantic analysis and ORS analysis accept simple sentences with no conjunction structures. Adding algorithms for handling compound sentences to those models as well as a possibility of using conjunctions will certainly increase the liberty in formulating queries in the Polish language. It will require, however, further development of the model of ORS analyzer, which is a part of query processing module.

Research aimed at boosting the system efficiency is also planned. Although, at present, its efficiency is sufficient for computer systems working on PC workstations, higher efficiency would certainly be an advantage in server systems. The basic source of increase of efficiency could be parallelization of the ORS analysis process aimed at identification of objects and their attributes referred to in a user's query.

A distinct increase of efficiency could be also achieved through changing the form of identifiers being a part of the object representation descriptors used in the ORS analysis process [7, 9]. In implementation of the PNL/JDO model applied in the prototype the identifiers had a form of character strings. Such form is convenient for developers describing bindings between ORS and elements of a database structure, but it would be much more effective to compare the identifiers having the form of integer numbers. Thus a conversion of all the identifiers retrieved from the database structure description to the integer-based form is planned.

The work was supported by finances of West Pomeranian Provincial Administration.

4. References

[1] Androutsopoulos I., Ritchie G.: Database Interfaces. [In:] Handbook of Natural Language Processing, (red.:) Dale R., Moisl H., Somers H., Marcel Dekker, 2000.

[2] Bach M.: Methods of constructing tasks of database searching within a process of translation of queries in natural language. Doctoral dissertation at the Faculty of Automatic Control, Electronics and Computer Science, The Silesian University of Technology, Gliwice, 2004, (in Polish).

[3] El-Mouadib F., Zubi Z., Almagrous A., El-Fegh I.: Generic interactive natural language interface to databases (GINLIDB). [In:] Proceedings of the 10th WSEAS international conference on evolutionary computing, WSEAS Press, 2009.

[4] Küpper D., Storbel M., Rösner D.: NAUDA: A cooperative natural language interface to relational databases. [In:] "Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data", ACM Press, 1993.

[5] Minock M., Olofsson P., Näslund A.: Towards Building Robust Natural Language Interfaces to Databases. [In:] "Proceedings of the 13th international conference on Natural Language and Information Systems: Applications of Natural Language to Information Systems", Lecture Notes In Computer Science. Vol. 5039, Springer-Verlag, 2008.

[6] Pazos Rangel R., Gelbukh A., González Barbosa J., Alarcón Ruiz E., Mendoza Mejía A., Domínguez Sánchez P.: Spanish Natural Language Interface for a Relational Database Querying System. [In:] "Proceedings of the 5th International Conference on Text, Speech and Dialogue", Lecture Notes In Computer Science. Vol. 2448, Springer-Verlag, 2002.

[7] Pechmann P.: An object-oriented model of the semantic analysis process of Polish natural language queries to medical databases. Doctoral dissertation at the Faculty of Computer Science and Information Systems, Szczecin University of Technology, 2006, (in Polish).

[8] Pechmann P.: The Model of Object-oriented Representation of Semantics. [In:] Proceedings of the Conference "SMI 2007", Polish Journal of Environmental Studies, Vol. 16, No. 4A, 2007.

[9] Pechmann P., Kiriłow P.: Application of the A-O-I model in communication with a JDO compliant object-oriented database, Report of Department of Multimedia Systems, No RKSM/1/2009, Faculty of Computer Science and Information Technology, West Pomeranian University of Technology, 2009 (in Polish).

[10] Popescu A., Etzioni O., Kautz H.: Towards a theory of natural language interfaces to databases. [In:] "Proceedings of the 8th international conference on Intelligent user interfaces", ACM Press, 2003.

[11] Tyagi S., Vorburger M., McCammon K., Bobzin H.: Core Java Data Objects, Prentice Hall, 2003.

[12] ObjectDB, http://www.objectdb.com/ (29.04.2010).

otrzymano / received: 17.07.2010
przyjęto do druku / accepted: 01.09.2010 *artykuł recenzowany*