BUILDING AN EFFECTIVE USER-INTERFACE FOR MEDICAL DATABASES

P. OLAH[1], M. MARUSTERI[2], MONICA SUCIU[3], H. SUCIU[4], DANIELA DOBRU[5]

[1]PhD candidate University of Medicine and Pharmacy Tg. Mureş, [2,3,4,5]University of Medicine and Pharmacy Tg. Mureş

Keywords: *medical databases, healthcare information systems, user interface*

Abstract: *Healthcare Information Systems (HIS) generally rely on large databases that store medical data. Apart from the specific issues related to the design of the data structures, these systems present a challenge to the developer regarding the architecture of the external level of the database, closely related to the building of the user interface. This paper discusses three specific issues related to the user interface of a medical database (The Temporal Property, The Allocation Process and The Open Attribute List Specialization) using a data-oriented approach for the external level. The approach is mostly based on data views, constructed on the server side and on the client application side. A small number of stored procedures however are necessary to implement the full functionality of the system. In our approach, most of the complexity of these processes is embedded in the queries that are run against the database. These queries can often be complex and time consuming. Two major issues arise from here: first, complexity can lead to inconsistent presentation of the same data (we are considering large information systems) and second, large queries can severely affect the overall performance. These are the factors which the designers of such a system must control in order to reach an acceptable balance. By using the presented techniques, software developers can easily build user interfaces that present basic or medium-complex functionality, while having the benefits of standardization. For highly complex functionality, more advanced patterns and technologies are needed, which will be presented in a future paper.*

Cuvinte cheie: *baze de date medicale, sisteme informatice în domeniul medical, interfaţa cu utilizatorul*

Rezumat: *Sistemele Informatice din domeniul Medical (SIM) se bazează în general pe baze de date mari care stochează date medicale. În afară de problemele specifice legate de proiectarea structurilor de date, aceste sisteme prezintă o provocare pentru dezvoltatori în ceea ce priveşte arhitectura nivelului extern al bazei de date, strâns legată de construirea interfeţei cu utilizatorul. Această lucrare prezintă trei aspecte specifice legate de interfaţa cu utilizatorul a unei baze de date medicale (Proprietatea temporală, Procesul de alocare şi Lista deschisă de atribute) folosind o abordare orientată spre date pentru nivelul extern. Abordarea se bazează mai mult pe vederi, construite pe partea de server şi pe partea de aplicatie client. Acestea pot fi completate de un număr redus de proceduri stocate pentru a implementa funcţionalitatea completă a sistemului. În modul nostru de abordare, complexitatea acestor procese este în mare parte încorporata în interogări care sunt rulate pe serverul bazei de date. Aceste interogări pot deveni însă complexe si consumatoare de timp. Două probleme majore apar de aici: în primul rând, complexitatea poate duce la prezentarea inconsistentă a aceloraşi date (avem în vedere sisteme informatice de mari dimensiuni) şi în al al doilea rând, interogări complexe pe volume mari de date pot afecta grav performanţa generală. Aceştia sunt factorii pe care designerii unui astfel de sistem trebuie să îi controleze, în scopul de a ajunge la un echilibru acceptabil. Prin utilizarea tehnicilor prezentate, dezvoltatorii de software pot construi cu uşurinţă interfeţe utilizator care prezintă o funcţionalitate de bază sau mediu-complexă, beneficiind în acelaşi timp şi de avantajele standardizării. Pentru o funcţionalitate mai complexă, sunt necesare modele şi tehnologii avansate, care vor fi prezentate într-o lucrare viitoare.*

INTRODUCTION

Healthcare Information Systems (HIS) generally rely on large databases that store medical data. Apart from the specific issues related to the design of the data structures, these systems present a challenge to the developer regarding the architecture of the external level of the database, closely related to the building of the user interface. The main issues that we have encountered while building our HIS are the following:

- The Filter Processing – filtering data by different criteria;
- The Order Processing – sorting data to fit the user's needs;
- The Synchronization Process – filtering subsets of data according to a currently selected value;
- The Temporal Property – managing data that has a limited validity in time;

[1]Corresponding author: P. Olah, Str. Ghe. Marinescu, Nr. 50, Cod 540136, Târgu Mureş, România, E-mail: olah_peter@yahoo.com, Tel: +40265 212111

Article received on 20.08.2012 and accepted for publication on 28.09.2012
ACTA MEDICA TRANSILVANICA December 2012;2(4):180-182

- The Allocation Process – associating datasets to a given record in the database;
- The Open Attribute List Specialization – managing data which has a growing list of properties associated;
- The processing of complex medical data structures- trees and multitrees;(1)
- The reporting process – generating useful information for the user.

The approach is mostly based on data views, constructed on the server side and on the client application side. A small number of stored procedures however are necessary to implement the full functionality of the system.

PURPOSE

In this paper we will discuss three specific issues related to the user interface of a medical database using a data-oriented (2) approach for the external level (The Temporal Property, The Allocation Process and The Open Attribute List Specialization).

METHODS

The software development techniques that we propose in this paper are derived from a real-life project involving medical data for gastroenterology patients. The requests of this system were complex and had an evolution over time during the software design and development process. Our approach was not only to deal with the problem at hand, but to develop solutions for a whole category of similar problems. Thus, using theoretical concepts found in literature, we have identified the above mentioned issues and have developed a set of solutions that involve database design patterns, application architecture solutions and user interface module prototypes.

The architecture of the system is client-server. We used a relational database management system (DBMS) on the server side. For the client side, we used a software development environment which provides Object Oriented programming facilities but it also has its own relational database engine, thus enabling the use of client-side data views.

RESULTS

A. Modelling Time – The Temporal Property

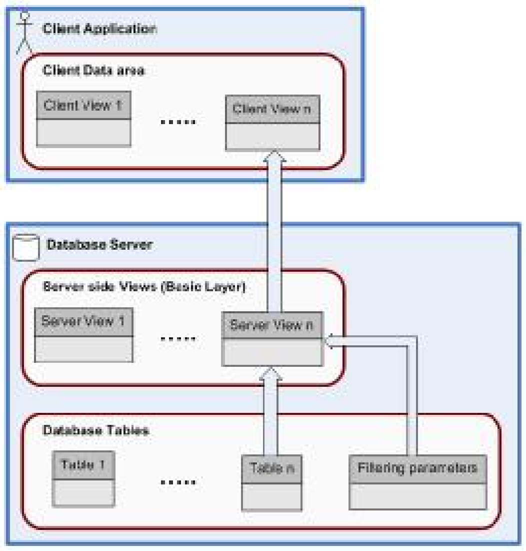
Time is an omnipresent dimension that influences all processes in our world. That includes also the evolution of medical data, be that clinical data or research related. That being said, one core requirement of a HIS will always be the ability to record chronological data or the evolution of data over time.

We implemented such a mechanism in our database by using a dual time-stamp which includes a start and an end date. This allowed us to model intervals of time rather than just have singular points in time, as would be the case when using single timestamps. When singular time-points are needed, the interval will be shrunken by registering the same date and time for both its ends. This approach provides a complete model of time. It requires however more processing power for interrogating the database.

Every query that retrieves time-bound data has to be filtered using a given or a predefined time interval. In order not to overload the user interface with time filtering controls we implemented a special table in the database that holds such time-intervals for every user.

As illustrated in figure no. 1, these time-interval filtering parameters are then used in an intermediate layer of views, stored on the server. The programmers that implement various parts of the interface will use these views in their queries instead of the basic tables. This way, in many cases, the developers do not even need to know that the data they are querying is time-bound. This approach proved to boost development efficiency significantly.

Figure no. 1. Graphic representation of the use of the filtering parameters

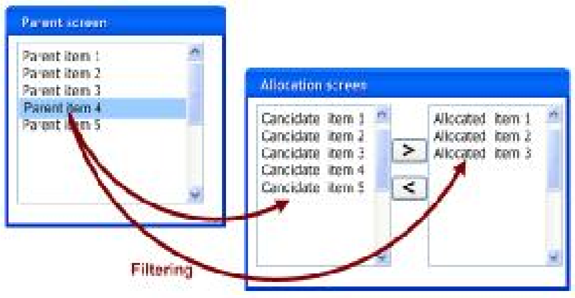


B. Modelling Subsets - The Allocation Process

Processes like assigning an item to a group or category are often found as a requirement at the user-interface level of a HIS. Usually, the user will assign a number of elements from a set to a given group or category. In turn, the group or category is usually represented by a record in a table, or an element in a second set.

So, if we look at this process from a set theory point of view, we need to associate some members of set A (i.e. a subset of A) to a single member of set B. The user will usually browse through set B and to each of its elements will allocate a subset of A. We will call set B the set of “Parent items”. The subset of A already allocated to a given parent will constitute the “Allocated items” list and the rest of A will represent potential items to be allocated, or “Candidate items”.

Figure no. 2. Graphic representation of the theory of items



When building the Graphical User Interface (GUI) we represent the parent set using a screen that offers full CRUD (Create, Retrieve, Update, Delete) functionality. This contains a data-set that we call the Parent View. The allocation will take place using a second screen synchronized with the first one. Here, we see the two subsets of A, the already allocated items and the candidate items along with the allocation/deallocation buttons (figure no. 2). The two subsets of A are implemented using two special views in the external level of the database.

The Allocated Items View

This view retrieves the database records of A that are already allocated to the current record in B. The most important feature of this view is that it is filtered according to the current record of the Parent View, using a key of the parent table.

The Candidate Items View

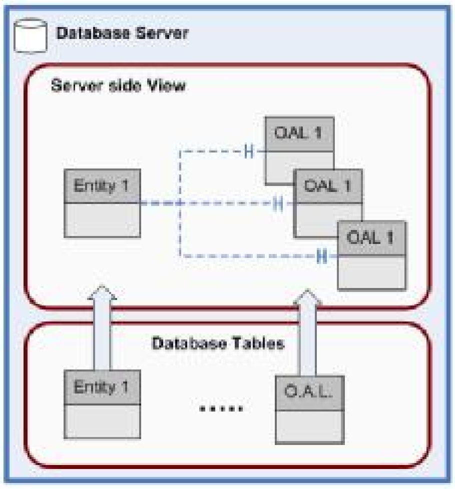
This view has to retrieve the “Candidate Items” list according to the current record of the Parent View. This means that it will have to construct a subset of A by subtracting the already allocated items from the entire set. Implementing such a view as an SQL query requires the use of a NOT IN clause, which can raise some performance issues.

If the table that represents set A holds large amounts of data, more filtering options can be implemented. These will be accessible through the GUI using specially designed controls.

C. The Open Attribute List Specialization

One of the challenges posed to the database designer when building a HIS that will be used in medical research is that not all the requirements can be formulated at an early stage. More and more requirements will come as the system is used and many of these will be database related. In order to prevent the constant redesigning of the database due to this process, we have used a design pattern that models some of the attributes of an entity extensionally rather than intensionally. We implemented this by a special table that holds attributes as records and not as columns. This table can then be connected to another table representing an entity via a many-to-many relationship (which can also hold the values of the attributes if needed). We called this design pattern an Open Attribute List (OAL).

Figure no. 3. Graphic representation of Open Attribute List



A specific issue that we have encountered with this design pattern was the extensive use of outer-joins when we needed to convert these attributes stored as rows into columns in various views (figure no. 3). These joins are imposed by a convention that we set in place stating that every element of an entity can have zero or more attributes recorded in the OAL. This provides great flexibility but can have a negative impact on the performance.

DISCUSSIONS

Healthcare Information Systems require highly specialized solutions from an Information Technology perspective.(3) However, this does not exclude the benefits of generalization which produces reusable elements in the development process. These elements can be database design patterns, application architecture solutions and user interface module prototypes. In the above presented approach, we have focused on the first and the second from this list.

Most of the complexity of the system is modelled in the views used to query the database. In order to keep the ever growing set of views manageable, we recommend structuring it into layers, a basic layer that should be stored on the database server, and one or more “user-layers” that will be developed using the views from the basic layer. The basic layer will provide some often used denormalizations, as well as some specific functionality, for instance filtering using predetermined time-intervals. The user views will be part of the external level of the database according to the ANSI/SPARC three levels architecture.(4) They will be constructed using the basic layer and/or the actual tables of the database.

This approach centralizes the maintenance for the sensitive parts of the database, thus improving consistency and development efficiency. However, for large databases, some performance issues may arise due to the extra processing needed for the basic layer of views.

CONCLUSIONS

By using the above presented techniques, software developers can easily build user interfaces that present basic or medium-to-complex functionality, while having the benefits of standardization. This will reduce the development time and more important, it will reduce the number of errors and/or inconsistencies in the system. The use of the open-attribute list specialization reduces the need for further interventions made by the software development team in case of new requests, as this type of data structure allows the end-user to define and configure new attributes. For highly complex functionality, such as modelling trees or multitrees (e.g. patient observation charts), more advanced patterns and technologies are needed. Solutions for this later issue will be presented in a future paper.

Acknowledgement:

This paper is partly supported by the Sectoral Operational Programme Human Resources Development (SOP HRD), financed from the European Social Fund and by the Romanian Government under the contract number POSDRU 64331 and POSDRU 60782.

REFERENCES

1. Furnas GW, Zacks J. Multitrees: Enriching and Reusing Hierarchical Structure. Proceedings of the ACM CHI 94 Human Factors in Computing Systems Conference; 1994. p. 330-336.
2. Lewis B. Data-Oriented Application Engineering: An Idea Whose Time Has Returned. The Data Administration Newsletter - TDAN.com. January 1; 2007.
3. Muji M, Ciupa RV, et al. Database Design Patterns for Healthcare Information Systems. MEDITECH 2009, IFMBE Proceedings 26, p. 63-66.
4. Date CJ. An Introduction to Database Systems (8th edition). Addison-Wesley; 2004.