International Conference on Computer Systems and Technologies - *CompSysTech'2003*

An Approach for Design of the Object-Oriented Databases with Restricted Natural Language Access

Silyan Arsov, Boris Rachev

Abstract: *The paper proposes a three-dimensional matrix model for representation of entities, attributes, relationships and attribute values of databases in order to access them through menus with predefined fields when posing queries.*
It is discussed that the databases, described by the proposed model can be synonymously defined by the structures, used in the modern object-oriented programming languages, such as classes, objects and dynamical arrays.
Key words: *Object-Oriented Databases, Natural Language Access, Query Language, Data Model, Entity, Attribute.*

1. INTRODUCTION

In most natural language interfaces to databases (NLIDB) the underlying database is assumed to be relational [6]. Some NLIDB support multiple underlying, possibly heterogeneous, databases [1, 8]. Research systems often assume idiosyncratic database models, especially designed to facilitate the development of the NLIDB [9]. When constructing a new NLIDB, one design issue is the interaction between the NLIDB software and the database management system's (DBMS) query language. A NLIDB may translate the question from natural language into Structured Query Language (SQL) instructions. User questions are first translated into a logic language and subsequently into SQL [6]. Research which is conducted on methods and algorithms for the purpose of realizing access to databases using data models different from the relational one is useful too [4]. Currently, there are two major developments in database technology that will have an impact on NLIDB's. The first is the growing importance of object-oriented database systems [2, 3, 5] , and the second is the trend in relational database technology towards more complex storage structures to facilitate advanced data modelling [7].

Besides the different methods used for designing of the preprocessors for processing of the query and its translation into query in a known database query language as SQL, research 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 natural language sentences, such as the verbs in the data structures.

The proposed implementation carry out research on the effect of development of the data model "Entity-relationships", by marking relationships names between attributes of different entities as well as between attributes inside in the entity. The experimental DBMS is implemented using the programme language Visual C++. The object-oriented approach gives new opportunities for continuing of the specified research.

2. PROBLEM FORMULATION

Database information is recorded in terms of files, records, and fields, while natural-language expressions refer to the same information in terms of entities and relationships in the world. A major problem in constructing a natural- language interface is determining how to encode and use the information needed to bridge these two views.

The purpose of the development is to propose data models and methods for data storage that will enable storing both the relationships between the entities as well as the relationships names, that are verbs in the most common case. The further purpose is to extract responses of the queries, formulated in the restricted natural language and to develop experimental DBMS for testing of the qualities of the proposed methods by using the stored relationships names.

3. SOLUTION OF THE PROBLEM

3.1. A method for three-dimensional representation of the database structure and the relationships between the entities and the attributes

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 the figure 1.

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

Figure 1. A three-dimensional representation of the database structure and contents

The following denotations are used in figure 1:

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 figure 2.

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, their degrees participate as components. A values vector is related to each attribute and contents the values of the corresponding attribute. An example is given in figure 3.

Following denotations are used in figure 3:

$A_{i,j}$, (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	...	$E_к$	...	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	$R_{n,1}$	$R_{n,2}$		$R_{n,k}$	...	*

Figure 2. A matrix representing description of the relationships between entities

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

Figure 3. 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 natural language queries. Methods and instruments for approbation of the proposed ideas are developed for representing the structures of databases by using an object-oriented approach.

3.2. Methods and instruments for representing the structures of databases by using an object-oriented approach

The object-oriented data model is based on the following basic concepts. In the data model, a real-world entity is represented as an object, which consists of methods and attributes. While methods, which are procedures and functions associated with the object, define the reactions of the object in response to messages from other objects, attributes represent the inner state of the object. Objects sharing the same methods and attributes are grouped into classes[5].

Class CEntity is defined in our development for description of the entities in the databases. The class structure for description of the entities is shown in figure 4.

The following denotations are used in figure 4:

E_i - name of i^{th} entity in 1-dimensional array of entities;

$A_{i,j}$ -name of j^{th} attribute of i^{th} entity in 2-dimensional array of attributes of class E_i;

As it is seen in the scheme each entity consists of a name and an array, in which its attributes are stored. This array is given size dynamically and each its element is a pointer to an object of a class defining attributes - CAttribute. As it was described in point 3.1 all entities of the databases are stored in an 1-dimensional array.

CAttribute is a class for defining the structure of attributes in databases.

The CAttribute class structure is given in figure 5:

The following denotations are used in figure 5:

Name - name of an attribute;

Type - type of an attribute;

Size - length of an attribute;

Decimals - length of the decimal part of an attribute;

Key(Yes/No) – a flag pointing if an attribute is a key or not;

Inheritor of CEntity - points which entity is inherited by the attribute;

Array of Values (Vi,j,l) - an array of values of the defined attribute.

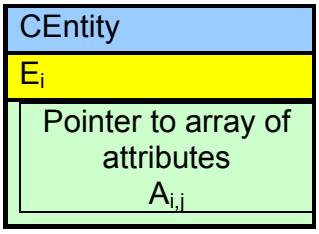

Figure.4. Class CEntity scheme

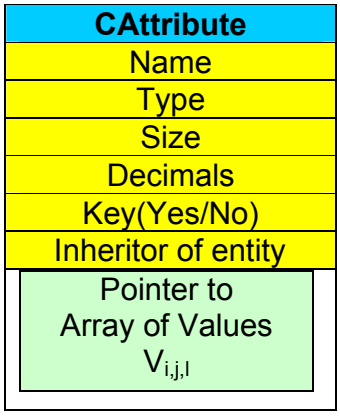

Figure 5. The class defining the attribute structure

Each object of the class CAttribute contains the structure of attributes name, type, size, decimal part and information if it is a key for a related entity. Besides these fields there is another field in the attribute "inheritor of ", which contains the entity name to which the given attribute belongs. Through the mentioned fields the attribute is identified synonymously and can be used for storing the data in databases. There is a field in the class CAttribute, which is an array that is given size dynamically and the values of the attribute are stored in it.

After defining the classes, which describe the entities and their attributes, it is seen that the database is stored as an array of entities and each entity contains an array of attributes and each attribute contains an array, in which its values are stored and these values are inheritors of the properties of the attribute. In this way the 3-dimensional matrix in figure 1 is defined by using the structures in the object-oriented programme language Visual C++. In this way each value is synonymously defined by the entity name, the attribute name and the value number in array of attribute values.

CRelationship is a class for defining of relationships. This class represents relationships between entities and attributes in databases. It contains the attributes names which participate in the relation, the relationship which relates them, the relationship degree and an array which contains the relations between the values of the two related attributes. The class of relationships inherits characteristics of the entities, attributes, and values participating in them. Class CRelationship structure is given in figure 6. The following denotations are used in figure 6:

Name of $Attribute_{k,i,j,}$ (k=1..3, i=1..m, j=1..n) - name of k^{th} attribute participating in the relationship, belonging to i^{th} entity with position in the array of attributes j.

Counter - number of the participants in the relationship;

Name of $Relationship_{1,i}$ - name of the relationship in the relation.

After representing the basic classes in the DBMS, it is necessary to remark that in itself database consists of objects of the defined classes CEntity, CAttribute, and CRelation.

The concept of class allows object-oriented database systems to model complex data more precisely and conveniently than the relational data model.

The represented model is more convenient to access to database by natural language queries, by developing suitable algorithms for searching in the object- oriented databases.

3.3. Methods and instruments for a graphical specification of the database conceptual schemes.

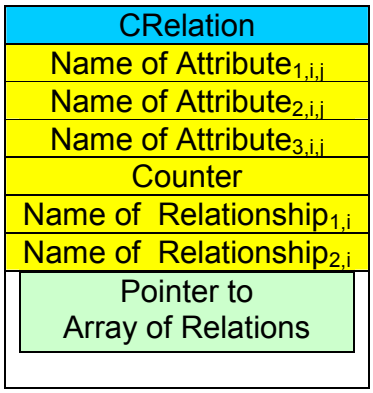

Figure 6. Class defining the relationship

Defining of entities, attributes, relationship names and database conceptual schemes, data adding and updating, as well user query formulation to database is provided by instruments of the implemented experimental DBMS. The end user manipulated by a screen of a graphical-text programme instrumental environment for describing the database conceptual scheme by using the proposed methods for representing the data and the object-oriented approach for database designing.

The first screen is for defining the database entities. It contains a field for entering the entity names and a field for the list of entities in the database.

The second screen is for defining the attributes. It contains fields for input of attribute names, their types, their lengths and the number of signs after the decimal point, and notifying if it is a key attribute as well.

The table with defined attributes is positioned below them and it is possible to select a row from the list using the mouse in order to delete or update.

The third screen is for forming a dictionary with relationship names. It contains a field for entering the relationship names. A list of the already input relationship names is output below this field.

The fourth screen is for defining the database conceptual schemes. it contains four lists in its upper part with the entity names, attribute names, relationship names, relationship degree.

The combination of relationship values is registered by the button "Add" in the lower part of the screen, which are selected from the upper window. A relationship is deleted by the button "Delete". The input row with last input relation is cleared by the button "Clear".

Last defined relations are written in the physical database by pressing the button "OK".

Once the data structures, relationship names and database conceptual schemes have been already defined, it is possible to begin entering and updating of data values in a database.

User query in restricted natural Bulgarian language for output of responses from database is formulated by selecting from menus, from special screen.

The proposed experimental DBMS is implemented on programming language Visual C++ in operational environment WINDOWS'2000. Experimental DBMS is designed for work with data and queries in restricted natural Bulgarian language.

4. CONCLUSIONS AND FUTURE WORK

The most known systems, which convert natural language query in internal representation, use this representation for query structuring in different known database query languages.

The proposed version of object-oriented database with natural language access gives the opportunity to combine, for the purpose of improving dialogue organization, the suitable data models with the suitable semantic query models. It gives premises for the optimization of systems structures of this kind.

The proposed methods are useful in databases represented by models different from the relational one.

5. REFERENCES

[1] Bobrow, R., P. Resnik, and R. Weischedel. Multiple Underlying Systems: Translating User Requests into Programs to Produce Answers. In Proceedings of the 28th Annual Meeting of ACL, Pittsburgh, Pennsylvania, pages 227-234, 1990.

[2] Kemper, A., G. Moerkotte. Advanced Query Processing in Object Bases Using Access Support Relations. Proceedings of 16th International Conference on Very Large Data Bases, pages 290-301, Brisbane, Australia, August, 1990.

[3] Lee, W., D. Lee. Path Dictionary: A New Access Method for Query Processing in Object - Oriented Databases. IEEE Transaction on Knowledge and Data Engineering, Vol. 10, No. 3. May/June 1998

[4] Pottinger, R., A. Levy. A Scalable Agorithm for Answering Queries Using Views. In Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egipt, pages 591-594, 2000.

[5] Rishe, N., S.-C. Chen, A. Vaschillo, J. Yuan, X. Lu, A. Shaposhnikov, R. Athauda, X. Ma, D. Vasilevsky. "Semantic Access: Semantic Interface for Querying Databases". In Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egipt, pages 591-594, 2000.

[6] Roeck, A., H. Jowsey, B. Lowden, R. Turner, and B. Walls. A Natural Language Front End to Relational Systems Based on Formal Semantics. In Proceedings of InfoJapan '90, Tokyo, Japan, 1990

[7] Stohr, T., H. Martens, and E. Rahm. Multi-Dimensional Database Allocation for Parallel Data Warehouses. In Proceedings of the 26th International Conference on Very Large Databases, Cairo, Egipt, pages 273-284, 2000.

[8] Thompson, B., F. Thompson. Introducing ASK, A Simple Knowledgeable System. In Proceedings of the 1st Conference on Applied Natural Language Processing, Santa Monica, California, pages 17-24, 1983.

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

6. ABOUT THE AUTHORS

Assist. Prof. Silyan Arsov, Department of Computer Systems and Technologies, University of Rousse,
Phone: +359 82 888 276, E-mail: sarsov@ecs.ru.acad.bg

Assoc. Prof. Boris Rachev, PhD, Department of Computer Systems and Technologies, Technical University of Varna
Phone: +359 52 302431 (407), E-mail: rachev@ieee.bg