Flexible Security and Search Capability for a Relational Database with Externally Linked Multimedia Data Files

Ira Rudowsky[1], Olga Kulyba[1], Mikhail Kunin[1], Dmitri Ogarodnikov[2] and Theodore Raphan[1,2]

[1]Institute of Neural & Intelligent Systems
Department of Computer and Information Science
Brooklyn College of CUNY
Brooklyn, New York, 11210
e-mail: rudowsky@brooklyn.cuny.edu
and
[2]Department of Neurology, Mt Sinai School of Medicine
New York, New York, 10029
e-mail: raphan@nsi.brooklyn.cuny.edu

Abstract

Recently, a prototype relational database system has been developed that has the capability of storing and querying analog and video information for interaction with data analysis applications [8]. The system provides for numerous querying categories via indexed meta-data using pre-filled, drop-own combo boxes. A shortcoming of the design is the growing number of categories that could in time become unmanageable. One purpose of this study was to explore a more flexible design that would reduce the number of categories for storing and querying the database. In this design, the user enters a compound, logical query statement with the desired keywords in addition to using the existing combo boxes. This proved to be a more efficient and flexible strategy for accommodating expanding search categories. Another purpose of this study was to add security features in accessing database information to restrict sharing of information about subject populations. This has become an important requirement in accessing subject information in research studies. To this end we have developed an easily reusable security hierarchy in which individual users and groups of users can be granted rights to objects of the system e.g., database tables or user interface tabs. The permissions granted to a user are dynamically updated by group permissions as the user joins or leaves these groups. The initial implementation of security features has proven to be efficiently handled by the interface. Thus, a prototype for a flexible and secure database querying capability has been implemented that can be scaled to meet the expanding needs of researchers.

Keywords: Relational Databases, Multimedia Data, Meta-data, User Interface, Security and Data Query.

1. INTRODUCTION

The development of relational databases has significantly improved the performance of storage, search, and retrieval functions [1], [2]. With the expansion of scientific data to include non-textual information such as digital streams representing analog and video information, relational databases have been extended to incorporate these new modalities [3], [4], [5], [6], [7]. Recently, we have designed and implemented a relational database system, using Oracle 9i, and interfaced it with a data analysis application. This demonstrated that a flexible interface between applications and databases are potentially important ways to enhance data mining capabilities [8]. The system stores metadata describing external files of various modalities, including digitized analog channels, event channels and video images linked to the data. A graphical, user interface provides for robust querying of the metadata and links have been established between the query results and the visualization/analysis program. As a result, selected query results can immediately be displayed. The rapid query and link capability enables the user to find potential relationships between trials of experiments that would otherwise be difficult for researchers to determine. The purpose of this study was to extend the design to address the problems associated with a growing number of categories and the need to have easily manageable and flexible security feature as the system is scaled for a wider range of experimental procedures and multiple user interactions. In this paper, we describe how these

problems can be addressed using free-format searching and querying and dynamically varying group and user permissions.

2. FLEXIBLE QUERYING

The user interface for the system (The Data Interface Application (DIA)) provides functionality via four screens that are accessible via the tabbed interface - *Logon, Query, Result* and *DBManage* [8]. When logging on to the system, the user provides an id and password to validate access and set the appropriate rights granted to that individual. The *Query* tab provides the capability to formulate SQL queries and submit them to the Oracle database in order to retrieve those experiments that fit the selected criteria. Drop-down combo boxes for such fields as Experiment ID, Trial ID, Subject ID, Apparatus, and Location as well as date fields to specify the Start and End Date of a trial are provided to enable the user to specify the search criteria. Some fields are related to and control the contents of other fields. For example, by selecting Experiment ID, the fields Trial ID, Subject ID, Apparatus, Start Date, End Date, and Location are limited to those occurring within that experiment. By selecting just Apparatus, a much broader result-set will be returned. In addition, Subject Type will be limited to human only or the various animal types. Selecting Subject ID will automatically determine Subject Type. When the *Submit* button is activated, the DIA displays the results on the *Result* tab screen. The user can view the returned rows from the query, along with some descriptive fields, and then click on a row to invoke the VMF Analyze program to graphically display the selected VMF file and employ the various analysis tools the program contains. The *DBManage* tab provides a series of user-friendly screens that enable the user to update, add or delete records from the various tables in the Oracle database to reflect the changing test environment (e.g., additional subjects, new apparatus, etc.). Thus, basic table maintenance can be performed without the need of a trained Oracle database administrator.

During the course of a research project, new types of apparatus are being developed, additional types of subjects are being tested and different types of tests are being devised. This had led to a wide variety of trial-specific information that can no longer be accommodated into the existing categories displayed on the query screen but nevertheless must be stored along with the other trial-related metadata. One approach would be to add a field within a table for each new type of test, subject and/or apparatus. However, this approach would lead to many fields, which have no entries, leading to a sparsely populated table and overly complicated user interface. The ongoing addition of new fields would also incur the cost and time of programming changes to the database and application code but in the end would be used infrequently as to not warrant the expenditures. To address these problem while maintaining a manageable number of search parameters, a free format text field, COMMENT, has been added to the EXPERIMENT table to hold trial related information that does not fit into the existing query categories. As opposed to the combo boxes that provide a drop down list of entries to choose from, the user is required to type specific text in the COMMENT query text field (Figure 1) – a minor inconvenience for the results obtained. It is also possible to combine key words in searching the COMMENTS field by using OR and AND connectors. The program parses the input, constructs the appropriate SQL statement and displays the returned rows in the *Results* tab (Figure 2). Thus, the COMMENTS field together with the ability to perform free-format search on the field is a simple mechanism to provide scalability without adding overhead and complexity to the interface.

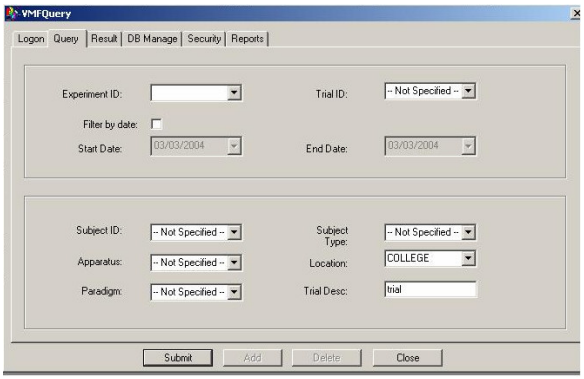

Figure 1. Free-text query in Trail Desc text box

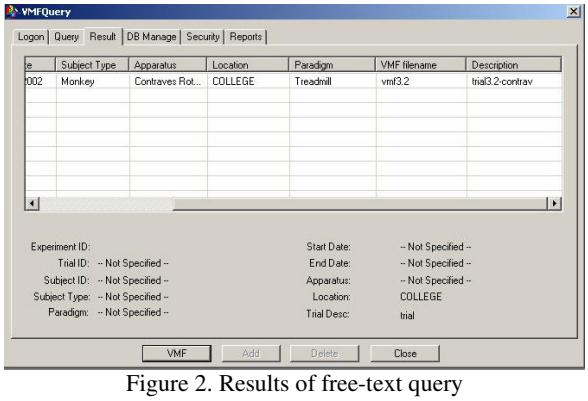

Figure 2. Results of free-text query

3. FLEXIBLE SECURITY

Another purpose of this work was to determine how security features could be embedded within the database retrieval design. Such features are critical if the data must be protected from unauthorized access, such as in multi-user environments. Although Oracle9i provides extensive security features, their implementation is not user friendly ans a database administrator (DBA) would be required to properly administer them. Security in the application is required not only to control access to database objects, such as tables and rows within tables, but to restrict functionality of the user interface as well. We therefore designed and implemented a security model that employs our own tables and an interface that is reusable in other applications and extendable to other relational database systems and platforms.

The security features are implemented using Microsoft Foundation Classes as well as our own classes written in C++ . Each *SecurityObject* i.e., any resource that needs to be secured (for example, a table, a particular column or a tab in our application), contains a *SecurityDescriptor*. The *SecurityDescriptor* contains two *Access Control Lists* (ACLs) - one holds user permissions and the other group permissions. An ACL entry consists of three *Access Control Entries* (ACEs), which control allowed, denied, and grantable permissions of a particular user or group. An ACL associates grantees (users and groups) to the permissions that apply to them. *SecurityObject*s are arranged in a hierarchical form so that permission settings may be inherited by a child. Every *SecurityObject* instance has a parent and zero to many children.

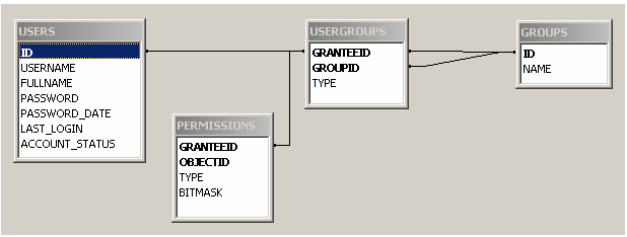

Figure 3. Security tables and their relationships

Four tables have been added to the database to implement security – USERS, GROUPS, USERGROUPS and PERMISSIONS (Figure 3). The USERS table contains information relating to individual users of the system while the GROUPS table is used to associate an individual user or group with another group. That is, an individual user can belong to one or more groups and a group can belong to one or more other groups. This information is stored in the USERGROUPS table.

An object that is to be secured, such as a table, has an entry in the PERMISSIONS table for each grantee (user or group) that is associated with it. Each bit in the BITMASK column identifies whether a specific right is granted. For example, if a bitmask has two bits – the first representing read permission and the second write permission, 0 in the first bit indicates no read access while a 1 in the second bit indicates that write access is granted. Thus, the bitmask string of 1's and 0's defines all rights to the object for a grantee. By ANDing the bitmask of an object for which the user is a grantee together with the bitmasks of all groups the user belongs to, the rights of a user relative to that object are determined. The classes and tables developed in this research provide an adaptable security infrastructure to grant detailed knowledge of individual experiments only to those individuals or groups permitted under security regulations.

The user interface implementing the security features can be visualized as a tree hierarchy. By clicking on the *Users* heading of the tree, a list of all users and their related information is displayed on the right side of the screen (Figure 4). Only select users will have permission to make changes to permissions. By selecting an individual user, the right hand side of the screen displays detail user information (Figure 5). Any changes entered can be saved by clicking on the *Save* button. A new user

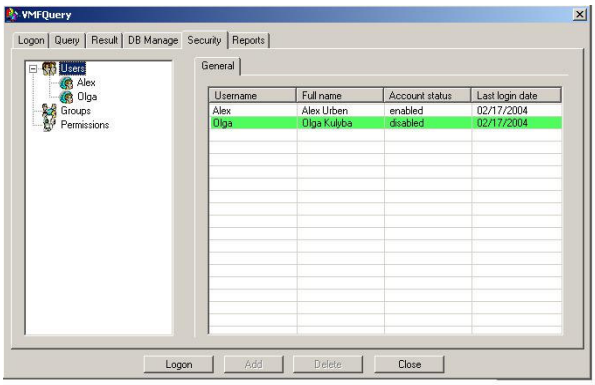

Figure 4. Security Screen – All Users

can be added to the tree by selecting the *Users* category, then clicking the right-hand button of the mouse and choosing *Add* from the menu. The detail screen will appear on the right side of the screen

where the user-related data is entered and saved. To delete an existing user, it is necessary to select that user in the tree, right-mouse click and a *Delete* menu will appear on the screen.

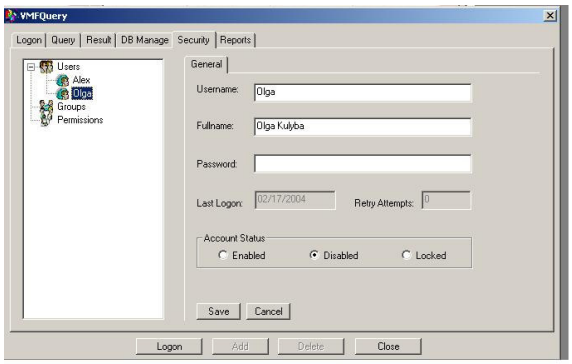

Figure 5. Security screen – individual user detail

Group information appears as another level in the tree (Figure 6). By clicking on the *Groups* heading, a list of all groups appears on the right side of the screen with detail information about each group. Selecting an individual group brings up, on the right of the screen, a form with two tabs (Figure 7). By default, the *General* tab is displayed. This tab provides the ability to change the group name. Selecting the *Role Members* tab displays the form shown in Figure 8. Here, individual users can be added to or removed from the group (top half of the screen) and other groups can be added to or removed from the group. This allows the user to

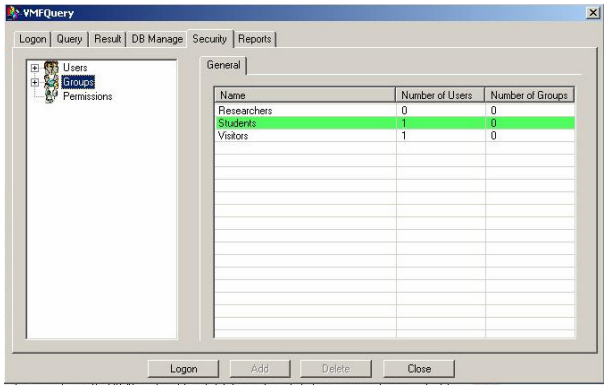

Figure 6. Security screen – All Groups

formulate the security in a modular way by using existing groups to serve as the basis for a new group.

Finally, the permissions branch can be viewed by clicking on the *Permissions* node of the tree (Figure 9). On the right side of the screen, each table is listed along with its current permission settings (SELECT, INSERT, UPDATE and DELETE). Those checked off indicate where permission is allowed while those disallowed are unchecked.

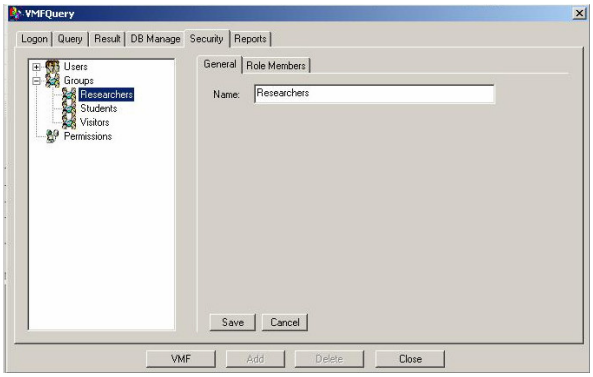

Figure 7. Security screen – individual group detail

These settings can be made at the user level or group level. For each table, an individual user's SELECT, INSERT, UPDATE and DELETE rights are then determined by ANDing the individual's bitmasks along with the corresponding bitmasks for each group of which the user is a member. We have thus added a user-friendly security feature to our system that provides protection at the table level. Security can be administered by a user with appropriate permission and having basic experience in traversing and manipulating a Windows-type graphical interface. By allowing individual users to belong to groups and groups to belong to other groups, the security is flexible and robust. In addition, the design is easily portable to other relational database systems as it does not employ any product specific features of the database.

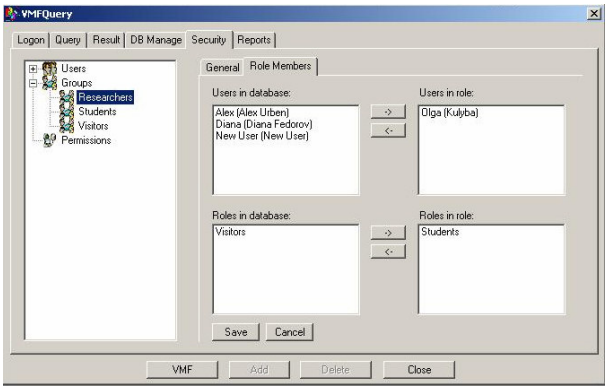

Figure 8. Security screen: Assigning users and groups to the Researchers group

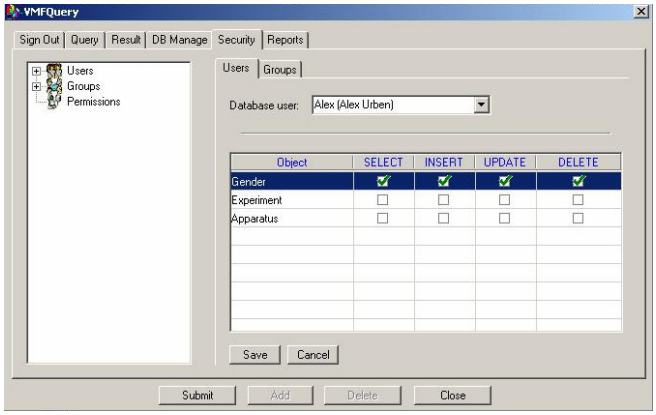

Figure 9. Permissions screen

4. CONCLUSION

Thus, we have designed and implemented a database system that can be accessed through data analysis applications having flexible and scalable querying capabilities within a secure data context. As the number and variety of experiments grows, the search capability can grow in parallel without requiring continuous software enhancements. By implementing security in an object oriented manner, new users and groups as well as tables and functionality can be made to inherit security features that have already been built into the system in a selective manner. The interface structures that we designed have the potential of being utilized across a wide range of relational database platforms.

5. ACKNOWLEDGEMENTS

This work was supported by grants 65397-00 34 from PSC-CUNY, P30 DC05204, DC05222, EY04148 from the NIH and NASA Cooperative Agreement NCC 9-58 with the National Space Biomedical Research Institute.

6. REFERENCES

[1] C. Meghini,, F. Sebastiani and U. Straccia, "A Model of Multimedia Information Retrieval", **Journal of the ACM** , 48(5), pp.909-970, 2001.

[2] R. Weber, J. Bolliger, T. Gross and H.-J. Schek, "Architecture of a Networked Image Search and Retrieval System", **Eighth International Conference on Information and Knowledge Management**, Kansas City, Missouri, USA , pp. 430-441, Nov. 1999.

[3] B. Ozden, R. Rastogi and A. Silberschatz, "Multimedia Support for Databases", **Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems**, pp 1-11, May 1997.

[4] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz, "Efficient and Effective Querying by Image Content", **J. Intell. Inf. Syst.** 3(4):231-262, 1994.

[5] C. Traina Jr., A.J.M. Traina, R.R. dos Santos and E.Y. Senzako, "Support to Content-Based Image Query in Object-Oriented Databases", **Proceedings of the ACM Symposium on Applied Computing**, February 1998.

[6] V.E. Ogle and M. Chabot, "Retrieval from a Relational Database of Images", **IEEE Computer**, 28(9) pp. 40-56, September 1995.

[7] S. Chaudhuri and L. Gravano, "Optimizing Queries over Multimedia Repositories", In **Proceedings of SIGMOD '96** (Montreal, Canada, June 1996). ACM Press, New York, 1996, pp. 91-102.

[8] I. Rudowsky et al, "Relational Database Linkage of Scientific Applications and Their Data Files", **Proceedings of the 2003 IEEE International Workshop on Soft Computing in Industrial Applications**, June 23-25, 2003, pp. 55-59.