USER VIEWS -- A MECHANISM FOR A USER-FRIENDLY INTERFACE TO A LARGE-SCALE DATABASE MANAGEMENT SYSTEM*

A. Adamska, H.D. Covvey, D. Corneil and E.D. Wigle

Toronto General Hospital and the University of Toronto
Toronto, Ontario, Canada

Abstract

The concept of an external views system (EVS) for a database management system (DBMS) was introduced a long time ago. The idea is to allow a user to perceive data in the form most convenient for him. Thus the user can focus on those data and those relationships that are of some interest, ignoring all others.

Few commercial DBMS support the above concept to even a limited extent. We have designed and implemented our EVS on top of the TOTAL (Cincom) database system.

The particular requirements of the medical research environment allowed us to restrict the operations on views to retrieval only and thus helped to avoid numerous problems connected with update operations via views. Using our EVS-system casual users can access a view with little or no instruction.

Introduction

The Problem and Why Solve it

A large number of institutions utilize computer-based DBMS as a tool for storage of patient and research data. These systems are useful tools for programmers and technically trained personnel, but usually fall short of fulfilling the needs of casual users.

We have addressed the problems of casual user access and will present here a solution, based on a commercial DBMS (TOTAL), that is designed to support external user views -- a mechanism for allowing users to access a DBMS as if it was created only to serve their specific needs. Our External Views System (EVS) has been developed for the existing DBMS in the Division of Cardiology of Toronto General Hospital.

Overview of the DB and EVS Concepts

In order to understand the idea of the external views, we must be aware of database architectures. The remaining part of this section introduces the DBM and EVS concepts. The second section presents some of the existing EVS systems. In the third section we describe the computing environment where our design took place. The fourth section contains the actual design of the EVS. Our conclusions are in the last, fifth section.

The ANSI/SPARC Architecture Overview. The need for a convenient users' interface was recognized in 1970 in the Guide and Share report. A similar interest was shown in the ANSI/X3/SPARC report[1] in 1975.

The main components of ANSI/SPARC framework are: persons with rotes, processing functions, and the interfaces among them. The ANSI/SPARC database structure is divided into three levels: the internal schema, the conceptual schema, and the external schema.

The internal level is closest to the physical storage of data.

The heart of the system is the conceptual schema, which is the next level. It describes the enterprise as an entity and should change only with the enterprise itself. This level is logically placed between the internal and the external levels.

The third level is the external level. It is closest to the user by providing an application oriented view of data. Logically, an external view is like a window presenting the user with an up-to-date version of the requested parts of data.

Data Independence and Security. An important advantage of viewing a database according to the ANSI/SPARC framework is the high level of data independence that is implicit. The external user views can be tailored to user's needs and all interaction with the database restricted to only the required data and structures. There is no need for any user to be aware of or concerned about the internal structure and it can be changed to ensure greater efficiency with no impact on the users. Although user views of the database may change, the conceptual level remains unchanged. The only changes required are the mappings between levels.

A user may select his view from the conceptual schema by selection of data items and by creating structures. Depending on the models a given system supports, the user may omit in the external schema data items, records, sets, etc. from those on menu in the conceptual level. The user may also want to use different domains for data items, such as: a) different units for numeric values may be used (e.g. kilograms instead of pounds); b) the data types may be different (by data type we mean the representation chosen for domain values, such as numbers in different bases, character strings, etc.); c) if a conceptual domain is coded, we may wish to use either the codes, or the decoded values of the conecptual domain; d) the external domain may be virtual, that is its values are computed, based on

*WORK SUPPORTED BY THE ONTARIO HEART FOUNDATION

0195-4210/82/0000/0847$00.75 © 1982 IEEE

some algorithm, from several conceptual domain values, or the conceptual domain may be split into two or more external domains.

A user may want to use a specific structure for his data different from its conceptual level structure.

The use of an external schema is one way security constraints can be enforced. User views can be created to deprive a user of access to information to which he or she does not have a right of access.

Support of the Multiple Views. A multiple view support system is a facility that allows the users to see a datababase in many different forms even when the underlying information is the same. The difference would lie in the data taken into consideration.

Just as for different views, different models for those views can also be supported. This allows a user to consider a database in a model of choice and to employ the desired data language. Examples of models would be hierarchical, network and relational.

Updates. As we have mentioned before, a view is a window. Data manipulation statements can be supported, in general, if there exists a one-to-one mapping between a view and an underlying relation. This means that if the view is a join of two or more relations and is defined by an application of aggregate functions, updates are complicated and require extra conditions.

If no update operations are supported through a view, then such a view is called a snapshot. It must be created each time there is a change in the underlying relations that concern this view.

An extensive study on possibility of update support can be found in[2] and [7]. Both papers present tables of transformations of view updates to updates in underlying relations. J.M. Osman in[10] presents an even more complicated system, where some of the views are kept explicitly for further reference. He states the conditions update operations have to satisfy in order for such requests to be allowed.

We will show here an example of update operation on a view obtained through a join operation. Let us consider the following relations:

Students and Their Courses

Student#	Course#
SM17	CS220
SD29	CS231
SK37	CS231

Teachers and Their Courses

Teacher#	Course#
TD4	CS220
TD5	CS220
TR6	CS231

Figure 1

The view would be a simple join over a COURSE# and would show for each student all the teachers he could choose from.

Student#	Teacher#	Course#
SM17	TD4	CS220
SM17	TD5	CS220
SD29	TR6	CS231
SK37	TR6	CS321

Figure 2

If the deletion of SD29, TR6, CS231 would be translated into deletions of SD29, CS231 and TR6, CS231 from underlying relations in Figure 1, then SK37, TR6, CS231 would also disappear from the view. On the other hand, an insertion of SM49, TR3, CS220 translated into insertions SM49, CS220 and TR3, CS220 will cause the appearance of SM17, TR3, CS220; SM49, TD4, CS220 and SM49, TD5, CS220.

The problem with a deletion in the above example can be solved by applying different translations. It is enough to perform a deletion in one instead of in two of the underlying relations. In our example it is a STUDENTS AND THEIR COURSES relation. There is no simple solution to the insertion problem. By eliminating all the views that cannot support correct updates, we end up with the class of updatable relations identified in[2].

The complexity of this problem sharply increases in a multi-model environment. In addition, some of the frequently accessed user views might be stored explicitly and the updates on them should be propagated as well. It is not surprising that existing commercial systems support little of the rich theory behind the external schema.

Previous Work on Views

Existing Systems Supporting User Views

Which models have commercial systems chosen for each level in the database architecture and especially for the external level? How sophisticated is this level and what are its main features?

SYSTEM R[3] is an experimental relational database system. It supports new relations (views) by storing their definitions expressed as queries. Any actions performed against the external views are done by query modification.

INGRES[11] is another relational experimental database system. It handles external views in a manner very much the same as in SYSTEM R. In both systems, users are presented with relational views only.

CHEOPS'[8] database offers a user a choice in the model of a view (relational, hierarchical or network) he or she would prefer to use. The queries and updates against a vew are translated into the ones against a conceptual schema. This feature saves the effort of creating a view. The extent of allowed operations is limited to the class of updatable relations mentioned in the Updates section.

ZETA[9] is yet another experimental relational system supporting user views. The user views are maintained by keeping a list of pointers to the tuples in the underlying conceptual relations.

There also exist query languages like ISBL (Information System Base Language)[12] and QBE

(Query-by-example)[13] that are able to support an external view. They contain a feature that allows the execution of a statement to be deferred. LIST or VIEW commands evoke such query definitions and present users with contents of their views.

Let us now look at commercial systems.

In IMS, external views defined over conceptual schemas may omit some of the conceptual segments, together with their children. Some portions of the tree may be inverted. Each external view has a list of operations allowed. In general, all update operations against a view may be performed.

IDMS (a DBTG database system)[5] has external views defined in terms of subschemas, which are simply subsets of schemas. Deletions and modifications (as well as retrieval) can be performed on views and resulting changes will be carried onto the conceptual schema.

We may conclude that the external views are rarely supported in entirety, and reality is far behind the rich external views theory. In most existing systems the flexibility in data items' domains is limited to the change of data types and the update operations are allowed on specific views only.

Descriptions of the Existing Database Systems

The Purpose of the Existing Database System

The Division of Cardiology at Toronto General Hospital uses a database system for the support of numerous clinical research projects. The main goal of this system is the collection of information about diagnostic investigations, surgical and medical treatment, and followup on patients with heart disease.

The system provides services for the generation of patient visit reports and supports clinical cardiovascular research projects. Researchers typically wish to select specific information about patients that satisfy some criteria, for example, all male patients with a given disease who had a given procedure during the year 1980.

The entry of information is done basically through optical mark-sense forms. Physicians or other hospital personnel fill in the proper form for each patient visit or procedure. A large volume of information is entered daily and the amount of data stored grows rapidly, as the information is to be kept for many years.

Another characteristic of the above system is that cardiovascular projects are initiated and terminated frequently. This requires the constant evolution of the database and the queries made on it.

System Design Overview

The architecture of the Cardiovascular Database System is based on ANSI/SPARC guidelines and consists of three levels.

The tools used for implementing the above system are network DBMS TOTAL (Cincom) and a package called DIET (Data Independence Extension to TOTAL).[6] All application programs are written in FORTRAN IV.

The Cardiovascular DBS has the following major hardware components: a V77-824 minicomputer system with .75 megabytes of internal memory; two 150 megabyte disk drives; two 800/1600 tape drives (75IPS); a 16 channel communication multiplexor; a 300LPM line printer and a variety of remote and local terminals.

System Design

The functions of the internal level are performed by TOTAL. An interface with TOTAL has been accomplished by creating an internal database schema that is fully compatible with TOTAL.

The conceptual schema is defined in terms of "forms," which in turn are organized in a network. This allows an easy interface with TOTAL and the possibility of either networks or hierarchies of forms at the external level. (Until, the works reported here, there was a one-to-one correspondence between external and conceptual forms.)

The operations allowed on the conceptual and external level are add, delete, modify and view.

Mapping between levels is done with help of the Data Dictionary/Directory (DD/D). The DD/D is a separate database that contains the information about internal, conceptual and external schemas, mapping between them, encoding and decoding tables for each field, field value constraints and security constraints.

User requests are translated through mappings from the conceptual schema to the internal schema and from the internal schema to the internal storage. Those mappings are performed by translation routines, which use run-time conversion files created from the DD/D for each conceptual form.

This system is straightforward for everyday maintenance and use, and powerful enough to satisfy the needs of some research projects. What it lacks, however, is a more sophisticated external schema, one that would allow a user to view the data in any fashion. Currently, a user is able to visualize the data as a collection of numerous studies on patients. The ability to see information from two or more studies at the same time would clearly constitute a step toward a better external schema.

Design of the External Views System

Restrictions and Selection of Options in Design

Restriction: No update operations supported through views.

Reason: The existing cardiovascular database in the Division of Cardiology presents data entry users and physicians making patient oriented retrieval requests with a specific view of the data. The users visualize the stored information in terms of forms, each form corresponding to the specific cardiovascular study or test. All updates are carried out at this level.

Restriction: Our initial external views system will use a relational model.

Reason: The relational model, which represents the information in the form of a table, is the most simple model, is easily understood and is commonly used. Some claim[3] that users visualize the data in

the form of hierarchies. Hospital input personnel may be thinking in terms of hierarchies, but researchers view the database as a table. The rows of a table correspond to different patients and the columns to patient results.

Choice: External views will be created when demanded.

Reason: There are two possible ways of keeping external views. Explicit external views have serious weaknesses -- storage overhead and the maintenance of structure caused by the duplication of data, or indices to provide access to the view data.

Data Required for an External Views System

Choice: Data about a new external views system should be kept as a part of the DD/D.

Reason: Keeping the data about the external views system as a part of the existing DD/D allows the use of the existing DD/D system and prevents possible inconsistencies between two DD/D's.

The External Views Definition Language

The entry of the data to define a view is done by means of the External Views Definition Language (EVDL). There are many possible choices for such a language. As our system is oriented simply to the casual user, the ease of use should be the main factor in our choice.

The solution is a man-computer dialogue (menu) with numerous on-line manuals and explanations.

The EVDL for the medical database places significant emphasis on the time factor. The time factor inherent in medical data influences the way users will like to see data in the database. Therefore, a specific mechanism should be employed to help the users in the manipulation of time factors. We have decided to incorporate the time factor into the specification of which occurrence of a given piece of information is required in the external view. For example FIXED RATE (LAST) would represent the fixed rate of a pacemaker at the very last visit in time. The choice of which is the last visit is done based on the visit or procedure date. Similarly, MAGNET RATE (LAST -- 3 MONTHS) will correspond to the magnet rate from the most recent visit that took place at least three months before the last one.

We will now present, in an example, the form of the EVDL:

```
ENTER THE NAME OF YOUR VIEW:
   testl
ENTER THE NAME OF AN EXTERNAL ELEMENT:
   pcmk rate
LIST THE ELEMENTS PCMK RATE IS TO BE
CALCULATED FROM FORM NAME?
   ccsu
QUESTION NUMBER WITH TIME FACTOR?
   #17(last-3visit)
FORM NAME?
   ccsu
QUESTION NUMBER WITH TIME FACTOR?
   #17(last-10visit)
FORM NAME?
   (* carriage return only *)
SPECIFY THE WAY PCMK RATE IS TO BE CALCULATED:
   1. EVALUATION OF CONDITION
   2. CONCATENATION OF ALL ELEMENTS
     1
ENTER THE CONDITION THIS ELEMENT HAS TO
SATISFY.
AS OPERANDS USE WORD "FROM" FOLLOWED BY A FORM
NAME AND A QUESTION NUMBER WITH TIME FACTOR.
from ccsu#17(last-10visit)-from ccsu#17(last-
3visit)
ENTER THE NAME OF AN EXTERNAL ELEMENT:
   .  (* repetition of the above *)
   .  (* for every new element   *)
SPECIFY THE BOOLEAN CONDITION
  from ccsu
JOINING CONDITION
  from ccsu #1 = from ccsu #1
WHAT ARE THE CONDITIONS YOUR VIEW HAS TO
SATISFY?
  (from ccsu #17 (last-3visit)> '0.5').
```

Modules Involved in the External Views System

The external views system consists of two parts: view establishment and view access.

View establishment deals with all the actions behind the definition and creation of external views. The information about each new view is gathered by means of the View Definition Language. Data thus obtained is checked for syntactical and structural correctness by the View Definition Verifier Module. A correct view is one for which retrieval is feasible, i.e. there exist access paths to all the desired elements. All the information necessary for this module is supplied by DD/D. Figure 3 shows the modules behind the external views system.

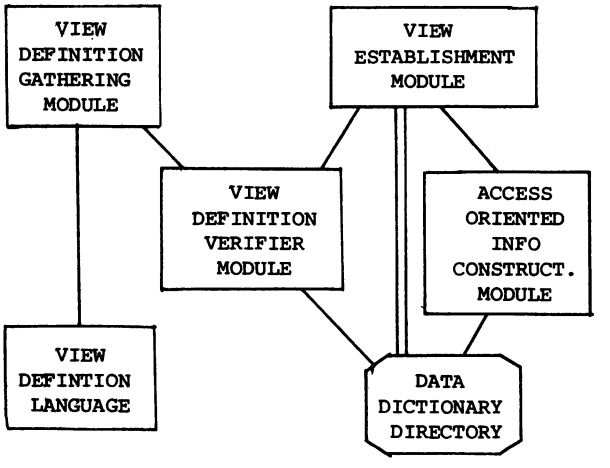

Figure 3

Once the consistency checking stage is finished, the data may be stored in DD/D.

The Access Oriented Information Construction Module performs certain pre-processing functions. It uses the information stored in the DD/D to construct the data structure to speed up the future display of the view contents. Since we are dealing with the I/O bound system, it is to our advantage

to be able to have all the information necessary for the retrieval in the main memory.

There is another module, not presented in the diagram on Figure 3, that will be required. It seems highly desirable to be able to present a user with the option of specifying his or her own output format. The Output Specification Module should be activated after the Access Oriented Information Construction Module.

At the present time, the output format is standard for all external views and consists basically of the list of external element names and retrieved values. In cases where two or more conceptual fields are concatenated to form one external element, blanks are inserted between conceptual fields.

Retrieval Language

The View Access Module will provide the user with the contents of his or her view. The request should be stated through the Retrieval Specification Language.

Choice: Simple 'LIST' statement with the option of a condition.

Reason: Since we are dealing with a casual user, simplicity is an important factor. Another factor is the ability to retrieve all the required information. Users are assisted in the process of the definition of the views by the computer personnel. Everyday use, however, should be straightforward and problem-free. The language used would be of the following format: LIST(list of attributes)FROM(view name)WHERE(condition).

Instead of the list of attributes we could have the word 'ALL' and all the view items defined would be retrieved. The part specifying the condition is optional and the condition itself may be specified only on the attributes belonging to the given view.

The present version of EVS does not include the above LIST statement. Instead it allows the user to see each record belonging to that particular view, one at a time. The conditions that could be specified in the LIST statement may be included during the view definition stage. This would mean the creation of more specific views. The implementation of the LIST statement will be one of the future enhancements of EVS.

Conclusions

The implementation of the external view system was completed in FORTRAN by the author over a 4-month period.

The implementation of EVS has shown that an external views system can be installed successfully on top of a commercial database.

The EVS allows the user to define views based on conceptual elements from all the physical databases. The time oriented definition gives more ease in conceptual element specification. The external element domains may differ from the conceptual. Such changes are possible through the specification of algorithms (concatenation and performance of conditions). Finally, the user is presented with a view in a table form even though the underlying database has a network structure.

References

1. ANSI/X3/SPARC Study Group on Database Management System Interim Report, ACM SIGMOD FDT 7,2 (1975).
2. Arora AK, Carlson CR: "On the Updaptability of Consistent Relational Views," Technical Report Bell Labs, (1980).
3. Astahan MM, et. al.: "System R: Relational Approach to DBM," ACM TODS 1,2, pp. 97-137, (1976).
4. Clemons K: "Design of a Prototype ANSI/SPARC Three Schema DBS," AFIPS, Vol. 48, pp. 689-695, (1979).
5. Codasyl Base Task Group 1978 Report, ACM Data Description Language Committee.
6. Dubien RJ, Covvey HD, Sevcik KC, Wigle ED: "A Database System Implementation Providing Data Independence for Medical Applications," MEDINFO 77, Proc. of the Second World Conference on Medical Informatics, pp. 87-94, (1977).
7. Furtado AL, Sevcik KC: "Permitting Updates Through Views on Databases," Information Systems, Vol. 4, pp. 269-283, (1977).
8. Klug A: "Multiple Views, Multiple Data Model Support in the CHEOPS Database Management System," CSC Tech. Rep. #418, (1981).
9. Mylopoulos J, Schuster S, Tsichritzis D: "A Multi-level Relational System," AFIPS, Vol. 44, pp. 403-408, (1975).
10. Osman JM: "Updating Defined Relations," AFIPS Vol. 48, pp. 733-740, (1979).
11. Stonbraker M, Wong E, Kreps P, Held G: "The Design and Implementation of INGRES," ACM TODS 1, pp. 189-222, (1976).
12. Todd SJP: "The Peterless Relational Test Vehicle a System Overview," IBM System J., Vol. 15,4, pp. 285-308, (1976).
13. Zloof MM: "Query-by Example: The Invocation and Definition of Tables and Forms," Proc. of ACM Int. Conference on VLDB, pp. 1-24, (1975).