INFORMATICA, 1998, Vol. 9, No. 1, 51–64

From QBE to Graph Queries

Valery GRECHKO, Peter TULCHINSKY, Vadim TULCHINSKY
Institute of Cybernetics
Acad. Glushkova 40, 252022 Kiev-22, Ukraine
e-mail: vogrec@olgal.pp.kiev.ua

Received: January 1998

Abstract. An extention of the QBE language in the form of graph queries is proposed for databases having complex schemata. These graph queries provide user interface for the ER model, like QBE queries provide user interface for the relational model. The implementation and usage of graph queries of ER-based "MicroPoisk" DBMS are presented.

Key words: DBMS, database, ER-model, user interface, QBE, relational/network enterprise model.

1. Introduction

Despite its ubiquity, the relational data model poorly fits for many types of problems now common in the enterprise (Codd, 1971). In recent years, a growing number of requirements has arisen because application developers are seeking for more flexibility and functionality in the data model. Object/relational, multidimensional, deductive, and active database models are well-known approaches of database capabilities extension. The entity-relationship (ER) model (Chen, 1976) is widely used now for the database design. It can be viewed as a means for the relational/network enterprise specification.

The importance of support of complex databases increases with the spread of data marts and data warehouses. Thus comprehensive tools for support of complex databases and maintenance of intelligent queries to such databases are needed. Most efforts in this field are aimed now at data mining and analysis of the aggregated information (OLAP). The above methods are based on simplifying the schema and decreasing the information dimension down to the acceptable level.

An alternative approach to operate the complex databases is proposed in this paper. It is based on extracting and processing the whole graph of semantically and structurally interconnected information from the ER-described database. "Graph queries" can be understood as a generalization and an alternative to ordinary "tabular queries".

There are three traditional kinds of user-oriented tools for ad-hoc queries formulation and data manipulation for relational databases:

- SQL and SQL-like languages (Informix 4GL etc.). This kind of tools is intended for developers;

- Query by example (QBE) is intended for end-users. This kind of tools provides simple table-based user's interface;
- Query by form (QBF) provides processing of tables through the pre-designed screen forms or dialogs. Generally, it is an analogue of QBE.

A user-oriented QBE/QBF query tool reminiscent is needed for the ER model. This tool should include visual formalism for obvious representation of graph structures on a computer display. It should generalize the relational QBE to complex hierarchical and network enterprise specifications. This tool (ER-QBE) should remain simple to be a means for the end-user.

The solution of the above problem based on *metagraph* and *information graph* (Grechko *et al*, 1995) is proposed in this paper. The solution was implemented in DBMS "MicroPoisk" and tested on a set of different applications.

2. Graph Query and Information Graph

The "MicroPoisk" *graph query* is a special description of the subscheme presented in the form of a graph. This graph description consists of a set of special SQL-queries. Generally it is based on the ER-schema of a complex database and is not limited by entities and relationships described in the schema. The graph query includes SQL-queries for selection, creation, and destruction of entities and relationships and provides both extraction and processing of the corresponding information graph.

The graph query is based on the metagraph. *Metagraph* is a connected oriented graph. Each node of the metagraph corresponds to a class of entities. Each arc corresponds to a class of relationships between the classes of entities corresponding to the nodes connected by this arc. The only node is selected and called the metagraph root. Different nodes of the metagraph may correspond to the same class of entities. In this case, different sets of entities can be represented by them. Any node of the metagraph represents either the whole set of entities of the corresponding class or its subset.

Each node and arc includes SQL-queries with one parameter. The parameter is replaced by the primary key of the corresponding record (entity or relationship) when a graph query is run.

We call *metatree* the tree of paths from the root node in a given metagraph (its transversal tree). Consider the metatree as a structural diagram of a hierarchical model of data. The extension of a virtual database whose schema is set by this structural diagram can be constructed. The extension is a wood of trees. The nodes correspond to entities and the arcs correspond to relationships between appropriate entities. The root of each tree is some entity from a set represented by the metagraph root. Each tree is referred to as a *tree of the virtual database*. We call such a maximum subtree of the tree of the virtual database that does not have an entity fixed to a node and its own descendant a *covering* tree. The wood of covering trees of the virtual database is called an *information graph.*

Both the metatree and information graph can be presented on a computer display as a tree view. This is a significant advantage of the proposed instrument, because generally

accepted visual formalism for computer presentation of an arbitrary graph doesn't exist by now. The graph query can be presented on the base of its metatree. "MicroPoisk" includes a special database called *metabase* that contains graph queries and some utilities for the graph query design and their interpretation as information graphs. Graph queries can also be presented as information graphs using a graph query for the metabase.

The QBE-like toolkit of "MicroPoisk" provides the specification and processing of data stored in the relational/network database. An MPQBE utility implements ER-QBE and proposes visual, end-user oriented tools for design and modification of a graph query. It provides the user with automatic taking into account the dependency among entities and the reference integrity in elementary SQL-queries of the graph query. Utility MPTREE implements the interpretation of the graph query. This means the ability to browse the information graph, to create/destroy entities and relationships, to search for specified entities, etc. Special components provide editing individual attributes of a currently selected entity or a group of entities of the same class.

The user can apply MPTREE in editing the metabase itself. In that way, the graph query and its nodes can be enriched by special queries (methods), connected programs, *visual formalisms* (Harel, 1988) etc. This enriched graph query can be viewed as the specification of a complete database application and so it is called a *graph prototype* of the application. The integral structure of the graph prototype is shown in Fig. 1.

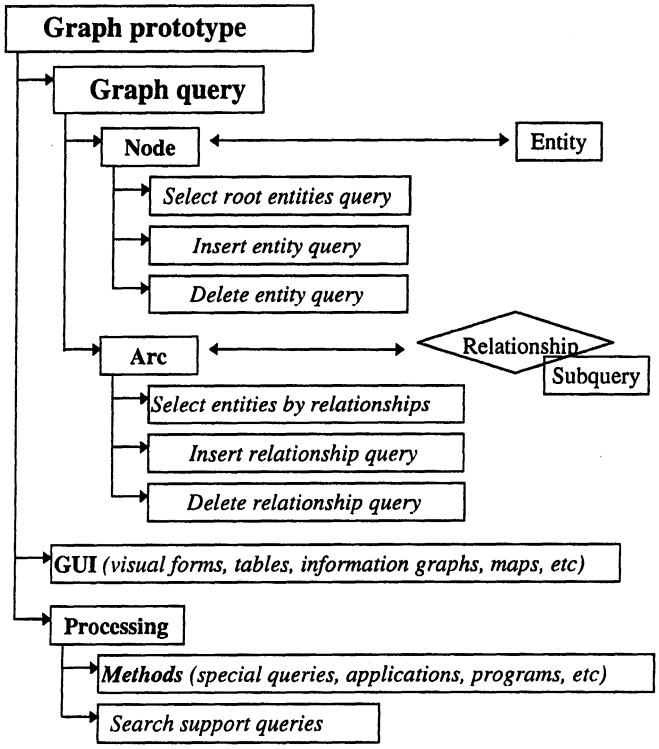

Fig. 1. Graph prototype and graph query.

3. Parallelism between the Relational QBE and ER-QBE

The source template (form) is filled in the relational QBE/QBF, and then the destination table is extracted from the database and automatically processed according to the source form. Unlike the relational QBE, the ER-QBE source template is not a form but an oriented graph of forms. And ER-QBE destination is not a table but an oriented graph of entities connected by relationships.

The visible/editable destination table of the relational QBE/QBF is commonly static. The user can control this table by simple navigation and scrolling only. But the destination information graph of ER-QBE also allows browsing and searching "in depth" and "to expanse", hiding some branches and disclosing all or some levels of descendants, etc.

Both the relational QBE/QBF and ER-QBE provide group and individual record processing. But relational QBE/QBF is intended above all for group processing of the whole destination table, while ER-QBE provides advanced tools for operating single entities and relationships.

Similarly to QBF, ER-QBE allows us to develop ready-to-use database applications (via the graph prototype mechanism).

In comparison to the relational QBE, ER-QBE provides a much more efficient simultaneous review and editing of heterogeneous entities. If an enterprise object is to be described by k dimensions, and there are N_i measures for each i-th dimension, the object can be shown as a QBE destination table of $N_1 * N_2 * \cdots * N_k$ rows, but it can also be shown as an ER-QBE information graph of $N_1 + N_2 + ... + N_k$ rows (see Fig. 2). Thus ER-QBE-like tools can be used as a *star schema*-based editor for multidimensional databases.

Contrary to the relational QBE/QBF, ER-QBE provides visualization and operating of recursive hierarchic and network structures, e.g., road-map, communication or trading network, hierarchy of distributors etc. An example is shown in Fig. 3.

The only SQL-query corresponds to each QBE-template. But ER-QBE creates two or three SQL-queries for each node-form (construction/destruction of entities and selection

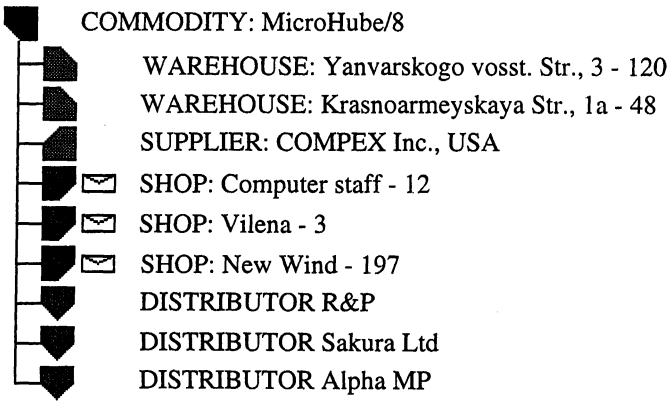

Fig. 2. Description of available commodity consists of several heterogeneous entities.

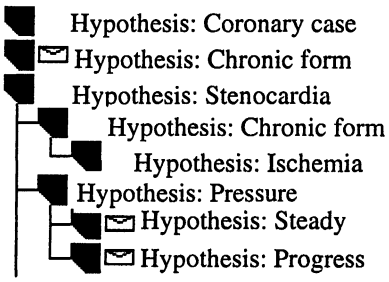

Fig. 3. Fragment of the hypothesis graph of the cardiology anamnesis.

of entities for the root node) and three SQL-queries for each arc (selection of descendant entities, construction and destruction of relationships from parent entity to descendant). And all those SQL-queries are sub-queries of one graph query. They are executed step-by-step during the graph query interpretation controlled by user's actions.

4. Examples of Applying the Graph Queries in Database Representations of Network Enterprises

The telephone communication network is one of well-known graph-based enterprises. Let us examine graph queries application by the example of the "TopoSviaz" application database fragment shown in Fig. 4. This database fragment describes the so-called "initial second network".

The initial secondary telephone network consists of Ukrainian towns (*Center*) connected by telephone channels (*Channel2*) of different carrying capacity. The description of *Center* and *Channel2* entities is shown in Table 1.

4.1. *First Graph Query Example: Sub-schema of the Telephone Network*

As spoken formerly, one can invent a number of different graph queries for any database according to different user's information needs. A graph query in Fig. 5a repeats the

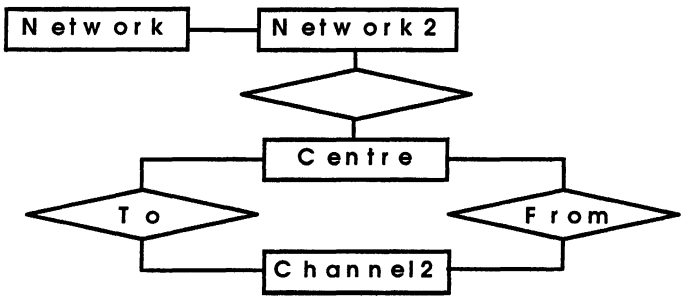

Fig. 4. The database schema fragment corresponding to the initial second telephone network.

sub-schema shown in Fig. 4. A fragment of the information graph selected by MPTREE according to the graph query is shown in Fig. 5b.

Table 1
Detailed description of the example database fragment

Object	Attribute	Description
Center	*Name*	Name of town
	Code	Long-distance telephone code
	Type	e.g., transit, final or a transit-final node
	x, y	geographical coordinates
	...	
Channel2	*SrcCentre*	Long-distance code of town at the begining of the channel
	DstCentre	Long-distance code of town at the end of the channel
	Distance	Physical length of the channel
	Assignment	Calculation value of the channel assignment
	Capacity	Calculation value of the channel carrying capacity
	...	

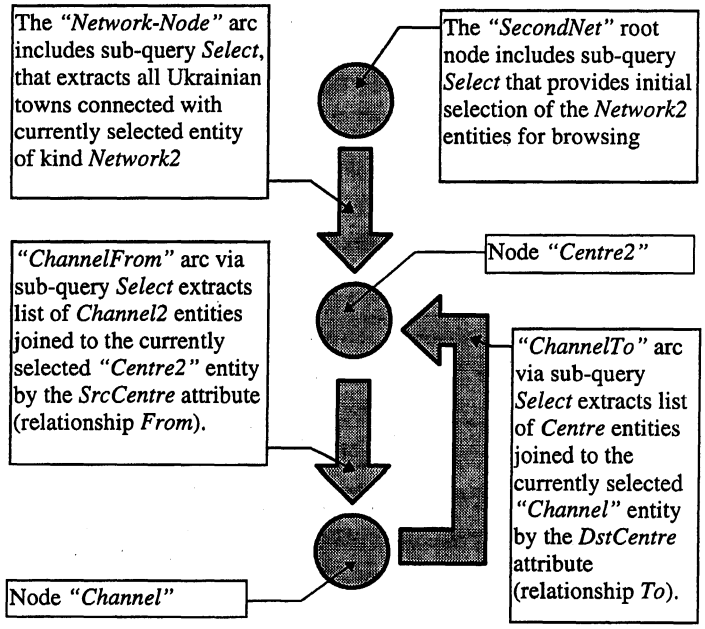

Fig. 5a. Structure of the graph query "Browsing of initial second network".

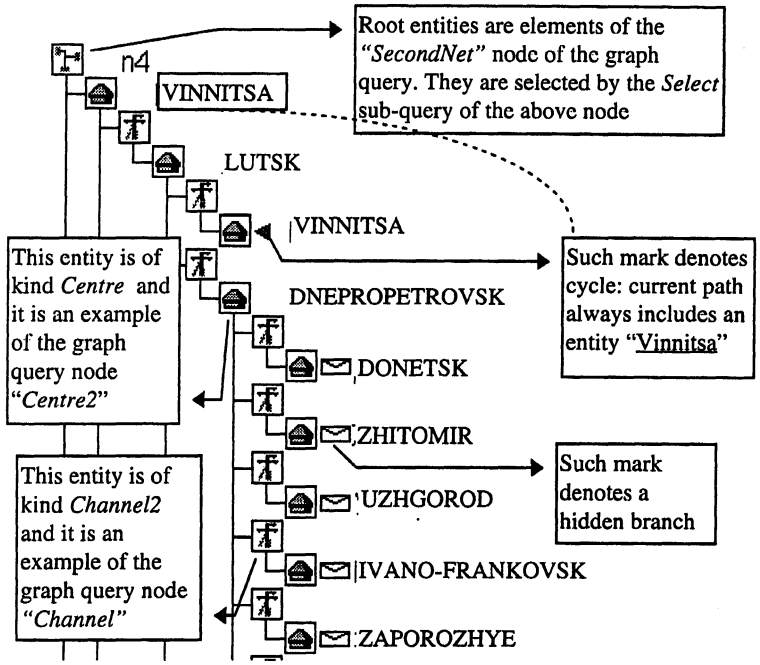

Fig. 5b. Fragment of the information graph corresponding to the graph query "Browsing of initial second network".

4.2. *Second Graph Query Example: Transit and Final Centers*

The same database fragment can be viewed as a list of transit centers supplemented by final centers connected to each of its elements. The corresponding graph query and information graph are shown in Figs. 6a–6b.

In the simplest case, there is the only entity corresponding to each node of the information graph and the only relationship corresponding to each of its arcs. The second example of the graph query includes 'virtual' relationship. Its "Transit" arc consists of two transitions: from the *Center* entity to *Channel2* via the *From* relationship and from *Channel2* entity to *Center* via the *To* relationship. Channels aren't shown in the information graph contrary to the previous example. Commonly one can use 'virtual' entities in the same manner as 'virtual' relationships. Such entities are useful for aggregation of dimensional information, e.g., by years, months, weeks, kinds of wares, etc. But the registration of aggregated entities complicates sub-queries, and its automatic implementation isn't supported in the current version of the "MicroPoisk" ER-QBE tools.

It should be noted that entities and relationships included in the 'virtual' relationship (e.g., channels) are taken into account in all its sub-queries, and if, e.g., one deletes the center, all its channels will be also deleted automatically.

This example also illustrates the use of different nodes corresponding to the same class of entities. Both '*TransitNode*' and '*FinalNode*' represent *Center* entities.

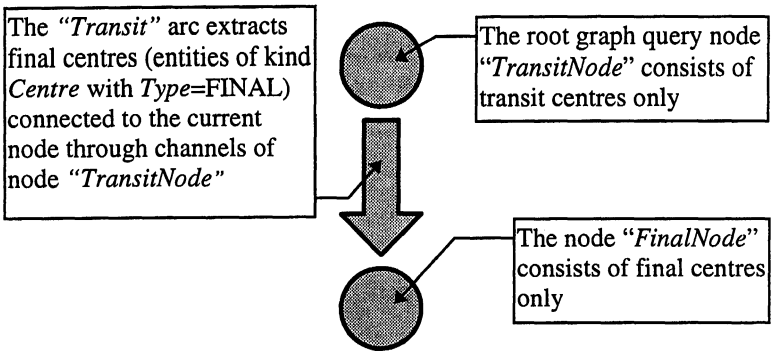

Fig. 6a. Structure of the graph query "Towns".

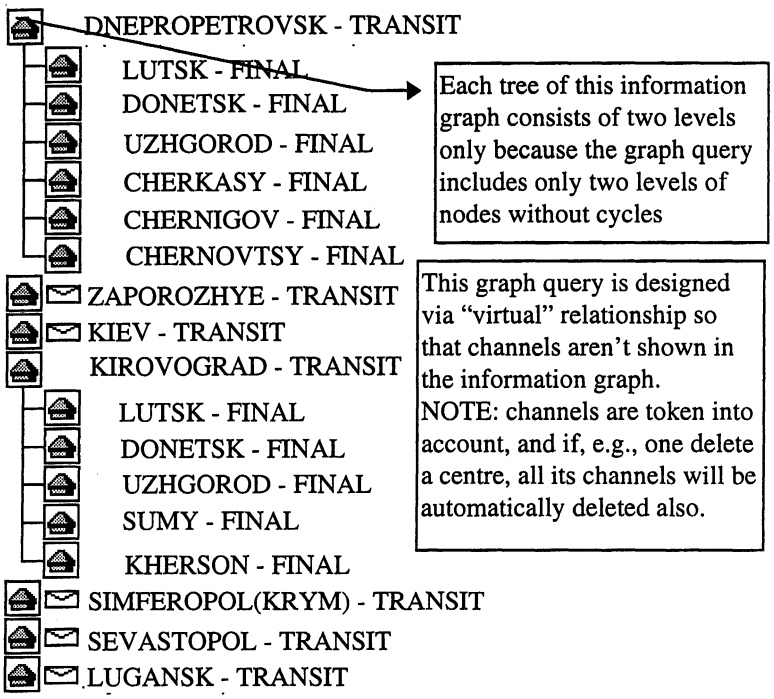

Fig. 6b. Fragment of the information graph corresponding to the graph query "Towns".

4.3. *Third Graph Query Example: Visual Search in the Nearest Neighbourhood*

This example (see Fig. 7a) can be viewed as a combination of the two previous ones. It presents both multilevel bypass trees and 'virtual' relationships. The information graph in Fig. 7b consists of such a sub-graph of the initial second telephone network that consists of connections from the center to its three nearest neighbours only.

One of the most interesting applications of ER-QBE tools is the built-in search for

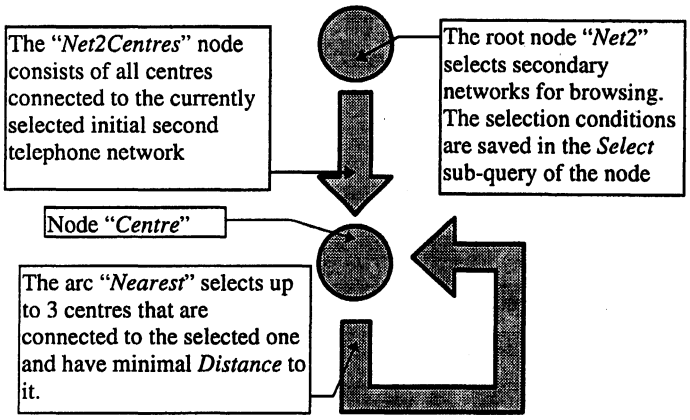

Fig. 7a. Structure of the graph query "3 nearest centers".

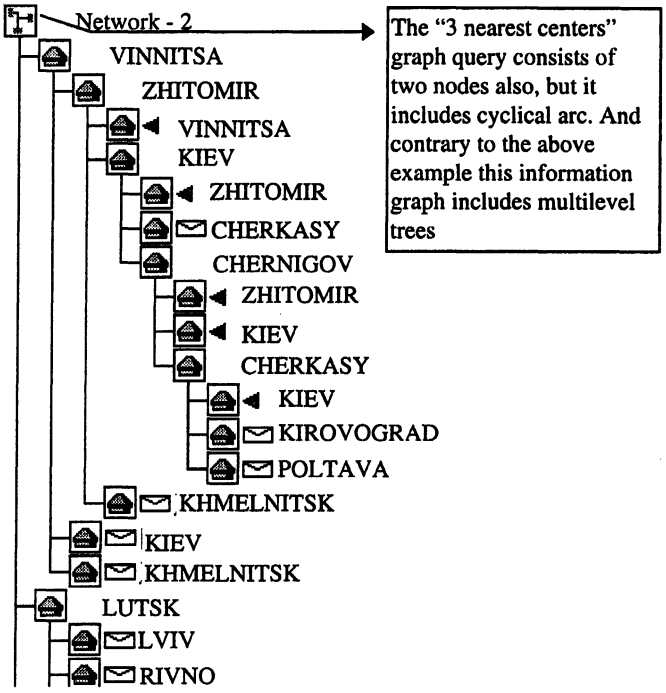

Fig. 7b. Fragment of the information graph corresponding to the graph query "3 nearest centers".

nodes of the information graph. As a result the program can find the nearest path, several paths or all non-cyclic paths from the current node to one or more destination nodes. An example of such a search is shown in Fig. 7c.

There are a number of problems that can be solved visually on the base of the visual search, e.g.:

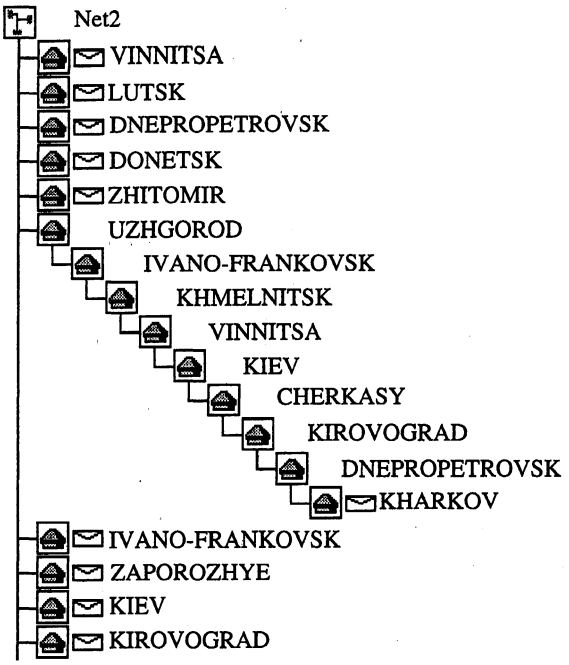

Fig. 7c. Route from Uzhgorod to Kharkov found in the information graph corresponding to the graph query "3 nearest centers".

? finding the nearest route from Kiev to Paris;
? finding ten mostly effective chains of distributors;
? finding the nearest common forefather;
? mutual offset of debts.

As to the implementation, the 'MicroPoisk' ER-QBE search support is part of a graph prototype, not the graph query. The user selects a destination node of the graph query and afterwards destination entities of the class corresponding to that node. Then the user can select an optional weight attribute or enter a weighing formula for each previous node. An SQL-query for selection of those entities and the weight calculation queries are automatically designed and put into the metabase. The program supports some parameters of the search: search "in depth" or "to expanse", maximal number of found paths (optional), and maximal search depth (optional). The user-break of the search process is also allowed.

4.4. *Visualization of Graph Queries*

As mentioned above, graph queries are kept in a metabase – a special repository. So they can be shown as trees of the information graph (refer to Fig. 8a). Fig. 8b shows a more detailed information graph that describes the whole graph prototype.

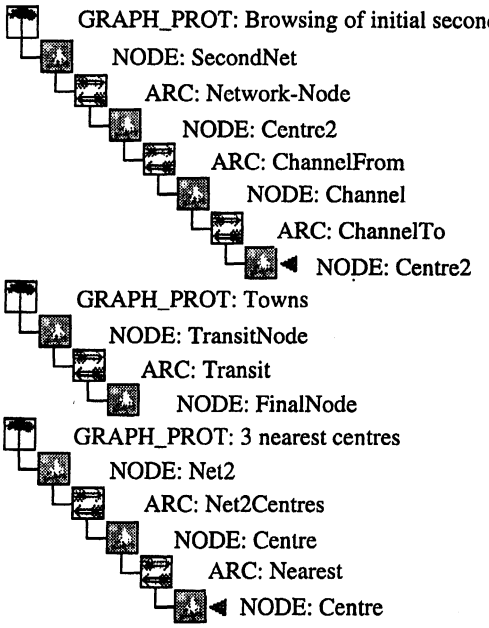

Fig. 8a. The above three graph queries saved in the metabase can be viewed as the information graph itself.

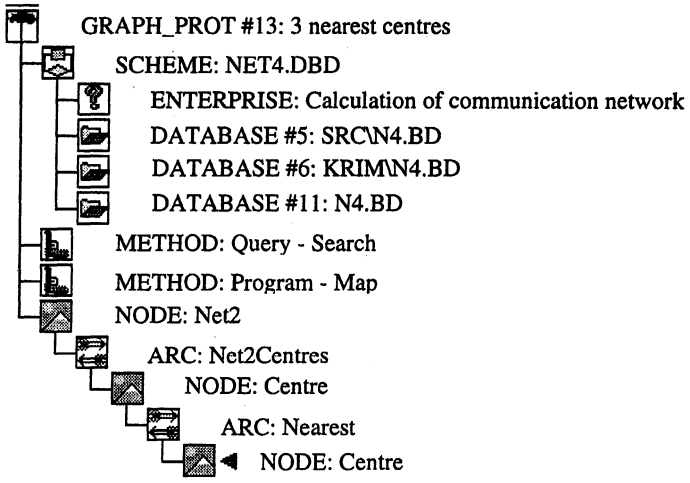

Fig. 8b. Graph prototype "3 nearest centers" shown as the information graph.

5. Application Domain

The graph query was discussed in this paper from the viewpoint of ER-based databases. We suggest that the idea of extensing QBE and using bypass trees for operating the network data structures isn't meaningless for a set of alternative approaches.

Multidimensional DBMS (Parsaye, 1997) are wide spread along with the OLAP technology intended for visual interpretation of the internal data pattern and verification of human-generated hypotheses involving multiple dimensions. OLAP can be successfully implemented for ordinary relational DBMS (so-called ROLAP), but their implementation for multidimensional DBMS (so-calledMOLAP) is much more effective. The multidimensional data model is based on a *cube* of data. The cube consists of events and each event is a union of several entities of different classes (dimensions). Hierarchies are attributes of dimensions. For example, hours, days, weeks, and months are hierarchically related elements within the time dimension. Closely related to the topic of hierarchies is the topic of aggregations. Aggregations by elements of the hierarchy are the most important functions of the multidimensional DBMS. A universal algebra of cubes has been developed as a mathematical basis of the multidimensional DBMS and some authors asseverate that the multidimensional model has to supplant the ER model from the database design market.

The result of OLAP is an information summary in tabular or graphical form. Unlike OLAP, ER-QBE can provide navigation and access to single entities in the multidimensional space. This feature is useful for multidimensional databases that are not automatically built from the relational sources and so are needed in edition. As shown above, graph queries are effective tools for the cube element visualization and processing (see Fig. 2). The 'Virtual' entities discussed in topic 4.2 support browsing through dimension hierarchies, that can be used as one of dynamic OLAP tools.

Object-oriented and object/relational DBMS (Atkinson *et al*, 1989). The object-oriented database design is a natural generalization of the ER one. Despite a number of successful software products developed during more than one decade history, object-oriented databases have not become a commercial standard. A new advance of this approach is related now with object/relational features of the SQL-3 standard (Frazer, 1998).

ER-QBE-like tools can be easily implemented in the object/relational environment, especially when an object GUI is based on a separate inner method or an object similar to the "MicroPoisk" visual formalism (Tulchinsky, 1996).

Deductive DBMS. The considered approach to the graph manipulation can be naturally expanded for the case of growing graphs. But the implementation of ER-QBE-like tools for deductive databases is strongly related to the application of the object-oriented paradigm for those structures.

6. Conclusion

General aspects of the purpose, capability, and implementation of the ER-QBE are considered in this paper from the practical viewpoint. More fundamental research of its power and limitations has not been finished so far. We intend to consider this subject in the next paper.

References

Atkinson, M., F. Bansilhon, D. DeWitt, K. Dittrich, D. Maier and S. Zdonik (1989). The object-oriented database system manifesto. *1st Int. Conf. Deductive and Object-Oriented Databases*, Kyoto, Japan, Dec. 4–6.

Chen, P.P. (1976). The entity-relationship model: Toward a unified of data. *ACM Trans. Database Syst.*, **1**, 9–36.

Codd, E.F. (1971). Normalized data base structure; a brief tutorial. In *Proc. of ACM SIGFIDET Workshop on Data Description Access and Control*, NY: ACM Press. pp.1–16.

Frazer, W.D. (1998). Object/relational grows up. *Database, programming & design*, **11**(1), 22–30.

Grechko, V.O., and V.G.Tulchinsky (1995). Building standard user interface: DBMS "MicroPoisk" approach. *Informatica*, **6**(2), 445–456.

Harrel, D. (1988). On visual formalism. *Comm. of ACM*, **31**, 514–520.

Kimball, R. (1997). A dimensional modeling manifesto. *DBMS*, **10**(9), 58–70.

Parsaye, K. (1997). OLAP and data mining: brinding the gap. *Database, Programming & Design*, **10**(2), 30–37.

Tulchinsky, V.G. (1996). An algebra-grammar approach to a user interface construction. *Kibernetika i Sistemny Analiz*, **5**, 169–179 (in Russian).

V.O. Grechko is a senior researcher at the Glushkov Institute of Cybernetics, Kiev, Ukraine. He received the Ph.D. in computer science from Glushkov Institute of Cybernetics in 1982. During the last decade he is a leader of research and programmer team developing ER-based "MicroPoisk" DBMS and their applications. His research interests include software development tools, databases, user's interface and knowledge representation.

P.G. Tulchinsky received the M.S. degree from Kiev Polytechnical Institute in 1996 and now studies computer science at the Glushkov Institute of Cybernetics, Kiev, Ukraine. His research interests include software engineering, user's interface development, databases, knowledge representation.

V.G. Tulchinsky is a member of research staff of computer science at the Glushkov Institute of Cybernetics, Kiev, Ukraine. He received the Ph.D. degree in computer science from the institute in 1995. His current research interests include databases, data mining and forecasting tools.

Nuo QBE prie grafinių užklausų

Valery GREČKO, Peter TULČINSKY, Vadim TULČINSKY

Straipsnyje pasiūlyta, kaip praplėsti QBE kalbą, įvedant grafinę notaciją, pritaikytą formuluoti užklausas sudėtingomis schemomis aprašomoms duomenų bazėms. Grafinėmis užklausomis sukuriamas vartotojo interfeisas ER modeliui, panašiai kaip QBE kalba jis sukuriamas reliaciniam duomenų modeliui. Straipsnyje taip pat aprašyta ER modeliu grindžiama DBVS "MicroPoisk" ir pateikti jos naudojimo pavyzdžiai.