# A Data Browsing from Various Sources Driven by the User's Data Models

Guntis Arnicans, Girts Karnitis

University of Latvia, Raina blvd. 19, Riga, Latvia
{Guntis.Arnicans, Girts.Karnitis}@lu.lv

**Abstract.** There are many various data extracting and browsing tools available that provide working with a physical database (relational, object-oriented, RDF, etc.). We propose to drive the browser by a logical data model that serves as a bridge between browser and physical data sources. This approach allows a creating so many different logical data models with integrated data as we like. Models can be designed for individual users, groups of user or specific business task. User can customize this model for further data integrating and better traversing and displaying. Data extracting and modifying wrappers can be created automatically for physical databases or created by developer if data are available via third part services. The browser provides browsing and traversing through data like internet browser allows user to retrieve, present, and traverse information resources on the World Wide Web. The browser contains predefined strategies for an automatic integration of linked entities into large semantically logical entities.

**Keywords:** data browsing and traversing, data integration, logical data model, user-oriented data integration

## 1 Introduction

There are many databases and data sources available for users. Most of users receive data or information from these data sources via specialized information systems. If a necessary visual user interface is not created then the data source is useful mostly only for users with appropriate IT skills. For instance, user has to use a database management tool to open a table or write an SQL statement to obtain related data from various tables, or call a Web service that returns data in XML format and inspect data by XML viewer.

Our goal is to create a methodology and tools that provide data browsing and inspection without of creating a traditional information system. Moreover data are extracted from number of data sources and integrated together. The interaction with a user is organized similarly to a browsing of Web pages. Let us call data browsing tool as a *browser* in this paper.

There are no good tools and methodologies that satisfy our goal regardless of the first papers in this field were written and the prototypes were created more than 30

years ago. One of the first works was paper about QBE [1] – language used today in many graphical front-ends for databases, for example *Microsoft Access* and *Microsoft SQL Server*.

A lot of visual browsers were developed for Object-oriented databases (OODB). Good examples for the OODB browsing include PESTO [2], QBB [3], IVORY [4]. In PESTO approach data at first are queried and then visualized. User defines a main class. The main class' objects are visualized and related objects can be visualized too. The greatest problem – user can see only one object from main class at any time. A basic idea behind the QBB is to present object hierarchy similarly to folder hierarchy in a file browser. Different restrictions can be defined via visual user interface in the QBE. The main advantage of the QBE – it is relationally complete language. The IVORY is somehow like the PRESTO. The IVORY allows users to browse OODB and then visually specify query from database. In IVORY users can browse objects in one class and by clicking on link button see related objects into another class. Users also can create data query by dragging and dropping classes and objects into special frame. The IVORY is ODMG-93 object model compliant.

Important work is done in the field of visualization of RDF data. Some of them are Longwell [5], Tabulator [6]. The Longwell is a part of MIT SIMILE project. The Tabulator is an RDF data browser created by T Berners-Lee and co-authors. An intention of Tabulator authors is to investigate new possibilities in usage of RDF data and new user metaphors.

Authors of [7] present approach to automatically generate GUI for browsing of RDF data based on existing RDF data graphs. Their main idea is a support of rapid creating a prototype for data browsing.

Many of browsers mentioned above have one or both of two main problems. First problem – browsers are created in such a way, that only small amount of data can be seen on a screen and the getting of another data portion requires some action. Second problem – user can see only data that he is queried, and to see related data he must make another query. OODB and RDF browsers usually have first problem, but Relational data browsers – second one.

In our approach we try to show minimum of technical information and a lot of data and allow user to move from some data to the related data with minimum activities. We have created browser that browses and traverses data according to logical data model. This model initially is created automatically at first connection to the source database. Later user can customize this model. If the data are available through third part services then data model has to be tied with these services. Browser contains various strategies for data integrating and displaying.

## 2 Traversing and Browsing of Relational Databases

Let us take a look at a simple situation – one in which the data are stored in a sufficiently popular relational database and we can select data and metadata from the database. For instance, let us use a database with physical data model that are shown, in simplified form, in Fig. 1. The database contains information about courses, instructors, students, and study results.

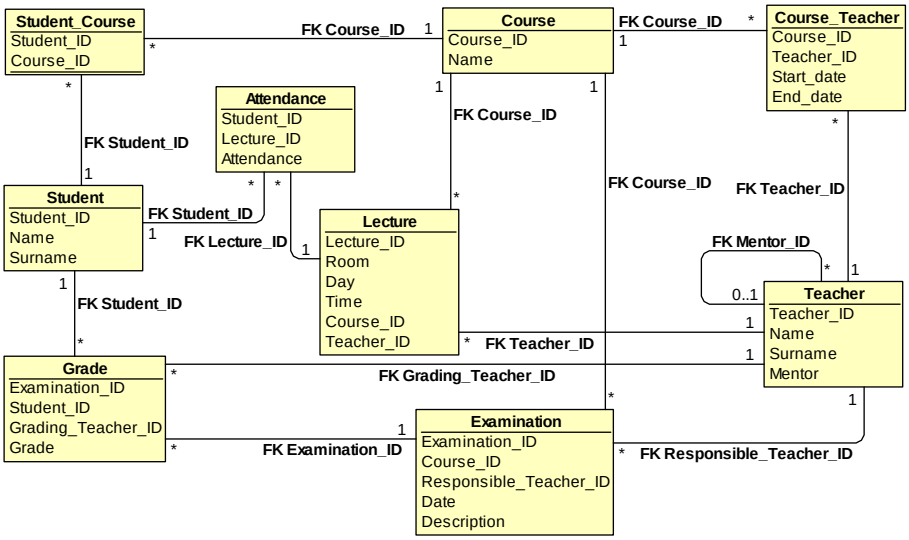


**Fig. 1.** A sample of data model in the relational database

Let us assume that we are looking at the table *Grade* or *Teacher* with browser that are part of the most relational database management systems, for instance, *SQL Server Management Studio* included with *Microsoft SQL Server 2008* (Fig. 2). From table Grade we see only the grades that have been given, as well as identifiers which identify the student, the instructor and the examination. We are interested in more precise information from other related tables. Traditionally, this would involve one major SQL query, but that bears with it the risk of errors. The desired result can be achieved gradually and via a number of smaller queries.

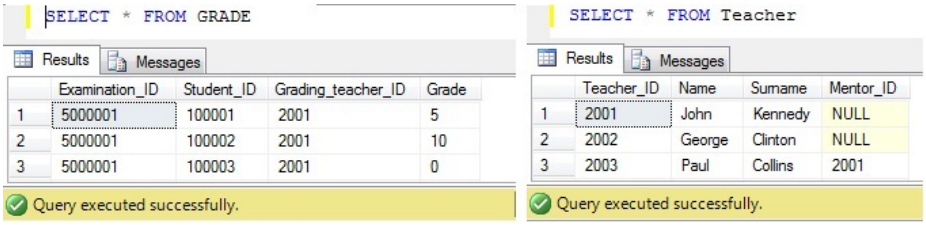


**Fig. 2.** Traditional table browsing (table Grade and table Teacher)

To form information from various tables we use a *Browsing View* (see more for this concept in [8, 9, 10, 11]). This is a mean for definition and obtaining a sub-graph, fixing the entity and linking other entities on the basis of specific principles. The essence of our approach is shown in Fig. 3, where you can see a browsing of 1) the chosen table *Teacher* (left part of picture); 2) instances of the table *Teacher* with related information (upper right part of picture); 3) defined relations as tabs (middle right part of picture); 4) details of the selected table *Lecture* with related information

to it (bottom right part of picture). The tool allows defining a virtual column that represents entity (for instance, "*Name & Surname*" for table *Teacher*).

Detailed data of lectures are shown for current teacher *John Kennedy*. By clicking on tab Grade all grades assigned by this teacher are shown (Fig. 4).

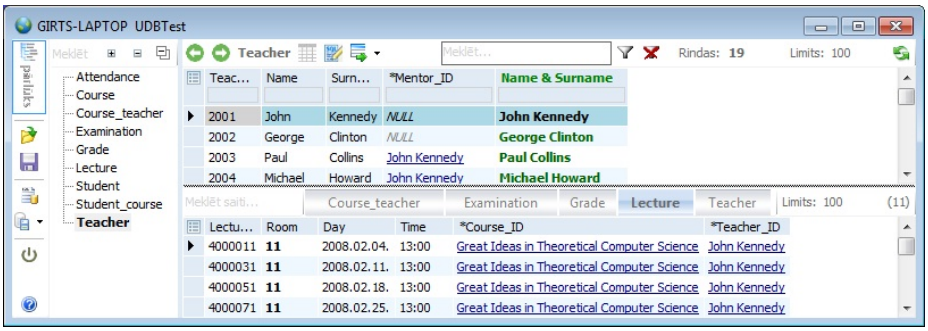


**Fig. 3.** Teachers data and Lectures for the single teacher

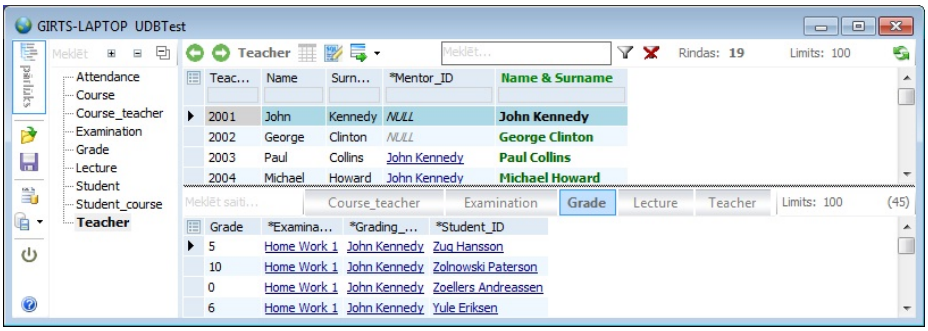


**Fig. 4.** Teachers data and Grades by the single teacher

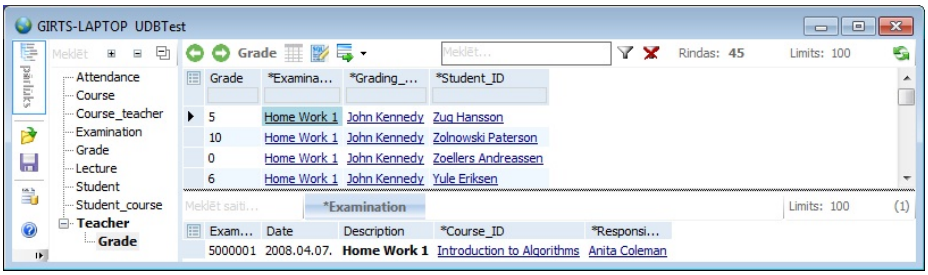


**Fig. 5.** Single teacher's Grades and information about the graded Examination

The traversing model is an inviolable part of the *Database Browsing Model* [8], which means a transition from the fixed central instance to an instance of linked entity applying actual *Browsing View*, similarly to the way used in data-intensive Web

applications. These applications involve a navigation model which ensures the transit from one information unit to another. The navigation model is closely linked to the data model which specifies the units of information.

For instance, in Fig. 4 we can now shift our attention to the entity *Grade*. By double clicking on the tab *Grade* the entity *Grade* immediately becomes the central one (Fig. 5). Upper right part contains exactly the same instances as in bottom right part of Fig. 4. By clicking on any link in Fig. 4 we select one instance of entity, and after the transition upper right part contains only this instance.

Data browsing and traversing is simple like browsing internet pages. User does not write any code or request statements. Without tool similar to the proposed browser the task is more complex. For instance, to obtain data displayed in upper right part of Fig. 5 we need to correctly write following SQL statement (*Microsoft SQL Server 2008*):

```
Select top 100
      g.Examination_ID, g.Student_ID, g.Grading_teacher_ID, g.Grade,
   (Select top 1 t.Description from Examination t
      where t.Examination_ID=g.Examination_ID)  as xExamination_ID,
   (Select top 1 coalesce(cast("Name" as varchar(250)),'') + ' ' +
           coalesce(cast("Surname" as varchar(250)),'') as xFormula_
      from Teacher t
      where t.Teacher_ID=g.Grading_teacher_ID)  as xGrading_teacher_ID,
   (Select top 1 coalesce(cast("Name" as varchar(250)),'') + ' ' +
           coalesce(cast("Surname" as varchar(250)),'') as xFormula_
      from Student s
      where s.Student_ID=g.Student_ID)  as xStudent_ID
   from Grade g
   where (g.Grading_teacher_ID in (2001))
```

## 3 Extracting Data from Data Sources and Integrating Them

If we have got the rights make a connection to database and make direct requests, then *Database Browsing Model* usually contains needed information of tables, attributes, relations. User refines the model by excluding unnecessary tables, attributes, relations, and by adding additional relations.

However the direct access to database is not always available. Third part legal entity usually offer access to data via services. Such case study is described in [10]. By analyzing offered services we construct a data model. Instead of classical relation between two entities we use link that correspond to some service. Link states if we know data of source entity than we can obtain some attributes of destination entity. We call these links the *functions*.

Function descriptions are stored in the browser's metadata base (Fig. 6). Each function has defined entry data, i.e., information about which entities must be presented by which fields at the entry, as well as information about whether they are or are not mandatory. Each function also has defined exit fields, i.e., which entities are returned by which fields as a result of this function. A wrapper is also defined for each function, which physically makes the request to the data source.

On the basis of their own experience, the authors would recommend the establishment of functions which have entry and exit fields from a single entity, because that makes it easier to establish the wrappers and the data model.

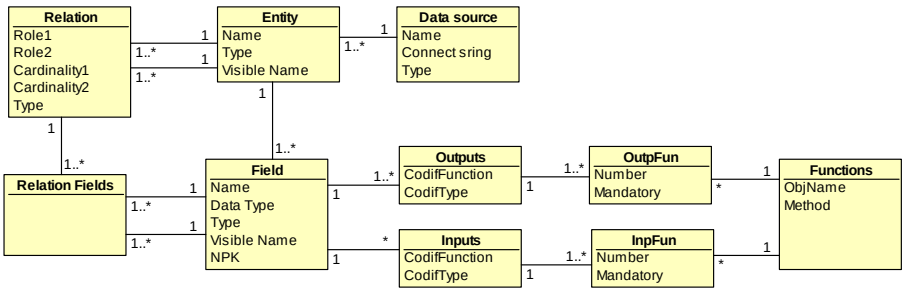


**Fig. 6.** A fragment of browser's metadata model

Data sources contain instances of objects. Each object can be viewed as a collection of attributes. User can view these attributes in the same collections as in the data sources. A sample is relational database browsing when we can make connection to database directly. However user may create his own data model based on attributes that he can receive from the data source. Generally we create a global dictionary of all available attributes. From these attributes we construct all necessary logical entities, links between these entities, and obtain basic elements for the *Database Browsing Model*.

Although attributes in global dictionary come from various data sources via different data accessing services nevertheless all attributes are equal in "their rights" to be composed into new logical entities. Integration of data is performed by several ways: 1) user creates his own collections of attributes (logical entities); 2) user creates his own relations between logical entities; 3) user selects one from number of predefined data integration strategies expressed by Browsing Views.; 4) user can create global link entity that serves as a super class for semantically related logical entities. Sample for fourth way of integration is given in Fig. 7. For instance, we automatically obtain information what books are taken by the particular teacher or student.

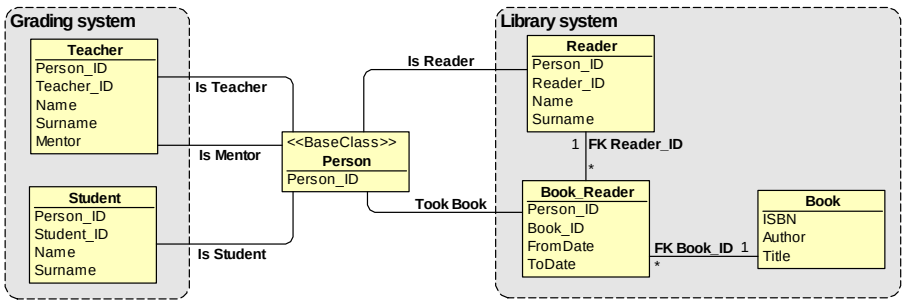


**Fig. 7.** Partial data model of two systems and logical class linking entities from both systems

## 4 Exploiting Number of User's Data Models

Integrated data browsing is partly verified by number of browser prototypes and domain specific applications. In this paper we have included screenshots of our tool *DIGIBrowser* that is exploited in some companies. Browser mostly is used for information systems maintenance discovering damaged data and averting the damage.

DIGIBrowser at this moment does not include all ideas that were verified by all prototypes. It can connect to any relational database stored in *Oracle Database*, *Microsoft SQL Server*, *MySQL*, *PostgreSQL*, and a browsing and traversing can be performed immediately. It is possible to connect to *Virtuoso Universal Server* (RDF data browsing) and XML file (browsing XML data like object-oriented or relational database), but this possibility is not verified in real IT projects.

DIGIBrowser allows filtering data (instances of logical data entities) and sorting similar data that are showed in table format, defining of additional information fields (aggregation of selected data), modifying of values and storing them into database, export selected data, see the SQL query script that extract data from database, etc.

A browser works only with the *Database Browsing Model* (DBM) that contains logical data model, *Browsing Transition Graph*, predefined *Browsing Views*, predefined wrappers to data sources, and descriptions for data visualization and customization. Such approach allows to create many DBMs and to choose the most appropriate one while browsing: 1) for an individual user (person); 2) for a user group; 3) for a particular business task; 4) for a specific conceptual or detailed level of presentation. For any of these groups many different models can be designed.

We can consider that we have got many different DBMs and can manage rights to use them like any other IT resource. In practice DBMs are stored in database, and rights can be managed with standard means that provides the database. Access to instances of data for the individual or the group can be regulated by each data source.

Browsing and traversing of data are performed step by step where basic steps are 1) transition from fixed entity to linked entity; 2) choosing another Browsing View; 3) data filtering; 4) choosing of visual presentation. An interesting future opportunity is switching between different DBMs while browsing (this possibility is not realized and verified in practice yet). User wants sometimes to see data at conceptual level to get overview, but sometimes he wants gradually switch to the more detailed model.

## 5 Conclusions and Future Work

Among number of traditional approaches how to integrate data and deliver it to user we offer to exploit specific data browser. Our browser provides data browsing and traversing like internet browser allows user to retrieve, present, and traverse information resources on the World Wide Web. The browser works according to logical data model that are assigned for browsing. Only logical data model determines what we get from data sources and how obtained data are integrated together. It is possible to create as many different logical data models as we need. It is allowed to customize data model, presentation, and choose data integration strategy by selecting appropriate *Browsing View*.

The created tool DIGIBrowser allows quickly and easy work with relational databases. The next step is to find a method how user can easily define and configure third part data retrieving services, and bind them to the logical data model.

Very promising chance to improve data browsing is switching between data models with different level of conceptualization. For simplest cases technically it is possible to create such solution, but an additional research is required to find more universal solution.

## Acknowledgement

The work is supported by a European Social Fund Project No. 2009/0216/1DP/1.1.1.2.0/09 /APIA/VIAA/044.

## References

1. Zloof, M.: Query By Example, IBM System Journal 16, Dec. (1977)
2. Carey, M., Haas, L., Magnaty, V., Williams, J.: PESTO: An Integrated Query / Browser for Object Databases. Proceedings of the 22nd VLDB Conference Mumbai (Bombay) (1996)
3. Polyviou, S., Samaras, G., Evripidou, P.: A Relationally Complete Visual Query Language for Heterogeneous Data Sources and Pervasive Querying. Proceedings of 21st International Conference on Data Engineering (2005)
4. Chang, S., Kim, H.: An Integrated Browsing and Querying System for ODMG-Compliant Object Databases (1998)
5. SIMILE: Longwell RDF Browser. [Online].http://simile.mit.edu/longwell/
6. Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R.: Tabulator: Exploring and Analyzing linked data on the Semantic Web. In 3rd International Semantic Web User Interaction Workshop in International Semantic Web Conference, Athens, Georgia (2006)
7. Pazenzia, M., Scarpato, N., Stellato, A.: Semi-Automatic Generation of GUIs for RDF Browsing. In proceedings of IV'2010, pp. 267—272 (2010)
8. Arnicans, G., Karnitis, G.: Prototype for Traversing and Browsing Related Data in a Relation Database, In: Bārzdiņš (ed.), Scientific Papers University of Latvia, Vol 756, Computer Science and Information Technologies, University of Latvia, pp. 59--74 (2010)
9. Arnicans, G.: Application generation for the simple database browser based on the ER diagram. In Jānis Bārzdiņš, editor, Databases and Information Systems, Proceedings of the Third International Baltic Workshop, Volume 1, pp. 198—209, Rīga, (1998)
10. Arnicans, G., Karnitis, G.: Heterogeneous Database Browsing in WWW Based on Meta Model of Data Sources. In Janis Barzdins and Albertas Caplinskas, editors, Databases and Information Systems, Fourth International Baltic Workshop, BalticDB&IS 2000 Vilnius, Lithuania, Selected Papers, pp. 167--178. Kluwer Academic Publishers (2000)
11. Arnicans, G., Karnitis, G.: Intelligent Integration of Information From Semi-Structured Web Data Sources on the Basis of Ontology and Meta-Models. In Proceedings of the 2006 Seventh International Baltic Conference on Databases and Information Systems, BalticDB&IS'2006, pages 177--186, (2006)