An Approach for Designing a Restricted Bulgarian Natural Language Database Query System

Silyan Arsov, Principal Assist. Prof., University of Rousse – Rousse, Department of Computer Systems and Technologies, 7017 Rousse, Bulgaria, sarsov@ecs.ru.acad.bg, http://www.ecs.ru.acad.bg/~SArsov/

Boris Rachev, Professor, PhD, Technical University of Varna, Department of Computer Systems and Technologies, 9010 Varna, Bulgaria, Bob_Ra@acm.org

Abstract

The aim of our researches is to propound a methodology for facilitation of the natural language (NL) access to databases. Besides the different methods used for designing of the pre-processors for processing of the query and its translation into query in a formal database query language, exploration is possible to be implemented on data models purposing (i) more suitable structuring of data in databases, (ii) storing in advance the basic elements of the NL sentences, such as the verbs in the data structures. With regard to this the proposed implementation carries out research on the effect of development of the Entity-Relationship-Attribute (ERA) model, by marking relationships names between different entities as well as between entities and their own attributes. In this connection methods are proposed for data storage that will enable storing both the relationships names between the entities as well as the relationships names between the objects and their own attributes, which are verbs in the most common case. Also we propose a suitable limited set of Bulgarian NL query constructions for the purpose of direct access to data in databases. Finally, for the aim of design, we propose an original generalized architecture of database management system (DBMS) with NL queries. For testing the qualities of the advanced methods by using the suggested architecture we have developed an experimental DBMS with Bulgarian NL queries.

1. Introduction

Different architectures of database systems with NL queries are possible depending on the methods used for query processing and data storing.

In syntax-based systems as Lunar [10] the user's question is parsed, and the resulting parse tree is directly mapped to an expression in some database query language (e. g. SQL).

In semantic grammar systems as Delphi [3], EUFID [8], and PLANES [9] the question-answering is still done by parsing the input and mapping the parse tree to a database query. The difference in this case, is that the grammars do not necessarily correspond to syntactic concepts. There exist systems, as that described in [5], which combine semantically retrieval with other specific methods. Semantic grammars contain hard-wired knowledge about a specific knowledge domain. A new semantic grammar has to be written whenever the NL interface is configured for a new knowledge domain.

Most current systems for NL interface to database use intermediate representation languages. In such architecture the NL interface system first transforms the NL question into an intermediate logical query, expressed in some internal meaning representation language. The intermediate logical query expresses the meaning of the user's question in terms of high-level world concepts which are independent of the database structure. The logical query is then translated to an expression in the database's query language, and evaluated against the database. Lots of NL front-ends as SQUIRREL [2], DATALOG [6], TEAM [7], EXACT [11] use several intermediate meaning representation languages, not just one.

The specified methods are used for access to relational databases, in which the semantics of the relationships in the database is not stored, that makes them inconvenient for direct access when using queries in a NL.

In the paper, for the purpose of a response generation from databases by queries in a restricted NL, a different approach is proposed from the known ones thereby. A further developed Entity-Relationship-Attribute data model is used. In defence of this approach we can indicate [1], where the data model is depicted as a heart of the interface design. The exploration work and the design of EXODUS system described in [4] are also proof of the method proposed in the paper. An appropriate internal representation of the database is used. A generalized architecture of database management system (DBMS) with restricted Bulgarian NL queries is propounded for the aim of the system design.

2. Model Entity-Relationship-Attribute

An extended ERA model is proposed for the description of the real world. A conceptual diagram of an exemplary database described by means of the developed model ERA is shown in Fig. 1.

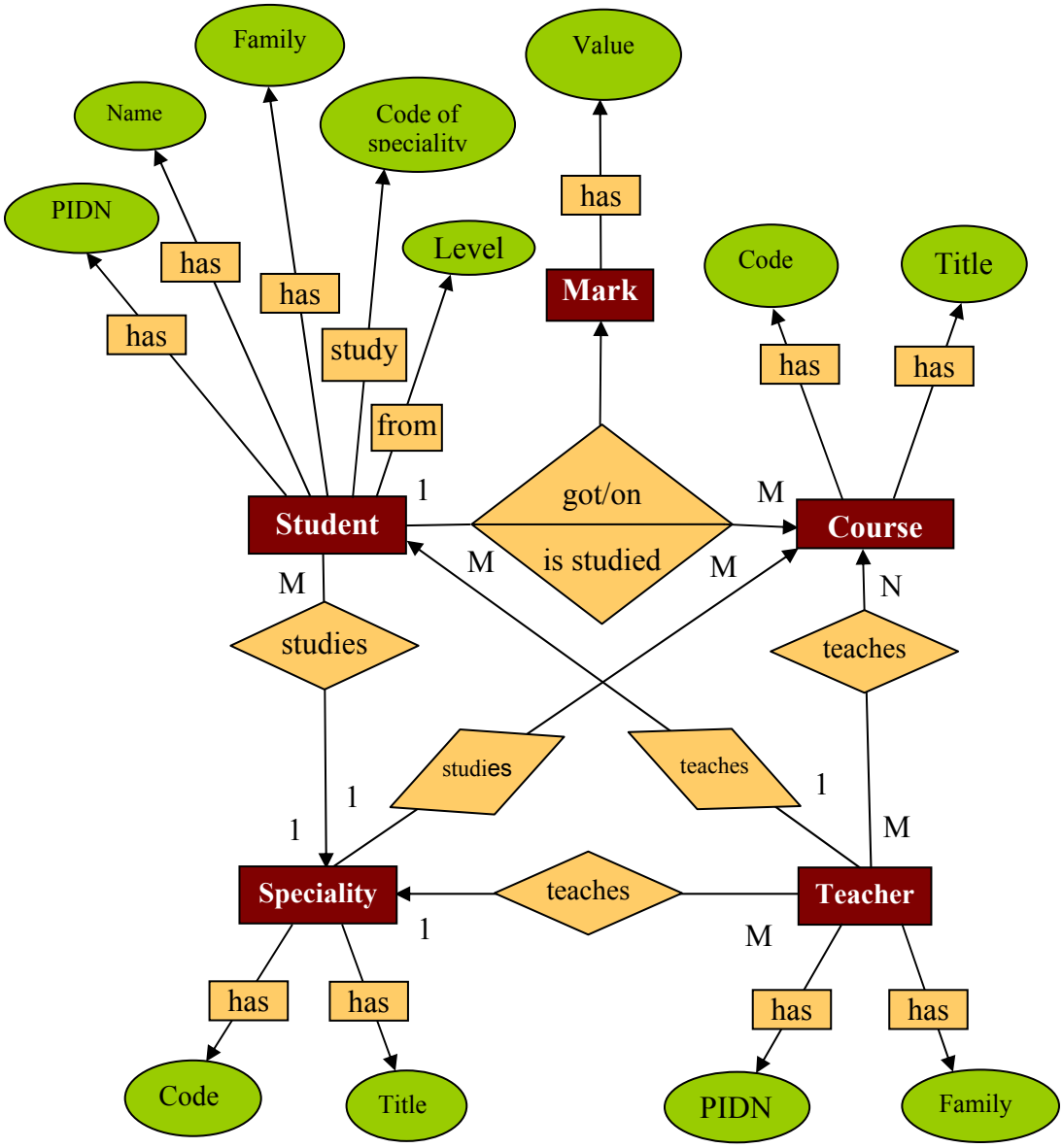

Fig. 1. Entity-Relationship-Attribute diagram of the database

The sets of entities are depicted by means of rectangles in which their names are inscribed. Attributes are presented by elliptic ovals, and the name of each attribute is inscribed in them. The relationships between the set of entities are depicted by means of rhombuses related to the respective entities by an

arrow. The relationships are described by their semantics (name), type. The relationships between the attributes inside the entities are also described by their semantics (name), type, the name of the entity and the name of each entity's attribute. This new unconventional element in the ERA model is added in order to enable the user to obtain direct access to all data elements by queries in a restricted NL. For research purposes the exemplary set of entities *Student, Teacher, Specialty, Course and Mark* is used, as shown in Fig. 1.

3. A method for three-dimensional internal representation of the database

$e_{1,p}$	$V_{1,1,p}$	$V_{1,2,p}$	...	$V_{1,k,p}$	...	$V_{1,n,p}$
	...	...	...	...	...	...
$e_{1,k}$	$V_{1,1,k}$	$V_{1,2,k}$	...	$V_{1,k,k}$	...	$V_{1,n,k,}$
	...	...	...	...	...	...
$e_{1,2}$	$V_{1,1,2}$	$V_{1,2,2}$	...	$V_{1,k,2}$	...	$V_{1,n,2,}$
$e_{1,1}$	$V_{1,1,1}$	$V_{1,2,1}$	...	$V_{1,k,1}$	...	$V_{1,n,1,}$
E_1	$A_{1,1}$	$A_{1,2}$	...	$A_{1,k}$	...	$A_{1,n}$
	....	...	...	...	...	...
	...	...	...	...	...	...
$e_{m,p}$	$V_{m,1,p}$	$V_{m,2,p}$	...	$V_{m,k,p}$	...	$V_{m,n,p}$
	...	...	...	...	...	...
$e_{m,k}$	$V_{m,1,k}$	$V_{m,2,k}$	...	$V_{m,k,k}$	...	$V_{m,n,k,}$
	...	...	...	...	...	...
$e_{m,2}$	$V_{m,1,2}$	$V_{m,2,2}$	...	$V_{m,k,2}$	...	$V_{m,n,2,}$
$e_{m,1}$	$V_{m,1,1}$	$V_{m,2,1}$	...	$V_{m,k,1}$	...	$V_{m,n,1}$
E_m	$A_{m,1}$	$A_{m,2}$	...	$A_{m,k}$	...	$A_{m,n}$

Fig. 2. A three-dimensional representation of the database structure and contents

Each database is represented as a vector, whose components are entities of the real world. A vector, whose components are the attributes, is related with each corresponding owner component of the entities vector. A vector, whose components are the values, is related with corresponding owner component of the attributes' vector. Thus a three-dimensional matrix for representing of the database structure and contents is formed, as shown in Fig. 2. The following denotations are used in Fig. 2: E_i, (i = 1..m) - name of i^{th} entity; $A_{i,j}$, (i = 1..m, j = 1..n) - name of j^{th} attribute of i^{th} entity; $V_{i,j,l,}$ (i = 1..m, j = 1..n, l = 1..p) - l^{th} value of j^{th} attribute of i^{th} entity.

The relationships names are positioned in an additional vector R, R = $||R_1, R_2,.., R_k,.., R_t||$. Additional matrixes are used to represent the relationships and the names of the relationships, both between the entities and the attributes as well as between their values. The matrix for the relationships positioning is bi-dimensional. The entities are positioned in the first row and the first column of this matrix and the names of the relationships between them are positioned in its crossing elements.

The entity - relationships diagram is given in a matrix form in Fig. 3. E_i, (i=1..n) - name of i^{th} database entity; $R_{i,j}$, (i=1..n, j=1..n) - name of the relationship between i^{th} entity and j^{th} entity of the database. It is possible to represent a concrete relationship by vectors, in which the related attributes, the names of the relationships and their degrees participate as components. A values vector is related to each attribute and contains the values of the corresponding attribute. An example is given in Fig. 4. The following denotations are used in Fig. 4: $A_{i,j}$, (i = 1..m, j = 1..n) - name of j^{th} attribute of i^{th} entity; $V_{i,j,l,}$(i = 1..m, j = 1..n, l = 1..p) - l^{th} value of j^{th} attribute of i^{th} entity; $R_{i,j,}$(i=1..n, j=1..n) -name of i^{th} relationship from the vector of relationships; $RD_{i,j}$, (1:1, 1:M, M:1, M:N)(i=1..n, j=1..m) - degree of the relationship between attributes.

	E_1	E_2	...	E_κ	...	E_n
E_1	*	$R_{1,2}$		$R_{1,k}$	...	$R_{1,n}$
E_2	$R_{2,1}$	*	...	$R_{2,k}$	...	$R_{2,n}$
...	...	...	...	...	...	...
E_k	$R_{k,1}$	$R_{k,2}$	...	*	...	$R_{k,n}$
...	...	...	...	...	...	...
E_n	$R_{n,1}$	$R_{n,2}$		$R_{n,k}$	...	*

Fig. 3. A matrix representing description of the relationships between entities

$A_{k,r}$	$V_{k,r,1}$	$V_{k,r,1}$	$V_{k,r,2}$	$V_{k,r,f}$
$A_{k,s}$	$V_{k,s,2}$	$V_{k,s,3}$	$V_{k,s,2}$	$V_{k,s,p}$
$R_{r,s}$	$R_{1,2}$	$R_{1,3}$	$R_{2,2}$	$R_{f,p}$
$RD_{r,s}$	$RD_{1,2}$	$RD_{1,3}$	$RD_{2,2}$	$RD_{f,p}$

Fig. 4. A matrix representing description of the relationships between attributes and values

If the databases are represented by the proposed methods, it is possible to compose algorithms and programs to extract the answers, using the NL queries. Methods and instruments for approbation of the proposed ideas are developed for representing the structures of databases by using an object-oriented approach.

4. A generalized architecture of the designed DBMS, with restricted Bulgarian NL queries.

The generalized architecture of DBMS with queries in restricted Bulgarian NL is presented on Fig. 5

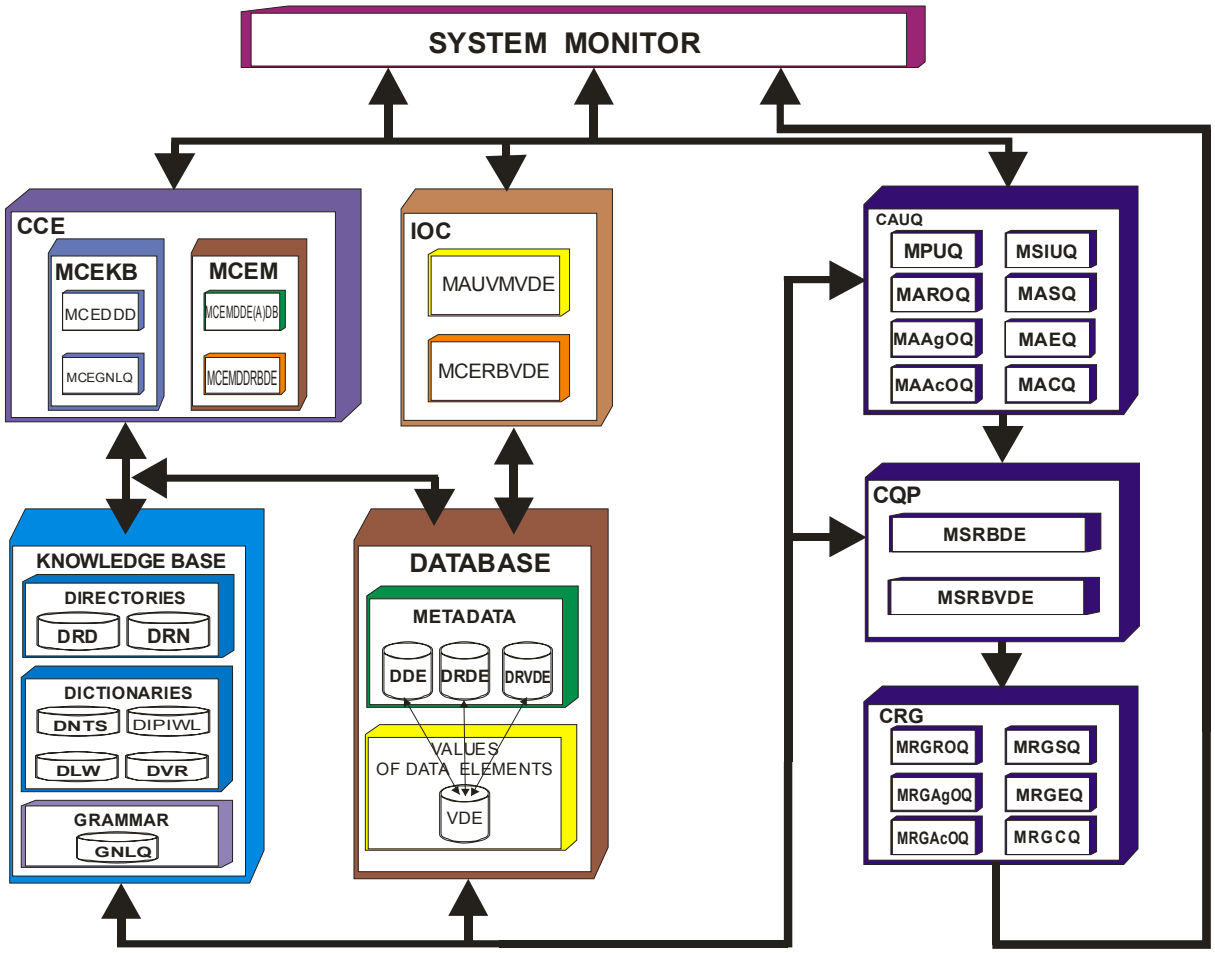

Fig. 5. A generalized architecture of DBMS with Bulgarian natural language queries

The component for creating and editing (CCE) consists of the following two compound modules: the module for creating and editing metadata (MCEM), which consists of the module for creating and editing of metadata, which describes name, type, and size of data elements (attributes) of database (MCEMDDE(A)DB) and the module for creating and editing of metadata, which describes relationships between data elements (MCEMDDRBDE); the module for creating and editing the knowledge base (MCEKB), which consists of module for creating and editing of dictionaries and data directories (MCEDDD) and module for creating and editing the grammar of NL queries (MCEGNLQ), through which are created and edited the restricted NL query constructions.

The input-output component (IOC) consists of a module for adding, updating, visualizing and monitoring the values of data elements (MAUVMVDE) of a database and module for creating and editing of relationships between the values of the data elements (MCERBVDE).
Database contains three types of metadata: metadata for describing data elements (attributes) (DDE), which describes the name, type and size of the attribute, and the decimal places of the numerical attributes; metadata for describing the relationships between data elements (DRBDE), which contains the name of the previous attribute, relationship name, relationship degree and the name of the next attribute; metadata for describing relationships between values of data elements (DRBVDE). The metadata is stored in files with elements of type records and of type one - dimensional or two - dimensional array. The values of data elements (VDE) are stored in file of type records.

The knowledge base contains dictionaries, directories and the grammar: dictionary of nonterminal symbols (DNTS), dictionary of interrogative pronouns and imperative words of the language (DIPIWL); dictionary of logical words (DLW); dictionary of various ratio symbols (DVR); directory of relationship degree (DDR); directory of relationship names (DRN); the grammar of the NL queries (GNLQ). The dictionaries and directories are stored in files. Dictionaries are a set of rules, which defines strings of characters as language elements (for example lexical units, etc.) and assign these elements to a category, which in this case are interrogative words, imperative words, attributes, names of relationships between attributes, various ratio symbols, and logical words. Directories contain knowledge that allows pointing where the stored data are located, how to address them, what their internal representation is and how to access them. Grammar in this case is a set of rules that determinate which of the combination of NL query elements is correct and which is not and identify what the NL query construction type is.

The component for analysis of user queries (CAUQ) controls the user's query, asked on restricted NL and converts it into formal representation, i. e. identifies its known and unknown attributes. The component for analysis of the user queries consists of the following modules: module for parsing of user queries (MPUQ); module for semantic interpreting of user queries (MSIUQ), which divides the attributes, their values, part from query, which they belong to, logical and interrogative words into arrays, which are used as a base for further operations of query processing; module for analysis of relational operations queries (MAROQ); module for analysis of aggregate operations queries (MAAgQ); module for analysis of actualization operations queries (MAAcQ); module for analysis of simple queries (MASQ), which identifies the type of simple query, and according to this type it addresses the module for the formulation of the set of values, which meet the conditions; module for analysis of extended queries (MAEQ); module for analysis of complex queries (MACQ), which after the determination of the type of the constructing simple queries address the module for response generation of simple queries.

The component for query processing (CQP) processes the data, received from the component for analysis of user queries and generates formal response. The component for query processing consists of the following modules: module for searching of relationships between data elements (MSRBDE), which searches relationships between two data elements (attributes), which are stored as metadata for description of relationships between data elements. Module for searching of relationships between values of data elements (MSRBVDE), which forms a set of values, which meet the conditions of the user query.

The component for response generation (CRG) transforms the formal response form, which is received from the component for query processing, in the adequate. The component for response generation consists of the following modules: module for response generation of relational operations queries (MRGROQ); module for response generation of aggregate operations queries (MRGAgQ); module for response generation of actualization operations queries (MRGAcQ); module for response generation of simple queries (MRGSQ); module for response generation of extended queries (MRGEQ); module for response generation of complex queries (MRGCQ).

5. Conclusions

The proposed architectural solution enables non-programming end user to describe and update the data structure and conceptual scheme of databases, and to insert and update the contents of databases.

The components of the described architecture enable users to formulate their queries to databases in restricted NL by choosing the words from the menu and to obtain adequate response.

It also allows updating the knowledge base, which contains knowledge for NL query constructions.

Finally the proposed architecture provides a possibility for system adjustment for working in different knowledge domains, with different constructions of NL queries of a language, and with different kinds of languages.

6. References

[1] Akscyn, R., E. Yoder, and D. McCracken. The Data Model is the Heart of Interface Design. In proceedings of the SIGCHI conference on Human factors in computing systems pages 115-120, Washington, 1988.

[2] Barros, F., and A. De Roeck. Resolving Anaphora in a Portable Natural Language Front End to a Database. In Proceedings of the 4th Conference on Applied Natural Language Processing, Stuttgart, Germany, pages 119-124,1994.

[3] Bates, M., R. Bobrow, R. Ingria, D. Stallard. The Delphi natural language understanding system. In Proceedings of the fourth conference on Applied natural language processing, October, 1994.

[4] Carey, M., D. DeWitt, S. Vandenberg. A Data Model and Query Language for EXODUS. In proceedings of the 1988 ACM SIGMOD international conference on Management of data, Volume 17, Issue 3, July, 1988.

[5] Glockner, I., and A. Knoll. Natural Language Navigation in Multimedia Archives: An Integrated Approach. In Proceedings of International Conference of ACM-Multimedia '99, Orlando, FL, USA, October, 1999.

[6] Hafner, C., K. Godden. Portability of syntax and semantics in DATALOG. ACM Transactions on Information Systems (TOIS), Volume 3, Issue 2, pages 141-164, April, 1985.

[7] Martin, P., D. Appelt, B. Grosz, F. Pereira. TEAM: an experimental transportable natural-language interface. In Proceedings of 1986 on fall joint computer conference, November, 1999

[8] Templeton, M., and J. Burger. Problems in Natural Language Interface to DBMS with Examples from EUFID. In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, pages 3-16, 1983.

[9] Waltz, D.. An English Language Question Answering System for a Large Relational Database. Communications of the ACM, Volume 21, Issue 7, July 1978.

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

[11] Yates, A., O. Etzioni, D. Weld. A Reliable Natural Language Interface to Household Appliances. In Proceedings of the Conference on IUI'03, Miami, Florida, USA, January 12-15, 2003.