MTA SZTAKI Közlemények/39 p. 5-26.

DEFINING A CONNECTION FUNCTION AS A BASE FOR A USER INTERFACE TO A RELATIONAL DATABASE

Zh. S. ANGELOV

"Interprograma"
P.o.b. 795
1000 Sofia, Bulgaria

Introduction

The wide proliferation of computers in information servicing puts forward the problem for the development of user interfaces (UI), i.e. the methods for the mapping of queries, formulated in terms of the problem area into queries, formulated in terms of the system storing and maintaining data. Such a problem arises in the development of information systems based on DBMS. The DBMS's used nowadays support data models, which offer simple structures for the modelling of the real world and are oriented to easier physical representation. In order to avoid redundancy the designer cannot capture in the schema all known semantic relationships from the problem area when modelling with the help of these structures. The end-user, who is not a specialist in data processing but is a specialist in the problem area and knows these relationships will probably use them formulating his queries. A problem area with a given semantics which is known to any end-user can often be modelled in different ways, for example depending on the frequency of

queries to DBMS. Therefore we need a way for mapping the user view about the part of the modelled area into operations over the structures of the stored data. One possibility is to define a connection function. This function is a mapping of the database state into relations over sets of the problem area terms. We can say that the connection function has as its values the usual user views. But user views in DBMS must be defined in advance for any set of attributes, while the connection function, once defined, gives us a way to compute these views for any subset of problem area terms.

This paper discusses the assumptions under which a connection function can be defined. Its properties and the way for its definition are also discussed. An algorithm, based on the connection function for the development of UI, is proposed.

Assumptions

The proposed approch for building an end-user interface assumes that the attributes carry the whole semantics of the problem area and that the relational schemes are constructed taking into consideration information processes and are not uniquely determined by the diven problem area. Hence, from user point of view the data describing the problem area are stored in a single relation over the set of all attributes. This is the so called universal relation (UR).

In order to ensure the adequacy of the idea of UR existence and the possibility of its automatical maintainance the database schema should possess some properties. For this reason

In the works adopting UR approch some assumptions about the database scheme are made [Ang85, Ang86b, FMU82, MRW83, MUV84, Men84, Osb79, Roz83, Sag83, Ull83].

The first basic assumption, which all authors accept, is:

Assumption 1. 'Universal Relation Scheme Assumption' (URSA). Any attribute in U corresponds to the same class of entities wherever it appears.

Most of the authors understand under this assumption the fact that any attribute plays only one role. For example, NUMBER cannot refer to the number of children and to the department number of an employee.

The next assumption discusses the connection among the given set of attributes $X \subset U$. In order to be able to bind together relations automatically, i.e. to bind together different attributes there should be set up a basic semantic relationship in the scheme. The user should have in mind the same relationship when thinking about the attributes of the given problem area.

Assumption 2. 'Relationship Uniqueness Assumption' (RUA). Let $X \subset U$. There exists only one basic semantical relationship among the attributes X. The user means this unique relationship (denoted [X]) when talking about the attributes X as a whole.

The relationship between the attributes in X-{TEACHER,STUDENT} is an example of the uniqueness of the basic semantic relationship. Speaking about a teacher and a

student together, first of all we have in mind that "The teacher teaches the student".

Let us consider an example taken from [MU83] and fairly often used in cases when the adequacy of one or another formalism, describing semantic relationship among attributes is discussed [An986a,D'AMS83, MRW83, MU83, Roz83].

Example 1. Let us consider a banking database. The attributes are BNK (bank), ACC (account), L (loan), C (customer), AMT (loan amount), BAL (account balance) and ADR (customer address), i.e. $U=\{BNK,ACC,L,C,AMT,BAL,ADR\}$. Let the database scheme be $D=\{R_1,R_2,\ldots,R_7\}$, where the relational schemes are respectively $R_1=\{C,L\}$, $R_2=\{C,ACC\}$, $R_3=\{C,ADR\}$, $R_4=\{AMT,L\}$, $R_5=\{L,BNK\}$, $R_6=\{BNK,ACC\}$ и $R_7=\{ACC,BAL\}$. The hypergraph of the database scheme is depicted on fig.1. □

If $X_1=\{C,ACC\}$, then $[X_1]$ means "The customers own accounts". If $X_2=\{C,ACC,BNK\}$, then $[X_2]$ means "The customers own accounts at the banks". Similarly for $X_3=\{C,L,BNK\}$, $[X_3]$ means "The customers have taken out loans from the banks". If we consider $X_4=\{C,BNK\}$, then under $[X_4]$ we must understand "The customers are served at the banks" or "The banks serve the customers". Finally, if $X_5=\{ACC,L\}$, then $[X_5]=\varnothing$, because there is no basic semantic relationship (The relationship is neither "the loans and accounts at one and the same bank", nor "the loans and accounts of one and the same customer").

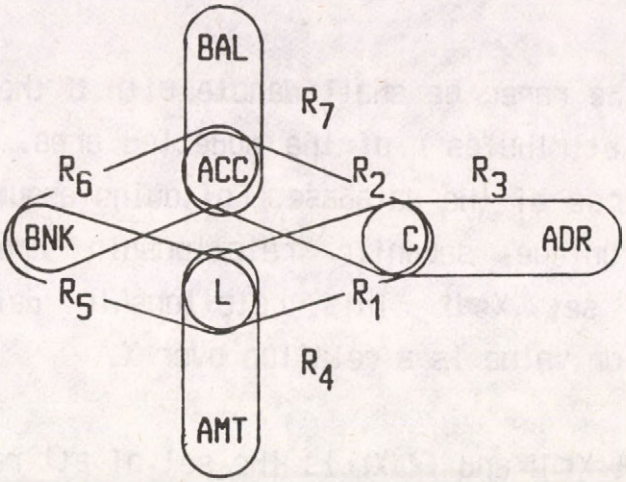

Fig. 1. The banking database scheme

Let us consider again the set X-{C,BNK}. If we connect the attributes C and BNK (see fig.1.) the access path can pass through the attribute ACC (i.e. the customers own accounts at the given banks, so they are served by them) or pass through the attribute L (i.e. the customers have taken out loans from the given banks, so they are again served by them). In this case two access paths are possible but both have the same 'flavor' (the customers are served by the bahks).

The possibility to connect attributes through more then one path arises when the database scheme is cyclic as the one in example 1. In this case the database scheme must satisfy the following assumption:

Assumption 3. 'One Flavor Assumption' (OFA). All access paths used to compute the connection on X represent the same 'flavor' of the relationship among the attributes in X.

Connection function and its properties

In the rest of the paper we shall denote with U the set of all terms (called attributes) of the modelled area. This is the so called **universe** of the database. Following assumption 2 there exists an unique semantic relationship among the attributes of the set $X \subset U$. This relationship defines a function. The function value is a relation over X.

Definition 1. Let $X \subset U$ and $\mathbf{R}(X)$ is the set of all relations over X. A function $[X](d)$ is a **connection function** if it maps the database state d to the set $\mathbf{R}(X)$. (We will omit (d) and denote only with [X] the value of the connection function. The functional [.] maps a subset of U to a function from database states to relations over that subset.)

The connection function is named also simply "connection" [FMU82] and "window function" or "window" [MRW83].

The use of the connection function has some effects which lead to the requirement that the connection function should possess certain properties. Such a property is the satisfaction of the containment condition.

Definition 2. [MRW83] The connection function [.] satisfies the **containment condition** if the inclusion $X \subset Y \subset U$ implies $\pi_X[Y] \subset [X]$.

In other words the following principle must be reflected in

the connection function: When one speaks in general (considering only a few attributes) one refers to more objects than when specifying more details (i.e. when one is interested in more attributes).

Let us consider the database from example 1. The containment condition implies the fact: "The customers served at the banks as a whole" are not less than "the customers who own accounts at the banks" (i.e. $\pi_{\{C,BNK\}}[\{C,L,BNK\}] \subset [\{C,BNK\}]$).

When asking for some data the end-user expects to retrieve the data which has been inserted, i.e. if the user has added a tuple t to a stored relation $r(R)$ then the value of the connection function for the set R must contain such a tuple t. In other words the connection function must provide visibility of all tuples stored in the database. This property is called "faithfulness".

Definition 3. [MRW83] The connection function is **faithful** (possesses the property **faithfulness**) if $r(R) = [R]$ holds for any relational scheme $R \subset U$ and any database state d.

This means that the connection function must neither hide tuples nor add any. Therefore we have to explicitly store in the database all known facts. Taking into consideration the trend to introduce deductive capabilities in DBMS, a more realistic definition of this property may be $r(R) \subset [R]$.

Approaches to the definition of the connection function

All known methods for the definition of a connection function include joins of the stored relations. Relations are the minimal objects which can be updated. Therefore the relational schemes can be called **update structures.** These are the base for the conection function definition. Following the fact that a "good" connection function must satisfy the containment condition and is a monotonously decreasing function we can construct for any definition method a special kind of structures - **retrieval structures.** These structures are sets of attributes.

Definition 4. The set of attributes $X \subset U$ is a retrieval structure if there exists a database state such that the connection function value is not the empty set ($[X] \neq \varnothing$) and the extension of X with any other attribute $A \in U-X$ does not have this property (i.e. $[X \cup \{A\}] = \varnothing$).

As the retrieval structure coincides with the union of some of the update structures we can consider the retrieval structures as sets of update structures.

The retrieval structure semantically corresponds to one of the possible aspects of the basic semantic relationship. Thus the definition of the connection function can be formulated as follows:

Definition 5. Let $V=\{R_1, R_2, \ldots, R_n\}$ is the set of the update

structures and $W=\{S_1, S_2, \ldots, S_m\}$ is the set of retrieval structures, where $S_i \subset V$. Then the connection function is

$$[X] = \bigcup_{S \in W, X \subseteq attr(S)} \pi_X \left(\bowtie_{R \in V, R \in S} r(R) \right),$$

where $attr(S) = \bigcup_{R \in S} R$.

Therefore, the way of constucting the retrieval structures is significant for the connection function definition. There exist two approaches of using (i.e. constructing) the retrieval structures:

- the structures are not explicitly given and have to be constructed from the database scheme using some additional information as functional dependencies (for example, building "lossless joins", in particular "extension joins");
- the structures are explicitly given, i.e. a set of new structures is added to the database scheme (for example "maximal objects" in System/U or "objects" in PIQUE).

The nonexplicitly defined retrieval structures are based on one or more classes of data dependencies (mainly functional ones). As a consequence if the modelled area is more complex and its semantics cannot be described using these kinds of dependences, we cannot obtain an adequate connection function definition. This gives us confidence to claim that the explicit definition gives better results.

The explicit definition raises some problems too. It is not absolutely clear how to create the retrieval structures. We can use some algorithms to construct possible retrival structures, but sometimes the obtained structures may be not very good. The

example in [KKFGU84] shows that little changes in the modelled area can imply essential reconstruction in the set of retrieval structures.

These difficulties show that the database scheme and integrity constraints are not sufficient for the adequate definition of retrieval structures which are used in UI. This is a consequence of the loss of some information in the process of the mapping the conceptual model of the problem area to model supported by the DBMS. All proposed methods discuss the creation of retrieval structures after defining the database scheme, i.e. after defining the update structures. The process is depicted on fig.2. Thus the way in which the database scheme

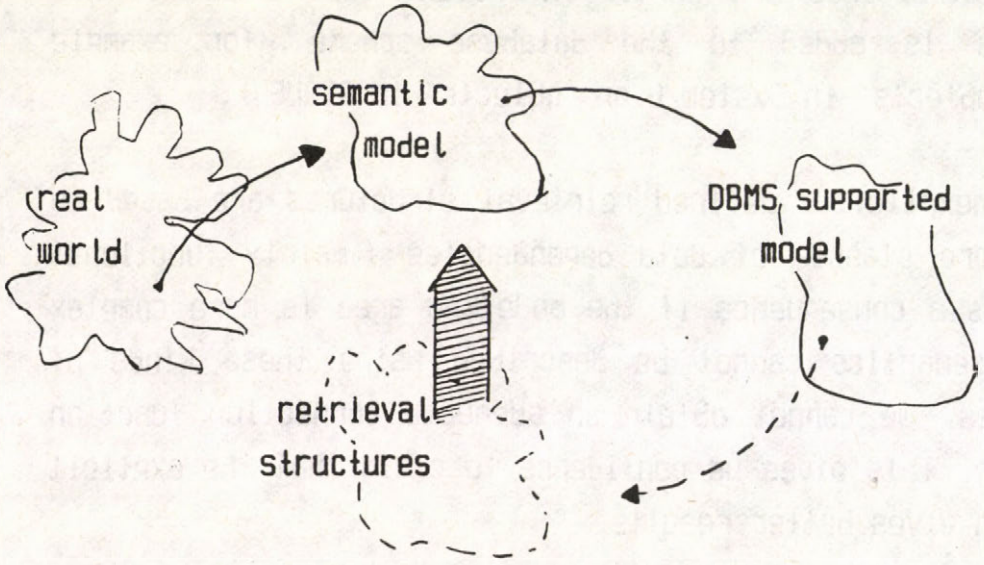

Fig.2. Mapping the models in the database scheme design process

is obtained, is lost. If we (in a semantic or a conceptual

model) define first the retrieval structures and after that map them onto the update structures, then we automatically obtain a definition of a connection function. In this way the last mapping in fig.2. may be removed and we have the chance to capture more meaning. This approach is followed in [Ang86a, Ang86b]. The retrieval structures are constructed using the aggregation hierarchy.

An algorithm for an user interface to a DBMS, supporting an universal relation as an user view

The query language used by the end user has to provide the following two capabilities:

- explanation of the target attributes;
- specification of the conditions which the target data must satisfy.

If the query language is intended to free the user from the logical navigation then it obeys the following principle: **The sentences of that language cannot include any structure related to data storage details.** Instantiating this principle in the relational data model we obtain the assertion that **the query language cannot include any relation names.** Thus a query may consist only of attribute names and the way they are related. In this case the user wants to extract information from a view. This view must include all attributes mentioned in the query and it can be calculated using a connection function. In this way for any query we can juxtapose a value of the connection function for the set of attributes mentioned in that query. The

tuples which contain the information of interest are among the tuples of the connection function. The former can be extracted through testing which of the latter satisfy the conditions in the query.

In almost every known system, based on the universal relation scheme, the query processing is separated into two steps [Ang86b, KKFGU84, MRS85, MRW83, MUV84]:

a) binding (i.e. user view creation). It consists of the construction of the connection function [X] for the set X of attributes mentioned in the query;

b) evaluation (i.e. target data extraction). Whatever operations must be applied to answer the query, are then applied to [X].

The following example illustrates these two staps.

Example 2. Let us consider the database from example 1 and let d be a database state. Let the connection function [X] be defined as a projection of joins of relations with relational schemes covering the set of attributes X, i.e. $[X]\text{-}\pi_X(\bowtie_{R\in M} r(R))$, where $X \subset attr(M)$ and $r(R) \in d$. If the user query is "Find all account balances of the customer named 'Angelov'", then the answer can be contructed in the afore said two steps. The first step is the binding of the attributes mentioned in the query, namely {BAL,C}. In other words, we build an expression whose value gives the value of the connection function for {BAL,C}. According to the above definition the expression is $[\{BAL,C\}]\text{-}\pi_{\{BAL,C\}}(r(R_2) \bowtie r(R_7))$. The second step should reflect the fact that the user would.

like to retrieve information referring to the customer named 'Angelov' only. Therefore we have to use the selection operation over [{ BAL,C }]. Finally, the answer to the user query is the value of the expression

$$\sigma_{C='Angelov'}\pi_{\{ BAL,C \}}(r(R_2) \bowtie r(R_7)) \qquad \square$$

Setting up the construction of a connection as a first step allows us to test easily new ideas for the building of the connection function (as in [Ang86b]). We would like to stress that this gives also the opportunity to use different algorithms depending on the available additional information. Below we will describe a generalized algorithm for an user interface to a RDBMS (without taking into consideration the type of the additional information). This algorithm allows the user to view the data in the database as stored into one unique relation. But firstly let us name some of the sets and relations, which will be used later: MENSET will denote the set of all attributes mentioned in the query, ANSSET - the set of the target attributes and $RETSTR_i$ - the set of relational schemes included in a given retrieval structure. RETSTRSET is the set of all sets $RETSTR_i$ which cover MENSET. The relation of the user view will be denoted by **window** and the answer will be received in the relation **answer**. **window** is a relation over MENSET and **answer** is a relation over ANSSET. Let **cond** be the condition formulated in the given query. As a query language we can assume a modification of SQL according to the principle stated above, i.e. only the use of attribute names is permitted.

Algorithm 1. Construction of the answer to a query to a

system, supporting an universal relation.

Input: An user query specified in "modified" SQL.

Output: The relation **answer** which is the answer to the user query or a message for the "meaninglessness" of the query if the set MENSET is unconnected.

Method:

1. Extract the names of all the attributes mentioned in the user query, thus constructing the set MENSET. Extract the names of the target attributes and construct the set ANSSET. Construct the expression **cond**.

2. Find all retrieval structures $RETSTR_I$ covering the set MENSET (i.e. $MENSET \subset attr(RETSTR_I)$).

3. If there exists no retrieval structure which covers MENSET (i.e. RETSTRSET-$\varnothing$), then output the message for the "meaninglessness" of the query and stop.

4. For each i reduce the retrieval structure $RETSTR_I$. Remove any set $RETSTR_I$ which is a superset for any other set $RETSTR_J$.

5. For each i compute the join of the relations over schemes from $RETSTR_I$. Project the result over MENSET and make its union with the temporary relation **window**.

6. Remove (using selection) all tuples which do not satisfy the condition **cond**.

7. Project the relation **window** over ANSSET in order to obtain the relation **answer**.

Step 1 includes only preliminary procedures. In step 2 the connection function is formed. Here are the main differences among the various approaches. As mentioned above, different algorithms can be used depending on the available information.

Step 3 has a control function. The procedures included in step 4 aim to optimize the creation of the relation **window** (the value of the connection function). Although most of the definitions of the connection function take into consideration some criteria for optimization, this leads to local optimization only. Generaly, it is assumed that the optimization is carried out by the DBMS and not by the interface. The reduction of a retrieval structure aims to remove relations which will not change the result. The information of interest, contained in those relations is included into the remaining relation. In other words, the remaining relations contain more general information about the particular case. As an example let us consider the expression

$$[\{C,L,BNK\}] - \pi_{\{C,L,BNK\}} (r(R_1) \bowtie r(R_3) \bowtie r(R_5) \bowtie r(R_4))$$

(Note that the definition of [.], given in example 2., is used here). The relation $r(R_3)$ contains information for all bank customers. The attribute L in the set {C,L,BNK} shows that we are interested only in the customers which have taken out loans. The absence of the attribute ADR shows that we are not interested in the customers' addresses. Therefore the inclusion of the relation $r(R_3)$ in the evaluation of the expression will be a pure loss of time. The reduction can be performed in different ways. For example, in System/U minimization under weak equivalence is used. A different method is given below, which is a modification of the Graham reduction. The proposed second part of step 4 follows directly from the Join property. Step 5 calculates the value of the connection function. Step 6 and 7 perform the second part of query processing.

It should be pointed out that step 4 is optional. If step 4

is omitted, the result will be the same but in this case we will lose much more resources than are needed for the execution of step 4. That is why we suggest to use Graham reduction algorithm. The original Graham algorithm [Gra80] is intended for testing hypergraph acyclicity. For the purpose of optimization we propose a modified algorithm. The idea is: Consider the attributes mentioned in the query as a goal set and the hypergraph, having as its nodes the attributes and as edges - the relational schemes from this structure. The algorithm removes edges and nodes preserving the goal set and the hypergraph connectivity.

Algorithm 2. Reduction of a set of relational schemes with respect to a goal set of attributes.

Input: A set of relational schemes $\mathbf{R}$ and a goal set of attributes $G \subset attr(\mathbf{R})$.

Output: A set of relational schemes $\mathbf{R}' \subset \mathbf{R}$ which when considered as a hypergraph has the following property: for any two elements of G, there exists a path between them.

Method:

1. Let $\mathbf{R}' - \mathbf{R}$.

2. Apply one of the following operations as many times as possible over the relational schemes in $\mathbf{R}$:

a) If A is not in G and is included in only one relational scheme $R_i \in \mathbf{R}$ then remove A from R_i;

b) If $R_i \in \mathbf{R}$ is a subset of $R_j \in \mathbf{R}$ then remove R_i from the sets $\mathbf{R}'$ and $\mathbf{R}$. □

Conclusion

In the paper an approach to the development of an user interface is considered. It is based on the use of the connection function. It is shown that the problems arising in development of the UI can be separated from the specific model of the problem area which is used as a database scheme. The introduction of the connection function as a base for the UI makes the latter independent from the manner of the real world modeling. This facilitates the transition towards the use of a more complex model if the one in use does not possess enough expressive power. There is a price to be paid for this freedom by the database administrator. He (or she) has to create and maintain a database scheme which satisfies the given conditions. In the paper, such conditions are formulated and discussed. Such an interface may serve as a workbench and can be an useful tool for testing new ideas in the field of the automation of query answering.

References

[Ang85] Angelov Zh.S. : Towards a Universal Relation View. In Proceedings Eighth International Seminar on Database Management Systems, Piestany, Czechoslovakia, 1985, pp. 9-17.

[Ang86a] Angelov Zh.S. : A Connection Function for End-User Interface. In Proceedings XV-th Spring conference of SMB, Sunny beach, Bulgaria, 1986, pp. 350-355. (in bulgarian)

[Ang86b] Angelov Zh.S. : Simulating a Universal Relation View

Introducing Aggregation Information. In Proceedings Ninth International Seminar on Database Management Systems, Reinhardsbrunn, GDR, 1986, pp. 189-194.

[D'AMS83] D'Atri A., Moscarini M., Spyratos N. : Answering queries in relational database. In Proceedings ACM SIGMOD Annial Meeting, San Jose, May 23-26, 1983, pp. 173-177.

[FMU82] Fagin R., Mendelzon A.O., Ullman J.D. : A Simplified Universal Relation Assumption and Its Properties. ACM Trans. Database Syst. Vol.7, No.3, 1982, pp. 343-360.

[Gra80] Graham M.H. : On the Universal Relation. In A Panache of DBMS Ideas III. (ed.) D Tsichritzis. Technical Report CSRG-111, University of Toronto, 1980, pp. 68-92.

[KKFGU84] Korth H.F., Kuper G.M., Fiegenbaum J., Gelder A., Ullman J. : System/U - A Database System based on the Universal Relation Assumption. ACM Trans. Database Syst. Vol.9, No.3, 1984, pp. 331-347.

[MRS85] Maier D., Rozenshtein D., Stein J. : Representing roles in universal scheme interfaces. IEEE Trans. on Software Engineering. Vol.SE-11, No.7, 1985, pp. 644-652.

[MRW83] Maier D., Rozenshtein D., Warren D.S. : Windows on the World. Proceedings ACM SIGMOD International Symposium on Management of data, San Jose, Calif., 1983, pp. 68-78.

[MU83] Maier D., Ullman J.D. : Maximal objects and the Semantics of Universal Relation Databases. ACM Trans. Database

Syst. Vol.8, No.1, 1983, pp.1-14.

[MUV84] Maier D., Ullman J.D., Vardi M.Y. : On the Foundations of the Universal Relation Model. ACM Trans. Database Syst. Vol.9, No.2, 1984, pp. 283-308.

[Men84] Mendelzon A.O. : Database states and their tableaux. ACM Trans. Database Syst. Vol.9, No.2, 1984, pp. 264-282.

[Osb79] Osborn S.L. : Towards a Universal Relation Interface. In Proceedings Conf. on Very Large Data Bases V, 1979, pp. 52-60.

[Roz83] Rozenshtein D. : Query and Role Playing in the Association-Object Data Model. Ph.D. dissertation, SUNY at Stony Brook, 1983.

[Sag83] Sagiv Y. : A Chacterization of Globally Consistent Databases and Their Correct Access Paths. ACM Trans. Database Syst. Vol.8, No.2, 1983, pp. 266-286.

[Ull83] Ullman J.D. : Universal Relation Interfaces for Database Systems. In Information Processing 83, R.E.Mason(ed), Elsevier Science Publishers B.V. (North-Holland), pp. 243-242.

DEFINING A CONNECTION FUNCTION AS A BASE FOR A USER INTERFACE TO A RELATIONAL DATABASE

Zh. S. Angelov

Summary

In the paper an approach to the development of a user interface is considered. It is based on the use of the connection function. The introduction of the connection function as a base for the user interface (UI) makes the latter independent from the way of the real world modelling. There is a price to be paid for this freedom by the database administrator. The administrator has to create and maintain a database scheme which satisfies the given conditions. In the paper such conditions are formulated and discussed. Such an interface may serve as a workbench and can be a useful tool for testing new ideas in the field of the automation of query answering.

KAPCSOLATFÜGGVÉNYEN ALAPULÓ FELHASZNÁLÓI INTERFACE RELÁCIOS ADATBÁZISOKHOZ

Zh. S. Angelov

Összefoglaló

A dolgozat egy kapcsolatfüggvényen alapuló, a valós világ modellezésének módjától független felhasználói interfaceszel foglalkozik. Ezt a függetlenséget természetesen nem adják ingyen: az adatbázis adminisztrátorának létre kell hoznia és karban kell tartania egy bizonyos feltételeknek eleget tevő sémát. A dolgozat megadja, és részletesen megvizsgálja ezeket a feltételeket. Az interface az automatizált lekérdező rendszer területén az uj ötletek kipróbálásának hasznos eszköze lehet.