Security Access Control Permissions for the security enhancement of Conceptual database

Dr.Mohmad Kashif Qureshi

Assistant Professor, Jazan University, Jizan

Saudi Arabia

srk1521@gmail.com

Abstract

It is a process of constructing a data model for each view of the real world problem which is independent of physical considerations. This step involves Constructing the ER Model, Check the model for redundancy, validating the model against user transactions to ensure all the scenarios are supported once the requirements are collected and analyzed, the designers go about creating the conceptual schema. Conceptual schema: concise description of data requirements of the users, and includes a detailed description of the entity types, relationships and constraints. The concepts do not include implementation details; therefore the end users easily understand them, and they can be used as a communication tool.

The conceptual schema is used to ensure all user requirements are met, and they do not conflict

High-level conceptual database design is a widespread method in database built with conceptual models I 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.

I will also indicate some database security aspects in our model. Database applications usually have to meet high security level, therefore I 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

Key words:

Security access, access rights permissions ,access matrix, database conceptual design, access permissions, database security, mapping, mapping phase, vertical restriction, horizontal restriction, EER,EERAP, classification of users .

Specification of the project which may reach the implementation through building the conceptual model. In our approach I illustrate this access design method from high-level to the implementation. First I have to define the user groups and their roles. In connection with the conceptual model (Enhanced Entity Relationship Model) I 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, I point out a certain realizations of the access control permissions.

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"
The conceptual schema is used to ensure all user requirements are met, and they do not conflict
High-level conceptual database design is a widespread method in database built with conceptual models I 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.

First I 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 betIen the designer and the database users."This communication can (and should) lead to a better and more precise problem identification"

HoIver, especially when working with large databases, I shall not forget that usually there is more than one user, who would access the database . 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 betIen 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.

I 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. Different groups (one user may belong to more than one group) I can speak of the number of user groups and the number of users in a group. The simplest case is when I 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 I have 1 user group with n users. The present paper focuses on the general case when I have more user groups, each of which may contain more users. I 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 I concentrate on the design process of the access permissions of the separate groups.

2. User Groups

A users' group (also user's group or user group) is a type of club focused on the use of a particular technology, usually (but not always) computer-related.
Users' groups started in the early days of mainframe computers, as a way to share sometimes hard-won knowledge and useful software, usually written by end users independently of the factory-supplied programming efforts. SHARE, a user group originated by aerospace industry corporate users of IBM mainframe computers, was founded in 1955 and is the oldest computer user group still active. DECUS, the DEC User's Society, was founded in 1961 and its descendant organization still operates. The Computer Measurement Group (CMG) was founded in 1974 by systems professionals with a common interest in (mainframe) capacity management, and continues today with a much broader mission. The first UNIX users' group organized in 1978.
Users' groups began to proliferate with the microcomputer revolution of the late 1970s and early 1980s as hobbyists united to help each other with programming and configuration and use of hardware and software. Especially prior to the emergence of the World Wide Web, obtaining technical assistance with computers was often onerous, while computer clubs would gladly provide free technical support. Users' groups today continue to provide "real life" opportunities for learning from the shared experience of the members and may provide other functions such as a newsletter, group purchasing opportunities, tours of facilities, or speakers at group meetings.

As I 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.
As in the design process of the access permissions I categories users into different groups (one user may belong to more than one group) I can speak of the number of user groups and the number of users in a group. The present paper focuses on the general case when I have more user groups, each of which may contain more users. I 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 I concentrate on the design process of the access permissions of the separate groups.

3. The conceptual design of access permissions

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.

"Conceptual design, as far as the user groups are concerned, starts with setting up the different groups". To make it simpler let us suppose that these groups are independent, without any hierarchical dependencies betIen them.
First I 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, I should discuss what permissions can be assigned to a group.

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

I 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, Iak entity),or not. Relationships betIen entity types model how entities are connected to each other .According to this access permissions assigned to the relationships define what operations the user may perform on the relationships betIen entities (create, delete, update, and query existing relationships).Attributes cannot exist on their own, during the modeling phase I 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. I 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 betIen the entities; hoIver this relationship may as Ill have different attributes. If the permissions Ire 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 university organizes training courses for its students. Teachers come from within the university and from outside as well. Several groups may be started from the same course according to interest. At the end of the courses students should take exams, which serve as feedback about their success. The database would make it easier to grasp the students educational progress, which might provide useful information for future organizational questions.

The hypothetical system would operate online, so students 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 students 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, I 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 Ill 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.

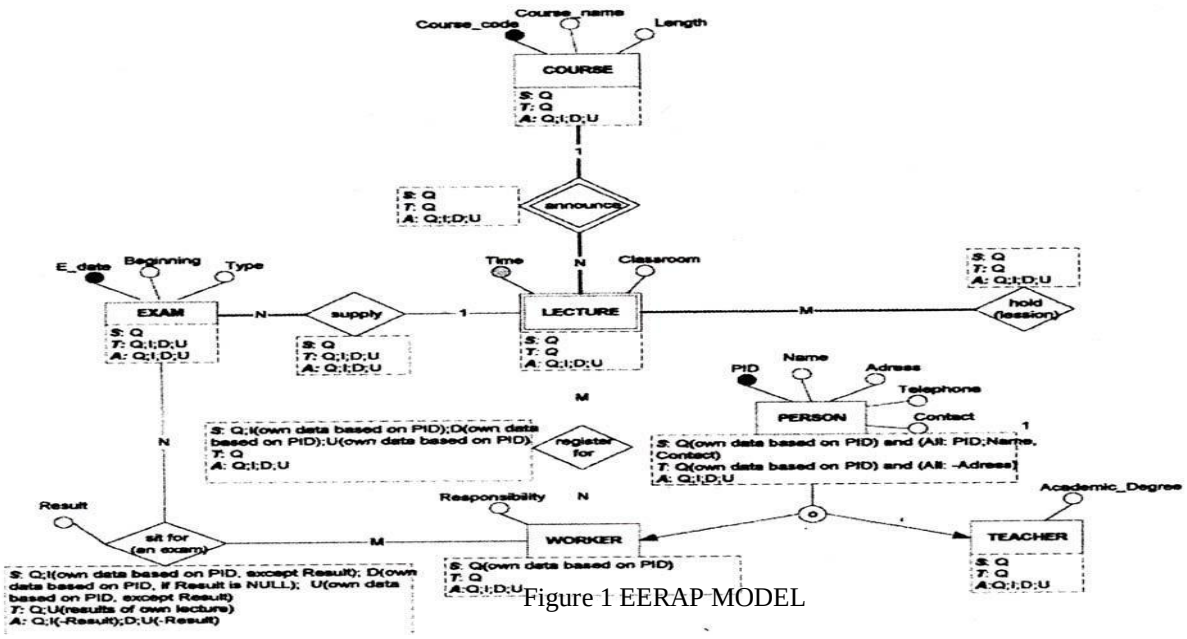


Figure 1 EERAP MODEL

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. Therefore EERAP models may help to solve misunderstandings of designer-customer.

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.

	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

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

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 I 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 I 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- I cannot give a full description of the mapping here because of space reasons. In the mapping phase I 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).
According to the model, the protection state of a computer system can be abstracted as a set of objects O, that is the set of entities that needs to be protected (e.g. processes, files, memory pages) and a set of subjects S, that consists

of all active entities (e.g. users, processes). Further there exists a set of rights $R$of the form $r(s, o)$, where $s \in S$, $o \in O$and $r(s, o) \subseteq R$. A right thereby specifies the kind of access a subject is allowed to process object.
In this matrix example there exists two processes, a file and a device. The first process has the ability to execute the second, read the file and write some information to the device, while the second process can only send information to the first.

	Asset 1	Asset 2	file	device
Role 1	read, write, execute, own	execute	read	write
Role 2	read	read, write, execute, own		

Utility

Because it does not define the granularity of protection mechanisms, the Access Control Matrix can be used as a model of the static access permissions in any type of access control system. It does not model the rules by which permissions can change in any particular system, and therefore only gives an incomplete description of the system's access control security policy.

An Access Control Matrix should be thought of only as an abstract model of permissions at a given point in time; a literal implementation of it as a two-dimensional array would have excessive memory requirements. Capability-based security and access control lists are categories of concrete access control mechanisms whose static permissions can be modeled using Access Control Matrices. Although these two mechanisms have sometimes been presented (for example in Butler Lampson's Protection paper) as simply row-based and column-based implementations of the Access Control Matrix, this view has been criticized as drawing a misleading equivalence between systems that does not take into account dynamic behaviour

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

5. Implementation

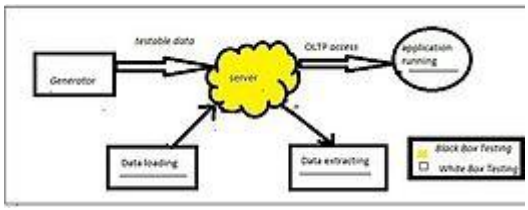
The final phase of building a database is realization, which includes testing and setting it up.
Databases, the collection of interconnected files on a server, storing information, may not deal with the same type of data, i.e. databases may be heterogeneous. As a result, many kinds of implementation and integration errors may occur in large database systems, which negatively affect the system's performance, reliability, consistency and security. Thus, it is important to test in order to obtain a database system which satisfies the ACID properties (Atomicity, Consistency, Isolation, and Durability) of a database management system.

Database testing usually consists of a layered process, including the user interface (UI) layer, the business layer, the data access layer and the database itself. The UI layer deals with the interface design of the database,[citation needed] while the business layer includes databases supporting business strategies. The most critical layer is the data access layer, which deals with databases directly during the communication process. Database testing mainly takes place at this layer and involves testing strategies such as quality control and quality assurance of the product databases. Testing at these different layers is frequently used to maintain consistency of database systems, most commonly seen in the following examples:

Data is critical from a business point of view. Companies such as Google or Symantec, who are associated with data storage, need to have a durable and consistent database system. If database operations such as insert, delete, and update are performed without testing the database for consistency first, the university risks a crash of the entire system.

Some companies have different types of databases, and also different goals and missions. In order to achieve a level of functionality to meet said goals, they need to test their database system.
The current approach of testing may not be sufficient in which developers formally test the databases. However, this approach is not sufficiently effective since database developers are likely to slow down the testing process due to communication gaps. A separate database testing team seems advisable.
Database testing mainly deals with finding errors in the databases so as to eliminate them. This will improve the quality of the database or web-based system.
Database testing should be distinguished from strategies to deal with other problems such as database crashes, broken insertions, deletions or updates. Here, database refactoring is an evolutionary technique that may apply.

Types of tastings and processes



Black box and white box testing in database testing
The figure indicates the areas of testing involved during different database testing methods, such as black-box testing and white-box testing.

Black Box testing in database testing

Black box testing involves testing interfaces and the integration of the database, which includes:
Mapping of data (including metadata)
Verifying incoming data
Verifying outgoing data from query functions
Various techniques such as Cause effect graphing technique, equivalence partitioning and boundary-value analysis.

With the help of these techniques, the functionality of the database can be tested thoroughly.
Pros and Cons of black box testing include: Test case generation in black box testing is fairly simple. Their generation is completely independent of software development and can be done in an early stage of development. As a consequence, the programmer has better knowledge of how to design the database application and uses less time for debugging. Cost for development of black box test cases is lower than development of white box test cases. The major drawback of black box testing is that it is unknown how much of the program is being tested. Also, certain errors cannot be detected.

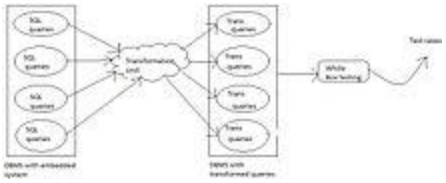
White Box Testing in database testing

White box testing mainly deals with the internal structure of the database. The specification details are hidden from the user.
It involves the testing of database triggers and logical views which are going to support database refactoring.
It performs module testing of database functions, triggers, views, SQL queries etc.
It validates database tables, data models, database schema etc.
It checks rules of Referential integrity.
It selects default table values to check on database consistency.

1. The techniques used in white box testing are condition coverage, decision coverage, statement coverage, cyclomatic complexity.

The main advantage of white box testing in database testing is that coding error are detected, so internal bugs in the database can be eliminated. The limitation of white box testing is that SQL statements are not covered.

The WHODATE approach for database testing



WHODATE approach for SQL statement transformation
While generating test cases for database testing, the semantics of SQL statement need to be reflected in the test cases. For that purpose, a technique called WHite bOx Database Application Technique "(WHODATE)" is used. As shown in the figure, SQL statements are independently converted into GPL statements, followed by traditional white box testing to generate test cases which include SQL semantics.

Four stages of database testing

- Set Fixture
- Test run
- Outcome verification
- Tear down

A test fixture describes the initial state of the database before entering the testing. After setting fixtures, database behavior is tested for defined test cases. Depending on the outcome, test cases are either modified or kept as is. The "tear down" stage either results in terminating testing or continuing with other test cases.

For successful database testing the following workflow executed by each single test is commonly executed:

Clean up the database: If the testable data is already present in the database, the database needs to be emptied.

Set up Fixture: A tool like PHPUnit will then iterate over fixtures and do insertions into the database.

Run test, Verify outcome and then Tear down: After resetting the database to empty and listing the fixtures, the test is run and the output is verified. If the output is as expected, the tear down process follows, otherwise testing is repeated.

Some problems in database testing

- The setup for database testing is costly and complex to maintain because database systems are constantly changing with expected insert, delete and update operations.
- Extra overhead is involved in order to determine the state of the database transactions.
- After cleaning up the database, new test cases have to be designed.[citation needed]
- An SQL generator is required to transform SQL statements in order to include the SQL semantic into database test cases.

Basic techniques

- SQL Query Analyzer is a helpful tool when using Microsoft SQL Server.
- One commonly used function,[vague] create_input_dialog["label"], is used to validate the output with user inputs.
- The design of forms for automated database testing, form front-end and back-end, is helpful to database maintenance students.
- Data load testing:
- For data load testing, knowledge about source database and destination database is required.
- Students check the compatibility between source database and destination database using the DTS package.
- When updating the source database, students make sure to compare it with the target database.
- Database load testing measures the capacity of the database server to handle queries as well as the response time of database server and client.[6]
- In database testing, issues such as atomicity, consistency, isolation, durability, integrity, execution of triggers, and recovery are often considered.

In order of secure operation I 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 I are to create the privilege system on SQL level, then user groups correspond to different roles. I can use the CREATE ROLE command to create user groups. Then with the help of the GRANT command I 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

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 Iide. 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) 415.

[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 Iide. 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-

[8] www.wikipedia.com

