# WDBQL: A Web-Based Query Language for Remote Relational and Object-Oriented Databases

Eric Andonoff[1], Christian Sallaberry[2] and Nicolas Belloir[1]

[1] IRIT/UT1, Université de Toulouse 1, 1 place Anatole France, 31042 Toulouse, France
ando@irit.fr
[2] Université de Pau, UFR Droit & IAE, Avenue du Doyen Poplawsky, 64000 Pau, France
Christian.Sallaberry@univ-pau.fr

**Abstract.** This paper presents WDBQL, a visual query language intended for casual users. WDBQL is a Web-based query language. It means that WDBQL implements the Web navigational mechanism to query databases. Such a mechanism has two main advantages: (i) the querying process is really easy to use for casual users and (ii) the querying process hides the semantics of the performed operations. WDBQL allows to query remote relational and object-oriented databases.

## 1 Introduction

We have collaborated with Local Communities to produce WDBQS (Web DataBase Query System), an ongoing project that aims to define a visual language intended for casual users for distant relational and object-oriented database querying [4]. Local Communities handle a set of information systems implemented in several databases. These databases are independent one from each others, are either relational or object-oriented and are distant: we call them remote databases. The problems we face when defining and implementing WDBQS are (i) the definition of an intranet architecture to query remote databases, and (ii) the definition of a query language well-suited for casual users. This paper addresses the WDBQS query language, called WDBQL (Web DataBase Query Language), we proposed for casual users.

Recent position papers in the database field [16,18] have highlighted the importance of providing user-friendly database query languages for casual users. Indeed, commercial database systems (except some of them such as Access or Paradox) only propose text-based query languages such as SQL or OQL. Such languages are not made suitable for casual users because they have a strict syntax and enforce the users to know the concepts of the underlying database model. Moreover, the Web popularity further enforces the need to develop user-friendly and easy to use query languages for casual users.

To face these limits, several visual query languages for casual users have been proposed in the literature. These languages may be classified into three categories [10]: form-based query languages which allow the expression of queries from forms (e.g. QBE [20], VQL[19], PESTO [6]), graph-based query languages which allow the expression of queries from graphs (e.g. QBD [2], SUPER [3], OHQL [1], QUIVER

[9]) and multi-paradigm query languages which combine text-based, form-based and graph-based paradigm for querying (e.g. ICI [7], EVE [10]). But even if these query languages, particularly the graph-based and multi-paradigm ones, are really an improvement with respect to text-based query languages, they still propose a query formulation which is sometimes non trivial for casual users who have both to know the principles of the querying process and the semantics of the performed operations.

We propose to use a Web interface to query databases. Indeed, the navigational mechanism which underlies Web interface is well accepted by end-users. We believe that database casual users will benefit from its simplicity of use when querying. Therefore, we defined a new category of visual query languages: the Web-based one. Languages belonging to this category permit the expression of Web queries i.e. queries expressed from a Web interface using a navigational language. To the best of our knowledge, DataGuides [12], which allows the querying of semi-structured databases, is the only Web-based language of the literature. Unfortunately, [12] only gives minimum information about the query language itself.

WDBQL, the query language we defined in the WDBQS system, continues the DataGuides works in the context of remote relational and object-oriented databases. It has the following characteristics:

- It is based on the ODMG model which is used to describe in a meta-schema the remote relational and object-oriented database schemas.
- It is a Web-based query language mainly intended for casual users. It is accessible via the Web in an intranet architecture. It advocates a navigational query formulation mechanism and hides the semantics of the performed operations.

The remainder of the paper is organised as follows. Section 2 first gives an example which will be used through the paper and then presents the WDBQL approach to model and visualise the remote relational and object-oriented database schemas. Section 3 describes WDBQL. It first explains the WDBQL querying principles, then introduces the interface implementing the query language, and finally gives examples of WDBQL queries. Section 4 gives a comparison with related works while section 5 concludes the paper.

## 2 The WDBQL Models

We first give the *County Council* database running example which manages boroughs, town halls, citizens, elected representatives and clerks. Because of space limitation, we only give an implementation of this example in a remote object-oriented database. O2 is chosen as object-oriented database system.

```
class Borough type tuple
( Bcode : integer,
  Bname : string, Bcity : string,
  Mayor : Elected,
  Deputies : set(Elected),
  Counselors : set(Elected),
  MainTownHall : TownHall,
  SecondaryTownHalls : set(TownHall) )
end;
```

```
class Person type tuple
( Pcode : integer,
  Surname : string,
  Firstnames : set(string),
  Address : tuple( street, zip, city : string ),
  Married : Person,
  Birthdate : Date )
end;
```

```
class Elected inherits Person type tuple
( Arrival : Date, Grade : string )
end;
```

We now introduce the two WDBQL data representation levels: the database level and the user level. The database level stores the description of the remote relational and object-oriented database schemas. The user level allows the interpretation of the meta-schema using a more simple representation model based on node, link and anchor notions.

## 2.1 The Database Level

WDBQL stores the description of the remote database schemas as instances of a meta-schema. This meta-schema is described using the standard ODMG-93 [8]. We choose the ODMG model because it generalises relational and object-oriented database concepts.

The meta-schema is composed of a set of meta-types implementing the main ODMG model concepts. The specified meta-types are the Base, Type, Property, TypeProperty, BasicTypeProperty, MultiTypeProperty, StructuredType-Property, Operations and Relationship ones. Base describes the remote databases which can be queried while Type describes the tables or classes defined in the remote databases. Property and TypeProperty describe the properties of theses tables or classes and their corresponding type which can be mono-valued (BasicTypeProperty is used for basic properties), multi-valued (MultiTypeProperty is used for collection properties), or structured (StructuredTypeProperty is used for structured properties). Operation describes the operations defined in the classes of the remote databases while Relationship describes the set of relationships existing between types. Relationships can also be mono-valued or multi-valued.

## 2.2 The User Level

WDBQL proposes a user model based on Web notions which are the node, link and anchor ones. Such a model obviously better suits casual users.

The user model is a representation of the meta-schema instances in terms of nodes, links and anchors. Each node has a name and is either terminal or non-terminal: terminal nodes are not connected to others by a link, while non terminal nodes are at least connected to one another by a link. The database model and the user model are perfectly matched as we can see in table 1.

| **Database Model** | **User Model** |
|---|---|
| Base | Non-terminal node |
| Type | Non-terminal node |
| Property<br>- Mono-valued<br>- Multi-valued<br>- Structured | <br>Terminal node<br>Terminal node<br>Non-terminal node |
| Operation | Terminal node |
| Relationship<br>- Mono-valued | <br>Non-terminal node |

| | |
|---|---|
| - Multi-valued | Non-terminal node |

**Table 1**. Correspondence between the Database Model and the User Model

The instances of the meta-types *Base*, *Type* and *Relationship* correspond to non-terminal nodes in the user model while instances of *Operation* correspond to terminal nodes. Instances of *BasicTypeProperty* (mono-valued property) and *MultiType-Property* (multi-valued property) are represented as terminal nodes while instances of *StructuredTypeProperty* (structured property) correspond to non-terminal nodes. Links are defined either between non-terminal nodes or between a non-terminal node and a terminal node. The figure below partially illustrates that correspondence with respect to the running example.

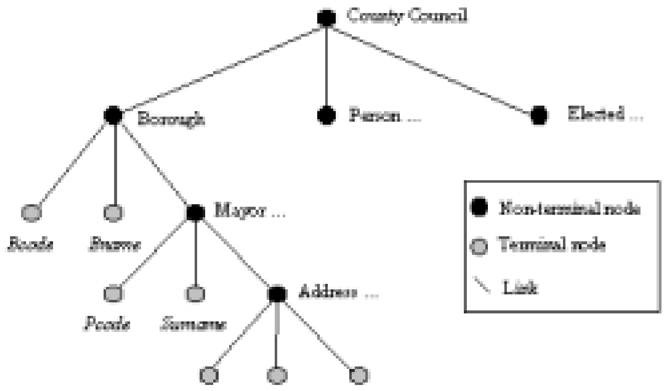


**Fig. 1.** User Model for the Running Example

## 3 The WDBQL Query Language

WDBQL is a Web-based query language intended for casual users. It is accessible via the Web in an intranet architecture. It advocates a navigational query formulation mechanism which is really easy to use for casual users, and which hides the semantics of the performed operations.

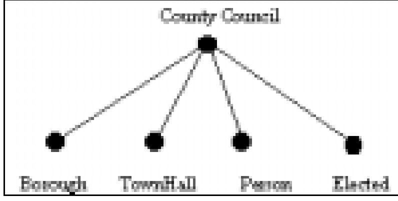
### 3.1 The Querying Principles

WDBQL supports the user model. That means that the queried database schema is visualised as a set of nodes, links and anchors. A query is a set of actions on the nodes. We distinguish two types of actions which are navigation and selection.

#### 3.1.1 Navigation

Navigation only concerns non-terminal nodes. It is possible thanks to the anchors associated to each non-terminal nodes of the database. The name of a node is used as the corresponding anchor flag. The selection of an anchor visualises all the nodes linked to the node corresponding to the anchor.

We illustrate the navigation in the *County Council* database. The *County Council* database is composed of the types *Borough*, *TownHall*, *Person* and *Elected*. When selecting the *County Council* database, these four types are visualised as shown below:

| User Model | Result of an Action of Navigation | |
|---|---|---|
|  | County Council<br>Borough<br>TownHall<br>Person<br>Elected | ← The selection of the node *County Council* implies the visualization of the linked nodes |

The user can go on its navigation selecting the non-terminal node Borough. All the nodes linked to the node Borough are then visualised.

### 3.1.2 Selection

Selection concerns both terminal and non-terminal nodes. It has different semantics which depend on the type of the node. We distinguish four types of nodes: mono-valued terminal node, multi-valued terminal node, mono-valued non-terminal node and multi-valued non-terminal node.

The following tables present the operators and operands which can be used to define selection predicates. Because of space limitation, we only give the tables for non-terminal nodes.

| Operators | Operands |
|---|---|
| =, ≠, in | Element (Query) or Query |
| =, ≠ | Node value (mono-valued) |

**Table 2.** Operators and Operands for a Mono-Valued Non-Terminal Node

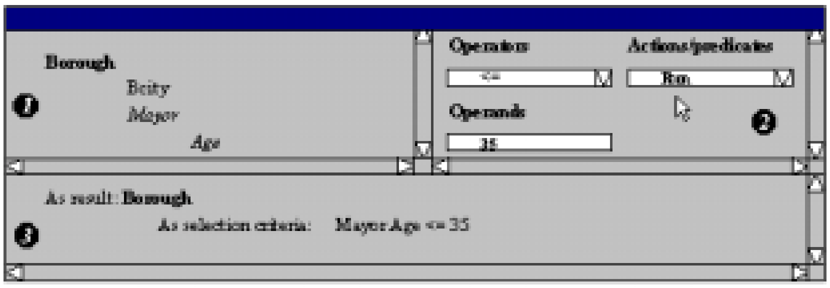
| Operators | Operands |
|---|---|
| =, ≠, <, >, ≥, ≤ | Query or Node value (multi-valued) |
| contains | Element(Query) |
| exists, forall, none | Query |

**Table 3.** Operators and Operands for a Multi-Valued Non-Terminal Node

The underlying idea is that operands may be the value of a node or a new query which can be embedded in the previous one.

## 3.2 The Querying Interface

The WDBQL querying interface is proposed via an HTML page. It contains three distinct interfaces which are the navigation, the contextual filter and the summary interfaces. The navigation interface allows the user to consult the database schema in accordance with the navigational process described before. This interface supports the user model. The contextual filter interface is used to define the result of the query and to express selections. It saves casual users from making inconsistent selections for one's query. Finally, the summary interface reminds the user the expressed query. We give the figure below as an example of the querying interface.



35

As result: Borough

As selection criteria: Mayor.Age <= 35

**Fig. 2.** The WDBQL Interface

The expressed query looks for all the boroughs having a young mayor, i.e. a mayor who is less than 35 years old. The navigation interface**1** shows the navigation from *Borough* to *Mayor*, and from *Mayor* to *Age*. The contextual filter interface**2** illustrates the expression of a selection. Operators and operands must be specified to express a predicate of selections. The user chooses an operator in the Operators contextual list. This list proposes all the operators available for the selected node (here *Age*). Then, the user enters the value of the operand. Finally, the user chooses in the Action/predicates list if he wants to execute the query, abort it, or go on the predicate specification using And or Or logical connectors. The summary interface**3** reminds the user the expressed query: the result is *Borough* and the selection criteria is *Age<=35*.

## 3.3 Query Example

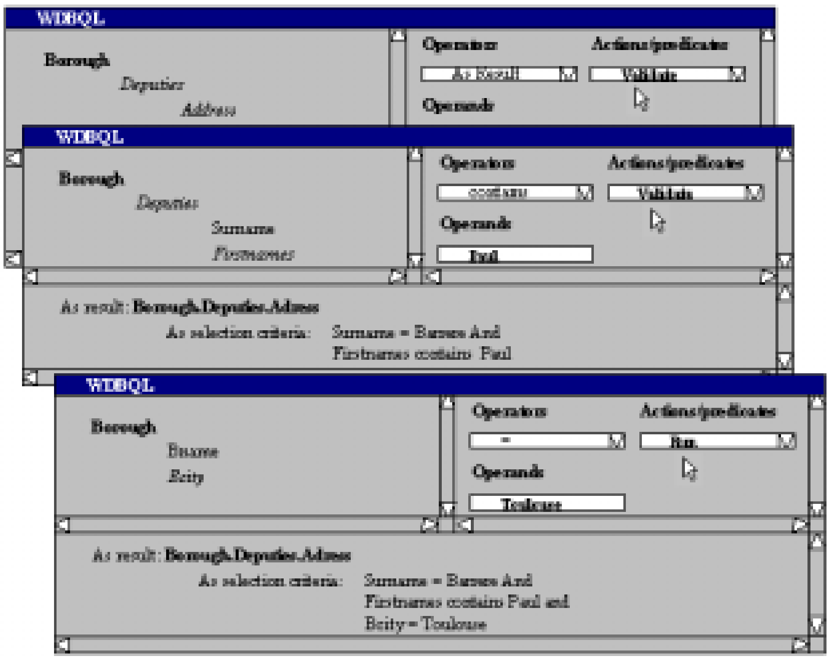
This section describes a WDBQL query. Proposed query illustrates the navigation through mono-valued or multi-valued non-terminal nodes. It also illustrates the expression of selections on mono-valued or multi-valued terminal nodes and on multi-valued non-terminal nodes.

The query given in figure 3 illustrates navigation through mono-valued and multi-valued non-terminal nodes (*Borough*, *Deputies*) and selection expressions on mono-valued terminal node (*Surname*, *Bcity*) or multi-valued terminal node (*Firstname*). Figure 3 defines the query result (*Address*) and the condition expressed on *Firstname*. As *Firstname* is a multi-valued node, the user specifies the *contains* operator to indicate that *Paul* is one of the first names. Figure 3 also illustrates the condition expression on *Bcity* and gives the final summary interface. The corresponding OQL query submitted to the O2 remote database is:

```
Select d.Address
From TheBorough b, b.Deputies d
Where b.Bcity = "Toulouse" and d.Surname = "Barrere" and "Paul" in d.Firstname
```

## 4 Comparison with Related Works

Recent development mixing Web and database technologies are numerous. Three classes of work related to information management on the Web are described in [11]: (i) modelling and querying the Web, (ii) information extraction and integration, and


WDBQL

Borough
Deputies
Address

Operators: As Result
Actions/predicates: Validate
Operands

WDBQL

Borough
Deputies
Surname
Firstnames

Operators: contains
Actions/predicates: Validate
Operands: Paul

As result: Borough.Deputies.Adress
As selection criteria: Surname = Barrere And
Firstnames contains Paul

WDBQL

Borough
Bname
Bcity

Operators: =
Actions/predicates: Run
Operands: Toulouse

As result: Borough.Deputies.Adress
As selection criteria: Surname = Barrere And
Firstnames contains Paul and
Bcity = Toulouse

**Fig.3.**: Address of the deputy Paul Barrere belonging to the borough of Toulouse

(iii) Web site construction and restructuring. The first class of works defines query languages for the Web which allows users to pose queries that go beyond the basic information retrieval paradigm supported by today's search engines. The second class of works specifies systems composed of wrappers and mediators for Web information integration. Finally, the third class of works applies techniques from databases to the process of building and maintaining Web sites.

Even if WDBQS mixes Web and database technologies, it has a quite different objective. Indeed, it defines WDBQL, a database query language for remote relational and object-oriented databases. WDBQL is mainly intended for casual users. WDBQL is comparable to the different visual query languages described in the literature. These languages were introduced to help casual users when querying databases.

Visual query languages described in the literature may be classified into form-based, graph-based, Web-based and multi-paradigm query languages. Form-based query languages permit the expression of queries from forms which represent the queried database schema. These forms visualise tables for relational databases (QBE [20]), nested tables for extended relational databases (VQL [19], QBEN [13]) and classes for object-oriented databases (OOQBE [17], PESTO [6], RYBE [15]). Graph-based query languages permit the expression of queries from graphs which describe the queried database schema. Some graph-based languages adopt a conceptual approach while others adopt an object-oriented one. In the conceptual approach, conceptual models are used to describe the queried database schema (QBD [2], SUPER[3], CONQUER [5]). In the object-oriented approach, object-oriented database models are used to describe the queried database schema. OHQL [1] is based on specific object-oriented models while QUIVER [9] is ODMG-compliant. Web-based query languages permit the expression of Web queries i.e. queries expressed from a Web interface using a navigational language. DataGuides [12] and WDBQL are examples of Web-based query languages. DataGuides is dedicated to semi-structured databases while WDBQL is dedicated to remote relational and object-oriented databases. WDBQL is ODMG-compliant. Finally, multi-paradigm query languages do not introduce a new paradigm for visual languages. They combine text-based, form-based and graph-based paradigms and supports automatic translation of queries between these paradigms. EVE [10] or ICI [7] are examples of multi-paradigm query languages. EVE combines text-based, form-based and graph- based query languages while ICI introduces, in addition with a graph-based query language, the icon notion for both data representation and manipulation.

Mechanisms used when query formulation not necessarily depend on the way the database schema is visualized. In some form-based query languages, users have to express selections, projections and joins (e.g. QBE or VQL). But in most of the form-based and graph-based query languages, query formulation only consists in condition expression because joins are already expressed. QBD, OOQBE, SUPER, OHQL, QBEN or RYBE define such a query formulation mechanism. For example, SUPER's users extract the querying graph form the database schema visualization graph. Then, they express the query conditions selecting on the query graph the corresponding attributes. A condition editor helps them with condition expression. Finally, they define the query result. In OHQL, the object-oriented database schema is represented as a set of nodes and links. Nodes correspond to the database classes while links correspond to the different relationships existing between these classes. OHQL's users define the scope of their queries selecting the participating nodes and links: active-joins between the corresponding classes are then established. Users express the query conditions and the query result respectively performing the selection and projection operations from the nodes of the graph. OHQL too proposes a condition editor to help users with condition expression. OHQL's originality lies in the fact that the query result is distributed into the different nodes of the query: users perform the display operation from each of these nodes to consult the query result. Another original aspect of OHQL lies in the active-joins. Active means that these joins are available until the end of the query; so, every selection operation performed on one of the nodes participating in the active-join has a consequence on the others nodes.

Just very few of these languages advocate a navigational query formulation process. Besides DataGuides and WDBQL, CONQUER and PESTO advocate a navigational mechanism for query formulation. CONQUER allows users to browse a conceptual schema to formulate queries. PESTO proposes the query-in-place mechanism for both browsing and querying object-oriented databases. However none of these systems are accessible via the Web. Finally, OHQL is not really navigational because active-joins do not correspond to navigation between nodes of the graph.

## 5 Conclusion

This paper has presented WDBQL, the WDBQS query language we developed to allow Local Communities casual users to easily query databases.

WDBQL belongs to a new category of visual query languages: the Web-based query language one. It advocates a Web-like query formulation mechanism. Such a mechanism has two main advantages: (i) it is really easy to use for casual users, and (ii) it hides the semantics of the performed operations. Moreover, WDBQL has a Client-Server architecture; it is accessible via the Web in an intranet context. Finally, WDBQL allows to query remote relational and object-oriented databases; it uses the ODMG model to describe the remote database schemas and defines a user model based on the notions of node, link and anchor to visualize these schemas. With WDBQL, Local Communities casual users have a unique and user-friendly tool accessible via the Web, to query remote relational and object-oriented databases.

WDBQL is original because it is a Web-based query language. To the best of our knowledge, only DataGuides [12], which allows the querying of semi-structured databases, belongs to that category. This paper continues the DataGuides works in the context of remote relational and object-oriented databases. It describes in a revealing way the WDBQL Web-type query language ( [12] only gives minimum information about the DataGuides query language).

However, WDBQL is less powerful than OQL. It needs to be extended in order to include:

- Nested queries. We have implemented an operator whose name is NewQuery, and it is possible to nest a query using the in operator. Predicate such as x in (Query) are available within WDBQL.
- Aggregation operators. Count, Max, Min, Avg and Sum are not available.
- Collection operators. Flatten, Element, Union, Intersect and Minus are neither available.

These operators need to be integrated to WDBQL in order to make it powerful and comparable to OQL. But, we are not sure we should implement them because we keep in mind that WDBQL is only intended for Local Communities casual users who express enough simple queries only including selection, projection or join operations. The user-friendly objective that WDBQL has for this kind of users must not be given up to the benefit of its power.

WDBQS, the system which implements WDBQL, is still an ongoing project. A first prototype is now available. O2, an object-oriented database, and ORACLE are accessible as WDBQL remote databases. We implemented two wrappers which respectively translates O2 database schemas and ORACLE database schemas as instances of the meta-schema. We used O2Web and ORACLE Web to display the WDBQL/OQL and WDBQL/SQL query results.

We now shall install WDBQS in the Local Communities site and shall set an evaluation of WDBQL up to Local Communities casual users.

## References

1 Andonoff E., Mendiboure C., Morin C., Rougier V., Zurfluh G., *OHQL: An Hypertext Approach for Manipulating Object-Oriented Database,* Information Processing and Management, Vol. 28, n°6, 1992

2 Angellacio M., T. Catarci T., G. Santucci, *Query By Diagram: a Fully Visual Query System,* Visual Languages and Computing, Vol. 1, n° 2, 1990

3 Auddino A., Dennebouy Y., Dupont Y., Fontana E., Spaccapietra S., Tari Z., *SUPER: Visual Interaction with an Object-Based Entity Relationship Model,* 11th International Conference on the Entity Relationship Approach, ER 92, Karlsruhe, Germany, October 1992

4 Belloir N., *Une Approche Navigationnelle pour l'Interrogation de Bases de Données via le Web,* DEA Informatique, Institut de Recherche en Informatique de Toulouse, Toulouse, France, Juin 1999

5 Bloesch A., Halpin T., *CONQUER: a Conceptual Query Language,* 15th International Conference on the Entity Relationship Approach, ER 96, Cottbus, Germany, October 1996

6 Carey M., Haas L., Maganty V. Williams J., *PESTO: An Integrated Query/Browser for Object Database,* 22nd International Conference on Very Large DataBases, VLDB 96, Mumbai, India, August 1996

7 Catarci T., Massari A., Santucci G., *Iconic and Diagrammatic Interfaces: an Integrated Approach,* International Workshop on Visual Languages, Kobe, Japan, June 1991

8 Cattell R., *ODMG-93: Le Standard des Bases de Données Objet,* International Thomson Publishing France, 1995

9 Chavda M., Wood P., *Towards an ODMG Compliant Visual Object Query Language,* 23rd International Conference on Very Large DataBase, VLDB 97, Athens, Greece, August 1997

10 Doan DK., Paton N., Kilgour A., *Design and User Testing of a Multi Paradigm Query Interface to an Object-Oriented Database*, Interacting with Computer, Vol. 7, n° 1, September 1995

11 Florescu D., Levy A., Meldenzon A., *Database Techniques for the World Wild Web: A Survey*, SIGMOD Record, Vol. 27, n° 3, September 1998

12 Goldman R., Widom J., *DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases*, 23rd International Conference on Very Large DataBase, VLDB 97, Athens, Greece, August 1997

13 Lorentzos N., Dondis K., *Query By Example for Nested Tables,* 9th International Workshop on Databases and Expert Systems Applications, Vienna, Austria, August 1998

14 Mendelzon A., Mihaila G., Milo T., *Querying the World Wide Web,* Digital Libraries, Vol. 1, n° 1, April 1997

15 Sallaberry C., Bessagnet MN., Kriaa H., *RYBE: a Tabular Interface for Querying a Multi-Database System,* 8th International Conference on Management of Data, COMAD 97, Madras, India, December 1997

16 Silberschatz A., Stonebraker M., Ullman J., *Database Research: Achievements and Opportunities Into the 21st Century,* SIGMOD Record, Vol. 25, n°1, March 1996

17 Staens F., Tarentino L., Tiems A., *A Graphical Query Language for Object-Oriented Databases*. International Workshop on Visual Languages, Kobe, Japan, June 1991.

18 Stonebraker M., *Are we Working on the Right Problems?,* 25th International Conference On Management of Data, SIGMOD 98, Seattle, Washington, USA, June 1998

19 Vadaparty K., Aslandigan Y., Ozsoyoglu G., *Towards a Unified Visual Database Access,* 20th International Conference On Management of Data, SIGMOD 93, Washington, DC, USA, May 1993

20 Zloof M., *Query By Example: A Database Language,* IBM System Journal, Vol. 16, n°4, 1977