A semantics based interactive query formulation technique

David Baer[1]
Paul Groenewoud[1]
Epaminondas Kapetanios[1,2]
S. Keuser[1]

[1]At the time of publication all authors were from the Swiss Federal Institute of Technology, Department of Computer Science, Institute of Information Systems, Switzerland.
[2]Epaminondas Kapetanios is now employed by the Harrow School of Computer Science, University of Westminster.

A Semantics Based Interactive Query Formulation Technique

D. Baer, P. Groenewoud, E. Kapetanios, S. Keuser

Swiss Federal Institute of Technology
Department of Computer Science
Institute of Information Systems
8092 Zurich, Switzerland
kapetanios@inf.ethz.ch

Abstract

We present an interactive query formulation technique which enables exploitation not only of structural properties of data but also of semantic constraints as posed by the contents of data. The technique aims at the formulation of a semantically consistent or meaningful query by the end-user without any previous knowledge of syntax formalisms and data model semantics. This has been achieved by end-user guidance in that an inference engine suggests semantically rich query terms for further consideration by the end-user. The set of suggested terms at each interaction stage comply with the already considered query terms with respect to structure and contents based semantics. Assignment or selection of operational terms are also allowed, if operational semantics comply with the semantics of data. The interactive query formulation component has been implemented in Java and runs on the client side of a client/server based query answering system architecture.

Keywords: *Visual query languages, Interactive query formulation, Semantic constraints, Knowledge-based querying.*

1. Introduction

Query languages as a means of communication between information systems and humans need to address all three levels of *syntax*, *semantics* and *pragmatics* in order to provide an intuitive and human oriented way of dealing with information extraction. From all three levels, *syntax* is the best understood. *Semantics* and *Pragmatics* are still considered a critical issue when a communication process in terms of querying of information systems by humans is going to be established.

Query languages, however, provided by database management systems are mainly designed for programmers and they underly a syntax formalism. Furthermore, it is required that end-users have a substantial knowledge of logic and the database structure in order to formulate queries. Even easy-to-use languages such as SQL are intimidating for average users and require formal training to use. A solution can be provided by application programming which forms a middle layer between database systems and end-users. But applications with pre-programmed queries cannot satisfy the dynamic query needs of users, especially when scientific databases and decision support systems are considered.

In order to meet the dynamic query needs of end-users, interactive and visual query formulation techniques have been proposed. The former relies upon guidance through semantic models for incremental or associative query answering for relevant information and incremental query sessions [32, 31]. The goal of such interactive query formulation techniques is to provide an end-user- and context-sensitive assistance based on database modeling and probabilistic reasoning techniques combined with linguistic-based ones. To this extend, query formulation takes place in terms either of query completion of incomplete queries or suggestions of further queries to reach a complex query goal, or relevant attributes or topics which might be of interest to the users [15, 10, 8, 20, 27, 11, 9].

Visual query languages, on the other side, rely upon visual formalisms and metaphors [22] which results in visual query languages [5] and systems (VQSs). A classification of such systems is given in [6] based on the criteria of visual representation of queries and query results. They are mainly classified into *form-based* [28, 18, 23, 29, 30, 33], *diagrammatic* [17, 3, 16] and *iconic* VQSs [26] according to the representation of the domain of interest for query formulation and/or the query result. Some hybrid systems [25, 21, 12] have also been proposed. They all refer to information repositories dealing with alphanumeric data and

1530-1893/01 $10.00 © 2001 IEEE

not with semi-structured, video or audio data.

In order to provide an interactive query formulation technique which embed semantic constraints, we elaborated a knowledge-based query formulation technique based on a semantic model which captures both the structure and semantic contents of data. The end-user/system interaction strategy enables the construction of a query in terms of *concepts, relationships, properties, values* or *value domains*, as well as *operations*. They all constitute the vocabulary of the visual query language [24] and are represented by a semantic model in a multi-lingual mode. This enables construction of a query with terms from a particular natural language (NL) without affecting the query results.

Since the interaction strategy exploits semantic constraints, only reasonable or meaningful queries can be constructed by the end-user. Therefore, a major part of semantic query optimization is done during the interactive query construction and, therefore, complex transformation rules are avoided when a query is being parsed after formulation in order to optimize it semantically [4]. Furthermore, addressing the same query result by having formulated a query with linguistic items in more than one natural language enhances dynamic query needs in a multi-lingual environment.

On the other side, since the suggested set of query terms must be compliant with the given semantic constraints, only a minimal set of query terms is presented to the end-user for selection. Therefore, the semantics based interaction technique frees the end-user from the task of understanding complex diagrammatic representations of data models as well as minimize the set of query elements which currently appear on the query formulation user interface.

2. System considerations

Prior to the description of the interaction strategy for the query construction (see next section), an overview of the implemented query system is depicted in figure 1. It gives an overview of the major components participating in a query construction and submission session which takes place at the client's workspace. The main components are:

a) the *query construction user interface*, where the query is being assembled in terms of natural language words and their signification. The words refer to both i) application domain terms such as *acute myocardial infarction, stress test* and ii) logical operators (AND, OR, NOT), comparison operators (e.g., less than, equals, etc.) and operations (e.g., average, deviation, frequencies, etc.).

b) the *semantic information space* of the *domain terminology*. This has been achieved by forming a semantic graph with preconditions where the nodes represent all potential query terms with their semantic depth due to a particular application domain. The semantic information space also includes semantic constraints as posed to value domains such as well-restricted or dynamically formed value domains, or set of attributes to be considered when application specific circumstances are taken into account.

c) an *inference engine* which acts upon the semantic information space in order to respond to the users' requests for suggestions of query terms during query construction. The inference engine is needed each time suggestions for the consideration of further query terms are requested by the end-user in order to extend the current set of query terms already included in the potential query.

d) the *query transformation engine*, where each constructed query with terms of the terminology base is transformed into a SELECT-PROJECT-JOIN query. This is achieved by transforming the semantic query tree which corresponds to the constructed query into either a query execution tree with operations from the relational algebra or to SELECT-FROM-WHERE statements as set of tokens. Moreover, the words in a particular natural language have to be mapped into the names given to the elements of the data storage model such as relations, attributes, values and/or operations. This enables the natural language independence of retrieving a query result, since an SQL statement can be reached regardless which natural language has been selected in order to express the query terms.

e) an object-relational DBMS such as ORACLE.

In order to start the interactive query formulation session, the inference engine needs to be downloaded to the client site. Subsequently, an NL-specific set of query vocabulary terms is imported from a terminology server according to the language preference of the end-user (figure 1). The inference mechanism is responsible for the suggestion of terms when a given query context as well as the structural and semantic constraints of data are given. These refer to both application domain terms and operational ones.

The final constructed query will be a rooted graph SQT for which it holds that $SQT \subset G(N, E)$, where SQT is a *tree* (there is a unique path from root to n for every $n \in N, n \neq root$) and $G(N, E)$ is the knowledge graph representing data and operational semantics of an application domain. The query tree, however, includes nodes which are semantically consistent with respect to the expressed semantic constraints.

The transformation algorithms for the constructed query trees out-lie the scope of this paper. However, given the semantic information as refers to the role of the query nodes (concept, relationship, property, value) as well as the relationships of the symbols on the query trees nodes to the elements of the implementation (target database) data model, we are in a position to construct simple SELECT-PROJECT-JOIN queries with AND-connected conditional statements which are free of semantic conflicts.

Currently, we are extending the expressive power of the

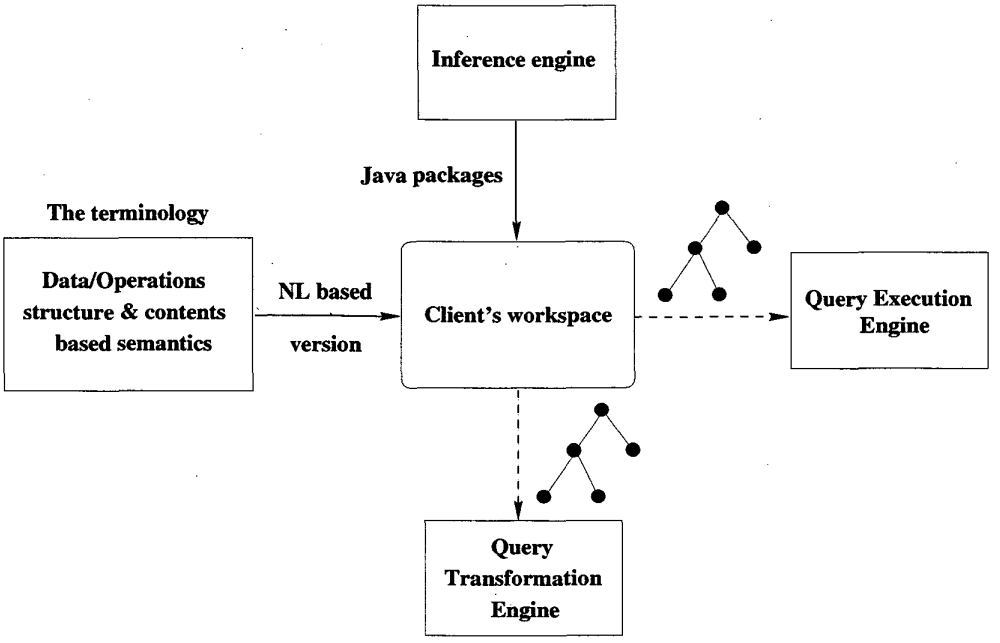

Figure 1. Overview of the interactive query construction system

interactively formulated queries in that comparison operators and operations can be selected by the end-user and assigned to the elements of the query. The assignment of comparison operators and operations to terms (nodes) by the end-user, however, is still subject to semantic constraints. These refer not only to the classifications of terms as *concepts* or *properties* but also to the notion of properties such as *categorical* or *numerical variables*. The same holds for comparison operators which are suggested to the end-user according to the notion of the value currently assigned to a property. For example, the comparison operators $>$ and $\geq$ are suggested in order to be assigned to a value node of the query tree, only if the value is a *numerical* one.

In this sense, the constructed query and, respectively, the submitted query tree, can be enhanced by operations/operators which are semantically consistent with the query terms (nodes) currently constituting the query.

Three-tier architecture: The components-based construction of the system can be mapped onto a three-tier system architecture in that component (e) is assigned to the back-end layer, components (b), and (d) are assigned to the middle layer and, finally, components (a) and (c) are assigned to the front-end layer. However, a query construction session which takes place at the client makes use of component (b) which is downloaded from the application server at the middle layer. In particular, the inference engine also runs at the client's site and, therefore, there is no network communication overhead during a query construction session. Moreover, in order to reduce the amount of data to be downloaded to the client's site as provided by component (b) and needed by component (c), only that part of (b) is downloaded which corresponds to the preferred natural language in which query terms should appear.

Given this system architecture, there is a shift of the query construction logic from the middle layer to the clients. The application server deals only with the query transformation logic (component (d)). Given also that components (a), (c) and (d) are implemented in the *Java* programming language, the interactive query construction components of the system, i.e. (a) and (c), can be installed on any client running under Windows 98/NT/2000 as well as Linux operating systems without the need of preinstalling a Java Virtual Machine (JVM). A JVM 2 (1.2 or 1.3) accompanies the software installation. Component (d) runs as an application server and is contacted only if the client (domain scientist) needs to submit a query for execution.

Furthermore, maintenance of the semantic information space (component (b)) is done at the middle layer independently of the interactive query construction software installed at the client site. Any changes to the semantic infor-

mation space become available immediately - change once run everywhere - after the start of a new query construction session. Hence, the query vocabulary can be adapted according to a dynamically change of application domain semantics.

In the following, we focus on the query formulation strategy as a means of user/system interaction in order to construct a query with respect to the semantic context as given by a particular application domain.

3. Interactive query formulation strategy

3.1. The modes of meaning for queries

We mainly distinguish between instances description oriented terms and operational terms. At the beginning of a query construction session, the end-user is requested to select a particular natural language in which the suggested terms will appear. Consequently, the end-user is requested to describe, first, the set of instances which she/he is interested in. The description of a requested set of instances reflects the formulation of a concept through the *extensional and intensional modes of meaning*[1] of natural language words.

Subsequently, comparison and/or logical operators as well as (analytical) operations can also be assigned to the constructed queries and, accordingly, to the corresponding query tree, on the basis of semantic constraints too. In the following, we mainly focus on meaning based interaction strategy as applied to instances description oriented terms.

The extensional mode of meaning: The description of relevant instances is done in terms of *concepts*, *relationships*, *properties* and *values*. *Concepts* are terms within the query vocabulary which stand for the *extension* of a word or phrase. This is understood to be the timeless class of all *things* which properly 'fall under' or are described by that phrase. The extension of a term is fixed: it is not at some time one thing and at some other time something else. The extension of a term includes all past things (if any) plus all present things (if any) plus all future things (if any) that fall under or are described by the term.

For example, the word "patient" - assuming that terms appear in English - has as its extension all patients in the past, present, and future. The phrase "postmenopausal patients" has as its extension all (past, present, and future) postmenopausal patients (i.e. a proper subset of the former class). There are (at least) two common synonyms of "extension". They are: "denotation" and "reference". The members of the denoted class are often spoken of as "the denotata" (sing. "denotatum") or "the referents".

[1] We rely on the definitions provided by the philosopher *Norman Swartz*

Initially, the end-user is requested to describe the instances by using the words (concept terms) which capture *extension* as mode of meaning. For this purpose, the system suggests a set of initial concepts or topics to start with. Having selected a particular concept, the system might suggest, on the request of the end-user, a set of concepts which are related to the selected one. Selecting a concept out of the set of suggested concepts and adding it to the previous one would refine the extensional meaning of the current word or phrase. Further refinements can take place recursively as long as there are related concepts to the lastly considered ones.

For example, having selected the concept 'patient' out of an initial set of suggested concepts, the system would further suggest concepts such as 'premenopausal', 'postmenopausal', which are related to the concept 'patient'. Having selected 'postmenopausal', the concept takes the form of a concept description path 'patient' - 'postmenopausal' which refers extensionally to all 'postmenopausal patients' as a subset of 'patients'. At a further stage, 'leiomyomata' can be added to this concept description path.

The intensional mode of meaning: Having defined what is the major concept we are talking about, the end-user is now requested to circumscribe the described instances or things by defining their intensional meaning. Intuitively, what we want to capture in our technical definition is the concept of the *defining characteristics* (sometimes called "the essential characteristics") of a thing. There is a certain relationship that holds between intension and extension. As the intension of an expression increases, the class denoted, i.e. the extension, (generally) decreases.

In this sense, *relationships*, *properties* and *values* serve as a mechanism for capturing the intensional meaning of things. For example, in order to define the intensional meaning of 'postmenopausal patients with leiomymata', the system suggests a set of potential properties which might be used to define the intensional meaning of a currently defined concept. Note that only those properties are suggested which are related to each of the concepts which currently participate in the concept description path. For instance, the properties 'having symptoms', 'major impairment', 'uterine growth' and 'hormonal replacement therapy' will be suggested to the end-user since 'having symptoms' is a property related to the concept 'postmenopausal', 'major impairment', 'uterine growth' and 'hormonal replacement therapy' are properties connected to the concept 'leiomyomata'.

Similar to the construction of concept description paths, property description paths can also be defined which refer to structural recursiveness of properties. This can be achieved by recursively requesting related properties to the lastly selected ones. This enables description of properties

in a multi-dimensional space. For example, 'major impairment' - 'abdominal' indicates the fact that we focus on that particular 'major impairment' as related to an abdominal area.

Another major contribution to the intensional mode of meaning is the consideration of any relationships which might relate concepts or concept description paths to each other. At a further stage, the end-user is requested to select a potential relationship out of a set of suggested relationships in order to further circumscribe the current concept. For instance, having selected the relationship 'operated by' leads to another concept such as 'hospital'. Consequently, the same philosophy for concept circumscription can be applied to the concept 'hospital'.

Data contents based semantic constraints: Up to now, we referred to the intensional mode of meaning in terms of structure (concept, relationships, properties) and not in terms of contents of a real world application. However, the latter plays an important role when we aim at interactively formulating a query which is consistent with the contents of data. For this purpose, the inference mechanism for term suggestion also takes into consideration any semantic constraints as posed by the data contents.

In this sense, the end-user might select values, in order to instantiate properties and form a conditional statement for the query, only out of a suggested set of values (value domain) which semantically reflects the data contents. This applies not only to suggested value terms but also to the suggested set of properties and relationships. Generally speaking, the suggestion of terms is made relevant to the consideration of *preconditions*.

For example, given that the query context includes the concept description path 'patient' - 'postmenopausal' - 'leiomyomata', the properties 'having symptoms', 'major impairment', 'uterine growth' and 'hormonal replacement therapy' might be instantiated with value terms as being members of well-restricted value domains such as {*yes, no*} to be assigned to the first two properties, {$\geq$ 20% *within 6 months, none*} to be assigned to the third one and {*received, not received*} to be assigned to the fourth property. The suggestions of these particular value domains are made upon the request of the end-user for a particular property instantiation.

In order to avoid combinations of {*property, value*} pairs within the given query context as stated above which lead to unreasonable or meaningless queries, suggestion of property and/or value terms are also subject to satisfaction of preconditions. For instance, instantiation of the property 'major impairment' makes sense only if property 'having symptoms' has been instantiated with the value term 'yes' and, therefore, the property 'major impairment' will be suggested to the end-user only if this particular precondition is satisfied. Similarly, the property 'hormonal replacement therapy' will be suggested to the end-user only if the precondition that the value of the property 'uterine growth' has been set to '$\geq$ 20% within 6 months' is satisfied.

Satisfaction of semantic constraints might also have an impact on the set of value terms to be inferred and suggested within a given query context. For example, we consider the concept description path 'patient' - 'pre/perimenopausal' - 'leiomyomata' - 'asymptomatic', and the relevant properties 'estimated uterine weight' and 'uterine growth'. The latter can be instantiated either with one of the value terms {$\geq$ 20% within 6 months, $<$ 20% within 6 months}, if it happens that the value term $\geq$ 300 $grams$ has been assigned to 'estimated uterine weight', or, alternatively, with one of the value terms {$\geq$ 50% within 6 months, $<$ 50% within 6 months}, if it happens that the value term $<$ 300 $grams$ has been assigned to 'estimated uterine weight'.

Semantic constraints can also be expressed by arithmetic value domains. They are usually expressed by arithmetic intervals and, in some cases, by characteristic measurement units. Assignment of preconditions to arithmetic value domains have an impact on the range of the values to be considered as a semantically consistent value domain. For instance, the arithmetic interval $[10, 20]$ is suggested for property instantiation only if the selected measurement unit is set to $dl/gram$. Similarly, the range of salaries of doctors is defined dynamically after making known to which hospital they belong. The consideration of preconditions, however, during query formulation has a major impact on the interaction strategy, since we first need to know all relevant information before deciding upon satisfaction of a precondition.

For example, given that the concept description path is 'patient' - 'postmenopausal' - 'leiomyomata', only the properties 'having symptoms', 'uterine growth' will be suggested first, since these properties and their value domains are not bound to any preconditions. Appearance of the rest of properties is determined upon instantiation of the first two properties. In the following, we will have a brief overview of the semantic model as represented by a knowledge graph, which provides the representation model for the structure and content based semantics of data.

4. An overview of the semantic data model

The data model represents all query vocabulary terms by capturing both structural properties and semantic constraints of the data is a (knowledge) acyclic graph (N, E), where N is the set of nodes $t \in N$ and E the set of edges $e \in E$. Associated with each edge $e \in E$ is an ordered pair of nodes, the source node $s(e)$ and the target node $t(e)$. A *path* is a sequence of edges $e_1, e_2, ..., e_k$ such that

$t(e_i) = s(e_{i+1}), 1 \leq i \leq k-1$. Such a path is called a path from the source $s(e_1)$ to the target $t(e_k)$. A node is a *terminal node* or a *leaf* if it is not the source of any edge in E.

The nodes of the knowledge graph carry most of the data referring to the vocabulary terms. Note that this is in contrast with edge-labeled graphs as data model proposals for *semi-structured* data [1] or self-describing data formats used for exchanging data [2, 19, 7] where most of the data are carried by edges. The edge-labeled graphs as data representation model goes back to GraphLog [13, 14].

In this sense, each node represents a query vocabulary term. However, nodes are also referred as *objects* having object identities called *tuis* (term unique identifiers). The syntax of node labels or structure of objects is given by a tuple such as {*tui, word, connotation, symbol, role :: precondition*}, where it holds that:

word is the natural language element representing the term in a particular natural language, e.g., *major impairment*,

connotation helps in clarifying the meaning of (synonym or homonym) terms, e.g., *'during a 3 month period, the patient stayed home (missed work or could not take part in social activities) for at least 1 day per month due to pain or feeling badly'*,

symbol is the chosen implementation symbol, e.g., the attribute $MImpair$ in a table,

role is its classification as concept or relationship, property, value or value domain.

Preconditions are optional and can be assigned to any node of the knowledge graph in order to express semantic integrity constraints. Terminal or leaf nodes represent value terms.

5. Conclusion

An interactive query formulation technique has been presented in this paper, which aims at the formulation of a semantically consistent or meaningful query by the end-user without any previous knowledge of syntax formalisms and data model semantics as well as at the avoidance of any diagrammatic notations for interactive query formulation. This has been achieved by end-user/system interaction strategy where the end-user is prompted with semantically rich query terms as suggested by an inference engine according to a preferred natural language and with respect to the represented structural and data contents based semantics of an application domain.

The query system is currently used in a variety of data intensive applications including medical information and social life case studies. A usability study with real users (ca. 5-7) showed that a constraints-based interaction with the system not only leads to the construction of meaningful or semantically optimized queries but also reduces the amount of terms and visual metaphors appearing on the graphical user interface for query formulation. Since these constraints also reflect the real world application, users' satisfiability increased when they noticed that the system inferences reflect their own world, e.g., meaningless values for particular properties of patients were unacceptable for end-users working in a clinical environment.

References

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to semistructured Data and XML.* Morgan Kaufmann Publishers, 2000.

[2] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The Lorel Query Language for Semistructured Data. *International Journal on Digital Libraries*, 1(1):68–88, April 1997.

[3] M. Angelaccio, T. Catarci, and G. Santucci. QBD*: a graphical query language with recursion. *IEEE Transactions on Software Engineering*, 16(10):1150–1163, 1990.

[4] E. Bertino and D. Musto. Query Optimization by Using Knowledge about Data Semantics. *IEEE Trans. on Knowledge and Data Engineering*, pages 121–155, 1992.

[5] J. Cardiff, T. Catarci, and G. Santucci. Semantic query processing in the VENUS environment. *International Journal of Cooperative Information Systems*, 6(2):151–192, June 1997.

[6] T. Catarci, M. Costabile, S. Levialdi, and C. Batini. Visual query systems for databases: A survey. *Journal of Visual Languages and Computing*, 8(2):215–260, April 1997.

[7] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proc. of Info. Processing Society of Japan*, Tokyo, Japan, October 1994.

[8] W. Chu and Q. Chen. Neighborhood and Associative Query Answering. *Intelligent Information Systems*, 1(3/4):355–382, 1992.

[9] W. Chu and Q. Chen. A Structured Approach for Cooperative Query Answering. *IEEE Transactions on Knowledge and Data Engineering*, 6(5):738–749, October 1994.

[10] W. Chu, R. Lei, and Q. Chen. Using Type Inference and Induced Rules to Provide Intensional Answers. In *Proc. of the 7th Intern. Conf. on Data Engineering*, April 1991.

[11] W. W. Chu, H. Yang, K. Chiang, M. Minock, G. Chow, and C. Larson. CoBase: A scalable and extensible cooperative information system. *Journal of Intelligent Information Systems*, 1996.

[12] L. Cinque, S. Levialdi, and F. Ferloni. An expert visual query system. *Journal of Visual Languages and Computing*, 2:101–113, 1991.

[13] M. Consens and A. Mendelzon. Graphlog: A visual formalism for real life recursion. In *Proc. of the ACM Symp. on Principles of Database Systems*, pages 404–416, 1990.

[14] M. Consens and A. Mendelzon. The G++/Graphlog visual query system. In *SIGMOD Conference*, 1990.

[15] F. Cuppens and R. Demolombe. Cooperative Answering: A methodology to provide intelligent access to databases. *Expert Database Systems*, pages 621–643, 1989.

[16] Y. Dennebouy, M. Anderson, A. Auddino, Y. Dupont, E. Fontana, M. Gentile, and S. Spaccapietra. SUPER: Visual interfaces for object + relationship data models. *Journal of Visual Languages and Computing*, 6:74–99, 1995.

[17] R. Elmarsi and G. Wiederhold. GORDAS: A formal high-level query language for the entity-relationship model. In *Proc. of the 2nd Inter. Conf. on Entity-Relationship Approach*, pages 49–72, Washington D.C., U.S.A., 1981.

[18] R. Epstein. The TableTalk query language. *Journal of Visual Languages and Computing*, 2:115–141, 1991.

[19] J. H. et al. Information translation, mediation and mosaic-based browsing in the tsimmis system. In *Proc. of the ACM SIGMOD Conference*, page 483, May 1995.

[20] G. Fouque, W. Chu, and H. Yau. A Case-Based Reasoning Approach for Associative Query Answering. In *Proc. of the 8th International Symposium on Methodologies for Intelligent Systems*, Charlotte, NC, October 1994.

[21] I. Groette and E. Nilsson. SICON: An icon presentation module for an E-R database. In *Proc. of the 7th Inter. Conf. on Entity-Relationship Approach*, pages 271–289, Roma, Italy, 1988.

[22] E. Haber, Y. Ioannidis, and M. Livny. Foundations of Visual Metaphors for Schema Display. *Journal of Intelligent Information Systems*, 3:263–298, 1994.

[23] G. Houben and J. Paredaens. A graphical interface formalism: specifying nested relational databases. In T. Kunji, editor, *Visual Database Systems*. North-Holland, 1989.

[24] E. Kapetanios, M. Norrie, and D. Fuhrer-Stakic. MDDQL: A Visual Query Language for Meta-data Driven Querying. In *5th IFIP Inter. Working Conf. on Visual Database Systems*, Fukuoka, Japan, May 2000. Kluwer Academic Publisher.

[25] R. King and S. Melville. SKI: A semantic knowledgeable interface. In *Proc. of the 10th VLDB*, pages 30–37, Singapore, 1984.

[26] A. Massari, S. Pavani, L. Saladini, and P. Chrysanthis. QBI: Query by icons. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, page 477, San Jose, USA, 1995. ACM Press.

[27] A. Motro. Intensional Answers to Database Queries. *IEEE Transactions on Knowledge and Data Engineering*, 6(3):444–454, June 1994.

[28] G. Ozsoyoglou and H. Wang. Example-based graphical database query languages. *COMPUTER*, 26(5):25–38, May 1993.

[29] Y. Shirota, Y. Shirai, and T. Kunii. Sophisticated form-oriented database interface for non-programmers. In T. Kunji, editor, *Visual Database Systems*, pages 127–155. North-Holland, 1989.

[30] L. Wegner. ESCHER: interactive visual handling of complex objects in the extended NF2 database model. In T. Kunji, editor, *Visual Database Systems*, pages 277–297. North-Holland, 1989.

[31] G. Zhang. *Interactive Query Formulation Techniques for Databases*. PhD thesis, University of California, Los Angeles, 1998.

[32] G. Zhang, W. W. Chu, F. Meng, and G. Kong. Query Formulation from High-Level Concepts for Relational Databases. In N. Paton and T. Griffiths, editors, *Proc. User Interfaces to Data Intensive Systems, UIDIS 99*, pages 64–74, Edinburgh, Scotland, September 1999. IEEE Computer Society Press.

[33] M. Zloof. Query-by-example; a database language. *IBM Systems Journal*, 16(4):324–343, 1977.