WDBQS: A Unified Access to Distant Databases Via a Simple Web-Tool

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

[1] Université de Pau et des Pays de l'Adour, UFR Droit & IAE, 64000 Pau, France,
christian.sallaberry@univ-pau.fr
[2] IRIT/UT1, Université Toulouse 1, 1 place Anatole France, 31042 Toulouse Cedex, France
ando@irit.fr

Abstract. Within organisations, Information Systems are characterised by heterogeneity. The main management tools are different Relational and Object Oriented DataBase Management Systems. Therefore, variety, complexity and lack of integration make it difficult for casual users to query them. This paper presents WDBQS (Web DataBase Query System); a Web-interface for querying distant Relational and Object Oriented DataBases. This system, dedicated to casual users, proposes simple Web mechanisms to navigate through database schemas, build queries and display results.

1 Introduction

We have collaborated with Local Communities (LC) to produce WDBQS (Web DataBase Query System), an ongoing project that aims to develop a tool intended for casual users for distant relational and object-oriented database querying (RDB and OODB). LC handle a set of Information Systems managed by Relational DBMS and Object-Oriented DBMS (RDBMS and OODBMS). These IS have been developed and managed independently; they are based on different software technologies and run on various hardware platforms. WDBQS offers to the LC users a common language to query the databases implementing the different IS. There is no need to develop approaches like DataWareHouse or MultiDataBase. Indeed, LC users just need a tool to query, one after another, the different distant and independent RDB and OODB.

So, WDBQS is a relational and object-oriented database query system. Its main characteristics are:

- ODMG data model,
- common access to distant RDB and OODB,
- accessible via the Web in an intranet architecture and advocates a navigational query formulation mechanism,
- reuse of data and systems in order to take advantage of existing resources and be able to rely on proven technologies (RDBMS and OODBMS),
- extendibility in order to access any new database, either achieved through the corresponding DBMS drivers already integrated in WDBQS or through the integration of a new DBMS drivers in WDBQS,

 specific interfaces so as to manage distant databases accessed by WDBQS, define queries and display results.

WDBQS makes accessible heterogeneous DBMS via a unique platform. It is based on the ODMG model and proposes a Web-like query language, called WDBQL (Web DataBase Query Language). WDBQL is intended for casual users i.e. users who do not have any knowledge in computer-science and consequently in databases and database query languages such as SQL and OQL. WDBQL proposes a Web-interface and implements a Web-like i.e. navigational mechanism to formulate queries. Such a mechanism is easy to use for casual users and really helps them when query formulation.

The remainder of the paper is organised as follows. Section 2 presents the WDBQS architecture. Section 3 introduces the WDBQS models and a running example. Section 4 describes the WDBQL querying principles and presents the WDBQL interface. It also gives a comparison with related works. Section 5 concludes the paper.

2 WDBQS architecture

WDBQS proposes a client-server architecture described in figure 1. It is composed of WDBQL, WDBQS Server (WDBQSS), Host DataBase Server (HDBS) and Remote DataBase Server (RDBS).

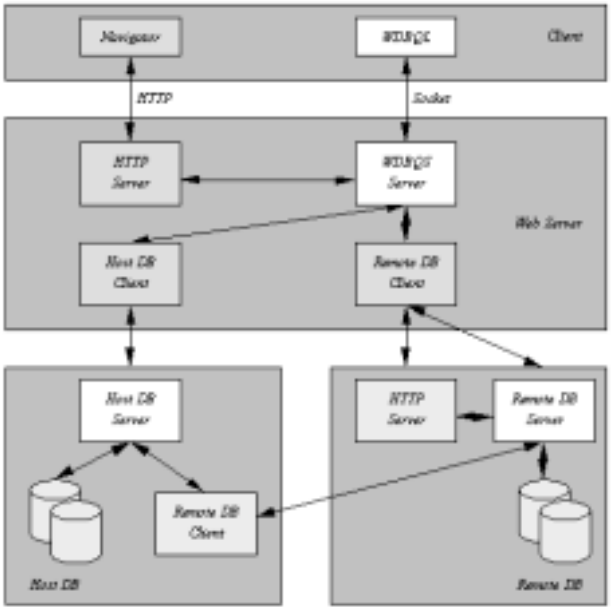


Fig. 1. WDBQS architecture

HDBS stores in a special database called the Host DataBase, a meta-schema which describes the schemas of all the distant RDB and OODB. The model of this Host DataBase is the ODMG one [9]. The distant RDB and OODB schemas are stored as instances of the meta-schema. HDBS uses a schema extractor and an ODMG wrapper to first extract the schema of a distant database and to then convert it as instances of the meta-schema. There are as many schema extractors as distant databases and as many wrappers as distant database distinct models. The interest of such an approach is that the number of accessible distant databases is easily extendable.

RDBS is the server of the distant database. It executes queries and returns the corresponding result. Two situations are possible: RDBS integrates a Web module (e.g. O2Web, OracleWeb) or not. If the Web module exists, RDBS displays the results by itself in a new client Web navigator window. If not, RDBS returns a set of data which are converted by the WDBQL Data Translator into the WDBQL visual format. RDBS also notifies HDBS of any database schema updates. HDBS then starts again schema extraction and conversion.

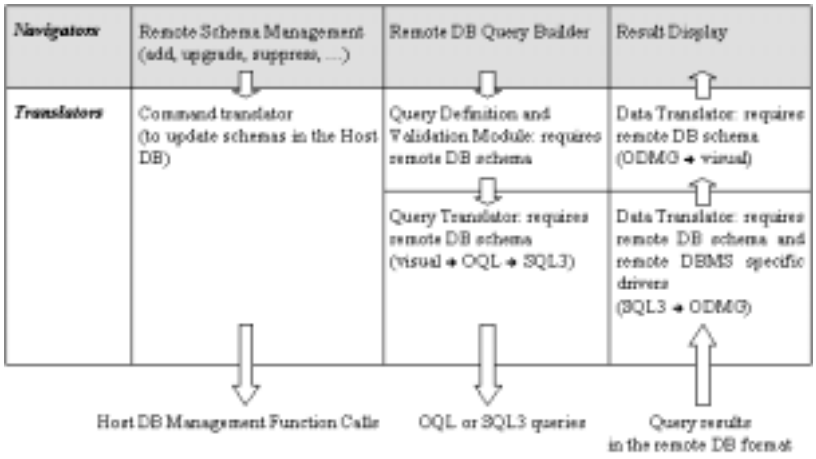


Fig. 2. WDBQS: navigators and translators

WDBQL offers two main navigators: one for remote database schema management and another for remote database querying. Each navigator translates users visual queries respectively into database management function calls or OQL (or SQL3) queries for WDBQSS. A third navigator possibly displays results via an HTML wrapper. The WDBQL querying navigator uses the Query Definition and Validation module to display the schema of the queried remote database and to help the user when query formulation. The WDBQL querying navigator also uses the Query Translator module to translate WDBQL queries into OQL ones. The query Translator module principally uses the translation algorithms presented in [10].

WDBQSS manages communication through all the modules of the system. The working of the system is the following (cf. Fig. 3.).

One user connects with WDBQSS via the HTTP protocol. WDBQSS sends back an HTML page integrating WDBQL to the client (Java applets). WDBQL displays the list of all the remote databases which can be queried. The user selects a database. The WDBQSS gets L in a text form d displays the c his client mach to be exe- cutec it has its own BQL Data Trans

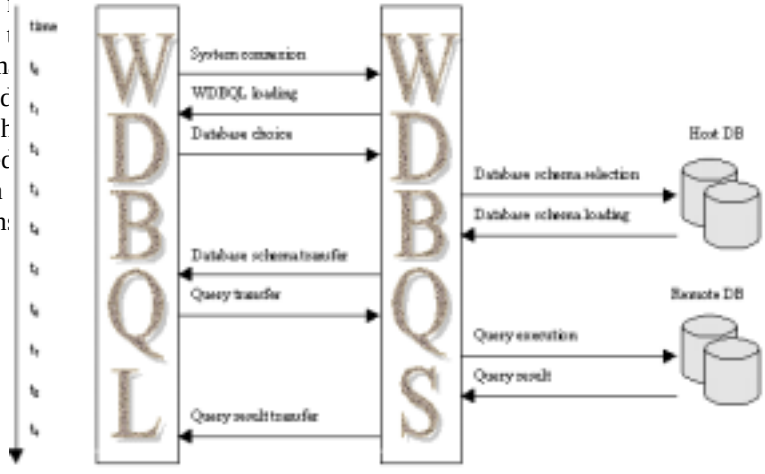


Fig. 3. Client (WDBQL) / Server (WDBQSS) communications

WDBQS architecture can be compared to that of GEM/WeBUSE [15]. The GEM/WeBUSE context is quite the same as the WDBQS one. The main difference is that GEM/WeBUSE proposes a remote GEM server for each distant database system. There is no central host database and HDBS but there are many remote meta-schema descriptors. This approach is not suitable in WDBQS as it makes too many communications (with the remote DBMS) when formulating queries. Moreover, in our context we are not allowed to use disk space within remote database servers.

3 The WDBQS models

We give here the *County Council* database 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 Elected inherits Person type tuple
( Arrival : Date,
  Grade : string )
end;
```

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

WDBQS supports two levels of data representation: the database level and the user level. We now present the two level of data representation.

3.1 The database level

The database level of WDBQS stores the description of the different remote RDB and OODB schemas as instances of a meta-schema. This meta-schema is described using the standard ODMG-93 [9]. We choose the ODMG model because it generalises the RDB and OODB concepts. Besides, it is chosen as a pivot model in several federated DBMS.

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, StructuredTypeProperty, Operations and Relationship ones. The meta-schema and its corresponding types are detailed in [5].

3.2 The user level

The user level allows interpretation of the meta-schema within a more simple representation model. The latter is described using a graph model based on the node, link and anchor notions. Such a model obviously better suits casual users.

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 the following table:

Database Model	**User Model**
Base	Non-terminal node
Type	Non-terminal node
Property	
- Mono-valued	Terminal node
- Multi-valued	Terminal node
- Structured	Non-terminal node
Operation	Terminal node
Relationship	
- Mono-valued	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. 4. Part of the County Council database described with the user model

4 WDBQL: a query language intended for casual users

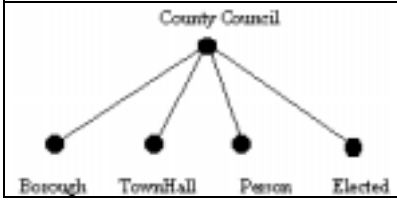
WDBQL is a Web-based query language intended for casual users. It allows the expression of queries from a Web interface using a navigational language.

4.1 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 WDBQL query is a set of actions on nodes of the user model. We distinguish two types of actions which are navigation and selection.

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 below 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: Then the user can go on its navigation selecting for example the non-terminal node Borough. All the nodes linked to the node Borough are then visualised.

User Model	**Result of an Action of Navigation**
	County Council Borough TownHall Person Elected The selection of the node *County Council* implies the visualization of the linked nodes

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 below the table for multi-valued non-terminal nodes.

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

Table 2. Operators and operands for a multi-valued non-terminal node

4.2 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 is a guide to the casual user as it saves him from making inconsistent selections for one's query. Finally, the summary interface reminds the user the expressed query. We give the figure 5 as an example of the querying 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*.

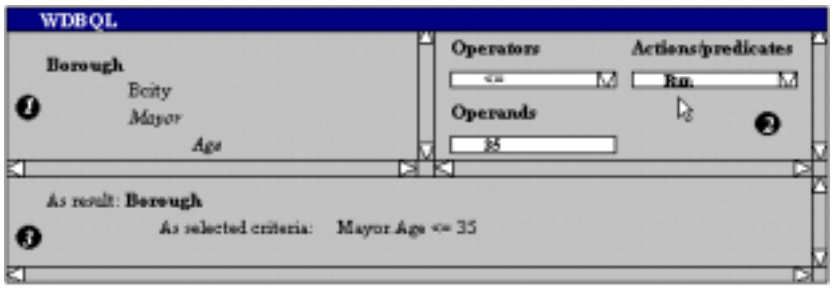


Fig. 5. The WDBQL interface

4.3 Comparison with related works

Recent development mixing Web and database technologies are numerous. We distinguish three classes of work related to information management on the Web: (i) modelling and querying the Web, (ii) information extraction and integration, and (iii) Web site construction and restructuring. Even if WDBQS mixes Web and database technologies, it has a quite different objective which is to define a Web-interface for querying distant RDB and OODB. WDBQS is mainly intended for casual users. The language it proposes, called WDBQL is comparable to the different visual query

languages described in the literature. These query languages may be classified into form-based, graph-based, icon-based and Web-based query languages.

Form-based query languages allow the expression of queries from forms which represent the queried database schema. Form-based query languages are proposed for RDB (QBE [18], Access [1]), for extended-RDB (VQL [17], QBEN [13]) and for OODB (OOQBE [16], GOQL [12], PESTO [7], RYBE [13]). 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 (QBD [3], CONQUER [6]) while others adopt an object-oriented one (OHQL [2], QUIVER [10]). Icon-based query languages use visual metaphors to both represent the queried database and the query language operators. ICI [8] and VISTA [4] are examples of icon-based query languages. Web-based query languages permit the expression of Web queries i.e. queries expressed from a Web interface using a navigational language. DataGuides [11] and WDBQL are examples of Web-based query languages. DataGuides is dedicated to semi-structured databases while WDBQL is dedicated to remote RDB and OODB. WDBQL is ODMG-compliant.

5 Conclusion and future works

This paper has presented WDBQS, a system we developed to allow LC casual users to easily query with a common language distant RDBs and OODBs. WDBQS has a client-server architecture ; it is accessible via the Web in an intranet context. WDBQS proposes WDBQL, a query language well-suited for LC casual users. WDBQL is a Web-based query languages. That means it advocates a Web-like query formulation mechanism. Such a mechanism is really easy to use for casual users.

If we compare WDBQS to other similar systems [6, 7, 11, 15], we can mention some of the main differences:

- WDBQS allows to query with a common language distant RDBs and OODBs. So, it is different from [11] which is dedicated to semi-structured databases.
- WDBQS is ODMG-compliant.
- WDBQS proposes a Web-based query language i.e. a language which is accessible via the Web and which advocates Web navigational mechanisms for query building. So, it is different from [6] and [7] which propose a navigational query language but which do not have a Web-interface.

However, WDBQS has some drawbacks. On the one hand, WDBQL is less powerful than OQL. It needs to be extended in order to include (i) 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 -, (ii) aggregation operators -Count, Max, Min, Avg and Sum are not yet available- and (iii) 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. On the other hand, we need to set evaluations of WDBQL up to LC casual users.

WDBQS is still an ongoing project. A first prototype is now available. Only the Oracle RDB and the O2 OODB are accessible as remote databases. We implemented two wrappers which respectively translate Oracle and O2 database schemas as instances of the meta-schema. We used OracleWeb and O2Web to display the WDBQL query results. We now implement the wrapper whose aim is to translate a query result in a remote database format into the WDBQL format. We also implement Jasmine as a remote database for WDBQS.

References

1. Access: Microsoft home page: http://home.Microsoft.com
2. 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
3. Angellacio, M., Catarci, T., Santucci, G.: Query By Diagram: a Fully Visual Query System. Visual Languages and Computing, Vol. 1, n° 2, 1990
4. Belieres, B., Trepied, C.: New Metaphors for a Visual Query Language. DEXA'96, International Workshop on Databases and Expert Systems Applications, Zurich, Switzerland, September 1996
5. Belloir, N.: Une approche navigationnelle pour l'interrogation de base de données via le Web. Mémoire de DEA, Institut de Recherche en Informatique de Toulouse, Toulouse, France, September 1999
6. Bloesch, A., Halpin, T.: CONQUER: a Conceptual Query Language. ER'96, 15th International Conference on the EntityRelationship Approach, Cottbus, Germany, October 1996
7. Carey, M., Haas, L., Maganty, V., Williams, J.: PESTO: An Integrated Query/Browser for Object Database. VLDB'96, 22nd International Conference on Large Data Base, Mumbai, India, August 1996
8. Catarci, T., Massari, A., Santucci, G.: Iconic and Diagrammatic Interfaces: an Integrated Approach. International Workshop on Visual Languages. Kobe, Japan, June 1991
9. Cattell, R.G.: ODMG-93: Le Standard des Bases de Données Objet. International Thomson Publishing, France, 1995
10. Chavda, M., Wood, P.: Towards an ODMG Compliant Visual Object Query Language. VLDB'97, 23rd International Conference on Very Large DataBase, Athens, Greece, August 1997
11. Goldman, R., Widom, J.: DATAGUIDES: Enabling Query Formulation and Optimisation in Semi-structured Databases. VLDB'97, 23rd International Conference on Very Large DataBase, Athens, Greece, August 1997
12. Keramopoulos, E., Pouyioutas, P., Sadler, C.: GOQL: a Graphical Query Language for Object-Oriented Database Systems. BIWIT'97, 3rd Basque International Workshop on Information Technology, Biarritz, France, July 1997
13. Lorentzos, N.A., Dondos, K.A.: Query By Example for Nested Tables. DEXA'98, International Workshop on Databases and Expert Systems Applications, Vienna, Austria, August 1998

14. Sallaberry, C., Bessagnet, M.N., Kriaa, H.: RYBE: a tabular interface for querying a multi-database system. COMAD'97, 8th International Conference on Management of Data, Madras, India. December 1997
15. Sivadas, M., Fernandez, G.,: GeM and WeBUSE: Towards a WWW-Database Interface. Workshop on Co-ordination Technology for Collaborative Applications, Asian'96, 2nd Asian Computer Science Conference, Singapore, 1996
16. Staens, F., Tarentino, L., Tiems, A.: A Graphical Query Language for Object-Oriented Databases. International Workshop on Visual Languages, Kobe, Japan, June 1991
17. Vadaparty, K., Aslandignan, Y.A., Ozsoyoglu, G.: Towards a Unified Visual Database Access. SIGMOD'93, 20th ACM SIGMOD International Conference On Management of Data, Washington, USA, 1993
18. Zloof, M.: Query By Example: A Database Language. IBM System Journal, Vol. 16, n°4, 1977