Conceptual database security access permissions

ZAKARIA SULIMAN ZUBI[1], MARWAN ALKABLAWI[2]

[1]Computer Science Department, Faculty of Science, Al-Tahadi University, Sirt, Libya, {zszubi@yahoo.com}

[2]Higher Institute of Technology, Sirt, Libya, {tafas2002@yahoo.com}

Abstract

High-level conceptual database design is a widespread method in database built with conceptual models we will illustrate the "mini world" of the database via Database Management System (DBMS) in an independent form. The form will be mapped by the use of a mapping method to reach a DBMS specific model. The database designer should keep in mind both data and functional requirements throughout the whole process.

We will also indicate some database security aspects in our model. Database applications usually have to meet high security level, therefore we must protect the database and the data stored in the database against those who do not have the appropriate access permissions. The design could be established of the access permission system with the specification of the project which may reach the implementation through building the conceptual model. In our approach we illustrate this access design method from high-level to the implementation. First we have to define the user groups and their roles. In connection with the conceptual model (Enhanced Entity Relationship Model) we recommend the use of an improved model, which already includes the privileges of the users, too. These permissions can be represented in a conceptual access matrix model. At this level the privileges of the users are still DBMS-independent, then by mapping, it reaches the low-level database model (relational model), which can be accompanied with the access matrix model. This is the point where access permissions become connected to DBMS-specific elements. Finally, we point out a certain realizations of the access control permissions.

Key words: Security access, access rights, database conceptual design, access permissions, database security.

1. Introduction

Conceptual database design is a major database design technique nowadays. Entity Relationship model (ER), which was introduced in the 70's, and its improved implementation, "Enhanced Entity Relationship model (EER), together with their descendants may play an important role in the process of the database design" [1].

The above mentioned two models which have two great advantages. First we provide a DBMS-independent representation of the modeled world. This means that in the early stages of the design process the designer should not deal with the actual database type, therefore the resulting model can serve as the basis for several different implementations. Second, the ER and the EER models give a graphic representation, which makes it easier to grasp even for the non-technical user. These database-centric models may serve as adequate bases for the continuous communication between the designer and the database users. "This communication can (and should) lead to a better and more precise problem identification" [2].

However, especially when working with large databases, we shall not forget that usually there is more than one user, who would access the database [3]. Generally there are several different users, with different access permissions. It is also the responsibility of the database designer to deal with the security issues. The basis for the design of the access permissions system is evidently the communication between the designer and the user [4]. As conceptual models are easily understood by the user, it is a good idea to start dealing with access permissions at this level. Also, as these models are DBMS-independent, the access permission system designed here can be easily adopted to any implemented database system.

In the following sections we introduce a new conceptual design method, which also includes designing the access permission system. Section 2 summarizes the theoretical background of user privileges. In the Section 3 we offer a possible extension of the EER model as far as notification is concerned. We also show an alternative matrix-based representation of the access permission system- Section 4 covers the relevant mapping issues. Section 5 touches on implementation problems.

2. User Groups

As we have already mentioned it is the duty of the designer to collect the different types of users who would use the system, and the different access permissions these possible users need. This does not mean that the designer should exactly know who would use the system and what for at this early point. The designer should only be able to design the different user groups. Those users belong to one user group who have the same access permissions in the given system. When the system is put into operation new users should be assigned to one of these groups, and in this way they are granted the proper access permissions.

For the sake of security, each user should enter an account identifier and a password whenever accessing the database. Assigning a user to a group basically means that the account of that user should be assigned to a certain group. Therefore users get the proper access permissions through their identifiers.

With databases where there is only one user, or a small group of equal users it is a generally accepted solution to protect the database, itself with a common password [5]. Certainly, this does not require any special planning, nor does it reach the requirements of a multi-user system. Therefore the present paper does not deal with this possibility any further.

As in the design process of the access permissions we categories users into different groups (one user may belong to more than one group) we can speak of the number of user groups and the number of users in a group. The simplest case is when we only have **1** user. Evidently the number of groups would be **1**, just like the number of the users. It can be imagined that there are more users (n), but all of them has the same access permissions in which case we have **1** user group with n users. *The present paper focuses on the general case when we have more user groups, each of which may contain more users.* We have to mention at this point, that what really counts during the design process is not the number of the users in a group but the number of the groups. That is why we concentrate on the design process of the access permissions of the separate groups.

3. The conceptual design of access permissions

"Conceptual design, as far as the user groups are concerned, starts with setting up the different groups"[6]. To make it simpler let us suppose that these groups are independent, without any hierarchical dependencies between them.

First we have to identify the separate groups of users, and give *unique names to these groups*. The next step would be to assign proper access permissions to our groups. Before that, we should discuss what permissions can be assigned to a group.

In everyday use, users would meet some kind of graphical (or character-based) interface through which he or she can manipulate the data. Every operation executed on the database is somehow connected to the data stored in the database. The possible operations are the following:

- INSERT - inserting new data.
- DELETE - removing data.
- UPDATE - modifying data.
- QUERY - selecting data.

This means that during the access permissions design process the designer should decide what data may the users belonging to the different groups *insert, delete, modify or select.*

At conceptual level we have to match these access permissions to the different elements of the scheme. This way we can assign access permissions to *entities, attributes, or relationships* (which connect the entities).

As entity types are nothing else than a group of entities that can be characterized by a set of closely related attributes, we can say that users perceive entities through their attributes. This way access permissions assigned to the different entities correspond to access permissions assigned to their attributes. This also means that access permissions assigned to an entity type always belong to the actual entity regardless of whether it is a special entity type (main class, subclass, weak entity), or not.

Relationships between entity types model how entities are connected to each other [7]. According to this access permissions assigned to the relationships define what operations the user may perform on the relationships between entities (*create, delete, update, and query existing relationships).*

Attributes cannot exist on their own, during the modeling phase we assign them to either entity types or to relationships. Therefore access permissions assigned to attributes can only be interpreted through the entity types and the relationships.

Sometimes access permissions cannot clearly be assigned to the whole of an entity or a relationship. Suppose that a user has certain privileges (insert, update, delete, and query) only to some of the attributes of an entity type. This is called *vertical restriction* of the access permissions. (For example a certain group of users may have access to the name, the address and the telephone number recorded in the database, but should not see the salary information. We say that the query permission is vertically restricted to the name, address and telephone number attributes). Another type of restriction is when the user may have access to all the attributes, but only to certain entities. (For example the users may modify only their own personal information). This is called *horizontal restriction*. It is common that a certain privilege is restricted both horizontally and vertically. (For example a user may only change his address and telephone number but not his name). This is called *mixed restriction.*

Vertical, horizontal and mixed restriction of permissions may also appear in connection with permissions assigned to relationships. Access permissions assigned to relationships refer primarily to permissions to the connection between the entities; however this relationship may as well have different attributes. If the permissions were not restricted vertically they would apply to all the attributes, too. Horizontal and mixed restrictions work analogously to entity types.

Let us look at an example after this theoretical introduction. Suppose that a company organizes training courses for its workers. Teachers come from within the company and from outside as well. Several groups may be started from the same course according to interest. At the end of the courses workers should take exams, which serve as feedback about their success. The database would make it easier to grasp the workers educational progress, which might provide useful information for future organizational questions.

The hypothetical system would operate online, so workers would be able to register for the courses and check their own results. Worker without online connection may register on forms, when the administrators would record their registration on the system. It is also the duty of the system administrators to maintain the details of the courses and the workers and to announce new courses- Teachers mainly use the system to record exam marks and to query personal information of the students.

The different users can be categorized into three groups, namely: administrators, teachers and students. According to the problem specification members of the different groups have different access permissions to the data stored in the database. Figure 1. Shows the EER model of the problem, with the access permissions added. As this diagram is basically an EER model with represented access permissions, we call it hereafter *Enhanced Entity Relationship with Access Permissions mode, (EERAP).*

The basic EER model is extended in a way that access permissions are added to the elements of the model in rectangles surrounded with dotted lines. Figure 1 shows for example that staff members and teachers can only query the details of courses, while administrators may also record, modify and delete them.

Vertical restriction can be seen at the PERSON entity type, where teachers may query every detail except the Address attribute

(Q(All:- Address)). This restriction may as well be indicated like this: Q(All: PID, Name, Telephone, Contact). This notation may seem more straightforward, however when there are several attributes, with only a few exceptions, the first type is much more efficient and advisable to use.

WORKER entity type exemplifies horizontal restriction. Every worker may manipulate only his own responsibilities. It is always practical to explicitly give the conditions that explain which entities should be visible for the users within the horizontal restriction. In the above example this condition may be given with the help of the user ID, which in the meantime identifies users in the system. If there is no connection between the condition and the user ID, the adequate user IDs should be stored in a system table for each condition.

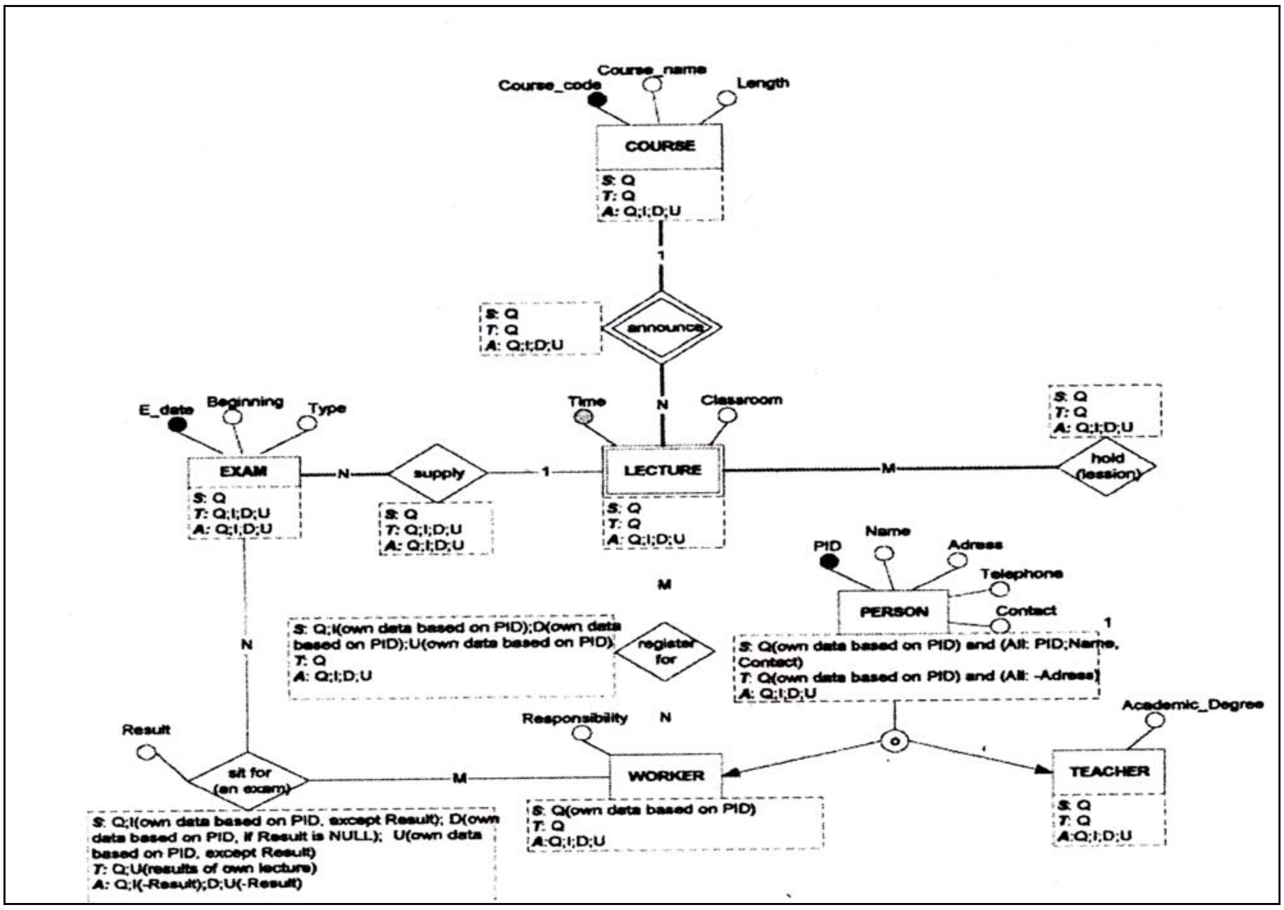

Figure 1. EERAP model

Figure 1 also gives more examples of mixed restriction. For example the query permission of the PERSON entity type is restricted both horizontally and vertically if the user belongs to the Student group.

HER Lind therefore EERAP models may help to solve designer-customer misunderstandings, and clarify problem specification. For future use it is advisable 10 sums up information gathered this way in matrices. The entity types, relationships and attributes of (he EER mode! provide the rows of the matrix- Columns of the matrix represent the user groups. Elements of the matrix show what permissions (insert, delete, update, and query) the different groups have for the certain attributes. Restrictions may be shown beside the elements. The Conceptual Access Matrix of the above example looks like this:

	Student	**Teacher**	**Administrator**
Course: Course code	Q	Q	Q;I;D;U
Course name	Q	Q	Q;I;0;U
Length	Q	Q	Q;I;D;U
Lecture: Time	Q	Q	Q;I;D;U
Classroom	Q	Q	Q;I;D;U
Exam: Exam date	Q	Q;I;D;U	Q;I;D;U
Beginning	Q	Q;I;D;U	Q;I;D;U
Type	Q	Q;I;D;U	Q;I;D;U
Person: PID	Q	Q	Q;I;D;U
Name	Q	Q	Q;I;D:U
Address	Q(own data based on SID)	Q(own data based on SID)	Q;I;D;U
Telephone	Q(own data based on SID}	Q	Q;I;D;U
Contact	Q	Q	Q;I;D;U
Worker: Responsibility	Q(own data based on SID)	Q	Q;I;D;U
Teacher: Academic-Degree	Q	Q	Q;I;D;U
announce	Q	Q	Q;I;D:U
supply	Q	Q:I;D;U	Q;I;D;U
register for	Q; I(own data based on SID); D(own data based on SID); U(own data based on SID)	Q	Q;I;D;U
Hold lesson)	Q	Q	Q;I;D;U
sit for (an exam)	Q; I(own data based on SID); D(own data based on SID, if Result is NULL); U(own data based on SID)	Q	Q;I;D;U
Result	Q	Q;U(own lecture)	Q;D

Figure 2. - Conceptual Access Matrix

The representation of the Conceptual Access Matrix is somewhat closer to the actual realization as the different access permissions are broken down to data. However this is still a DBMS independent representation of the access permissions.

4. Mapping the EERAP model

The next step of the conceptual design process is the selection of the type of the database system (relational, network, hierarchical). Following the selection we have to map the elements of the high level model to the elements of a relational, a hierarchical or a network model according to our selection. As the high level model includes the system of access permissions we have to expand this step with the *mapping rules of these permissions*. As the most popular logical model is the relational model, let us now briefly cover how these permissions can be mapped to the elements of this model type- We cannot give a full description of the mapping here because of space reasons.

In the mapping phase we create a system of relations out of the system of the entities-relationships-attributes. The final relational scheme is reached with optimizing storage and running needs after applying the known mapping rules. The privilege system designed at conceptual level should be adjusted to this scheme. Attributes represented in the Conceptual Access Matrix are converted to table fields. The final relational scheme together with the attached access permissions - similarly to the conceptual matrix - may be represented in a matrix (Access Matrix).

The next phase of the conceptual design process is the physical design [8]. The present paper does not deal with this step, as it has no influence on design of the privilege system.

5. Implementation

The final phase of building a database is realization, which includes testing and setting it up.

In order of secure operation we have to pay enough time to establishing the adequate access permission system in the realization phase. The different DBMS systems offer different solutions in this respect. Some of them (like MS Access) have built-in components for defining user groups and make it possible to assign system and object level access permissions to them (for example you may control what forms a certain group can use). The final step is the classification of users into the groups, If we are to create the privilege system on SQL level, then user groups correspond to different roles. We can use the CREATE ROLE command to create user groups. Then with the help of the GRANT command we can assign system and object privileges to these roles. Finally, the newly created users (CREATE USER) get their appropriate roles (GRANT). The different views may play an important role in realizing the security expectations of the database system (CREATE VIEW).

Whenever creating a multi-user system it should be assured that in the final version it would be possible to create new users, to delete users, and to modify the privileges of existing users. (Often in large systems one user may belong to more groups, therefore the user should choose at start-up which accounts to use.) To ensure such system functions there is a need for creating the appropriate user interfaces where the adequate person can accomplish them. This person is usually called the DBA (Database Administrator). Sometimes more people belong to a group called DBA, who own the highest possible privileges in the system.

6. Summary

For the sake of security user privileges should already be dealt with at design time in multi-user systems- The present paper offered a possible way of including access permissions in the conceptual design process- The modified conceptual design process according to this is as follows:

1- Problem specification

2. Conceptual design

2.1. Building the EER model

2.2. Creating the EERAP model

2.3. Filling in the Conceptual Access Matrix
3- Choosing the DBMS type

4. Logical design

4.1. Creating the appropriate low level model (relational, network, hierarchical)

4.2. Building the Access Matrix

5. Physical design

6. Implementation

Although conceptual design includes not only the design of database requirements, but the design of functional requirements as well, the present paper focuses mainly on the design process of access permissions connected to the former, with special regards to extend the conceptual model.

References

[1] C. Batini, S. Ceri and S.B. Navathe, Conceptual Database Design - An Emily-Relationship approach (Benjamin Cummings, Redwood City. CA, 1992).

[2] R. Elmasri and S.B. Navathe, Fundamentals of Database Systems Benjamin Cummings, Redwood City, CA, 1994).

[3] A-H.M. ler Hofstede and Th. P. van der Weide. Expressiveness in conceptual data modelling. Data & Knowledge Engineering 10(1) (1993), 65-100.

[4] P. Palvia, C. Lio and P. To, The impact of conceptual data models on end-user performance, J. of Database Management, Vol. 3(4) (1992) 4-15.

[5] H. A. Proper, Data schema design as a schema evolution process. Data &
Knowledge Engineering 22 (1997), 159-189.

[6] H-A. Proper and Th.P. van der Weide. EVORM: General theory for the evolution of application models. IEEE Trans. on Data & Knowledge Engineering 7(6) (1997), W 996.

[7] P. Shoval. S. Shiran, Entity-relationship and object-oriented data modelling – an experimental comparison of design quality. Data & Knowledge Engineering 21 (1997), 297-315.

[8] F. Steimann, On the representation of rotes in object-oriented and conceptual modelling, Data & Knowledge Engineering 35 (2000), 83-106.