Large-scale database modeling: Extended Er diagrams and Uml

Tong Feng
University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation
Feng, Tong, "Large-scale database modeling: Extended Er diagrams and Uml" (2002). UNLV Retrospective Theses & Dissertations. 1414.
https://digitalscholarship.unlv.edu/rtds/1414

This Thesis is brought to you for free and open access by Digital Scholarship@UNLV. It has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, Mi 48106-1346 USA
800-521-0600

UMI®

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
LARGE SCALE DATABASE MODELING:
EXTENDED ER DIAGRAMS AND UML

by

Tong Feng

Bachelor of Science
Nankai University, P.R.China
1996

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science Degree
Department of Computer Science
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
August 2002
Thesis Approval
The Graduate College
University of Nevada, Las Vegas

26 JULY 2002

The Thesis prepared by

TONG FENG

Entitled

LARGE SCALE DATABASE MODELING: EXTENDED ER DIAGRAMS AND UML

is approved in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative
ABSTRACT

Large Scale Database Modeling: Extended ER Diagrams and UML

by

Tong Feng

Dr. Kazem Tagva, Examination Committee Chair
Professor of Computer Science
University of Las Vegas, Nevada

This thesis is concerned with the team efforts to develop a large database to track medical information. An Extended Entity Relationship diagram is developed using UML notation to describe the design of the database. Special attention was given for features of EER diagram which can not easily be represented by ER diagram.
TABLE OF CONTENTS

ABSTRACT ............................................................................................................... iii

LIST OF FIGURES ................................................................................................. v

ACKNOWLEDGEMENTS .................................................................................... vi

CHAPTER 1 INTRODUCTION .......................................................................... 1

CHAPTER 2 DESIGN DATABASE ................................................................. 4
  Conceptual Design of Database ........................................................................ 4
  Entity Relationship Model ............................................................................... 5
    1. Entity Relationship Model ........................................................................ 6
    2. ER Diagram Represented By UML .................................................... 10
  Extended Entity Relationship Model ............................................................ 12
    1. Generalization/Specialization ............................................................. 13
    2. Aggregation ......................................................................................... 15
    3. EER Diagram Represented By UML ............................................. 16

CHAPTER 3 OMR DATABASE DESIGN USING EER ............................. 19
  Generalization and Specialization in the OMR Database .......................... 19
    1. Superclass and Subclass .................................................................... 19
    2. Type Inheritance ............................................................................... 24
    3. Specialization and Generalization Process ..................................... 25
    4. Specialization/Generalization Hierarchy ......................................... 30
    5. Constraints on Specialization/Generalization .............................. 30
  Aggregation and Composition in the OMR Database .............................. 33
    1. Aggregation ....................................................................................... 33
    2. Composition ...................................................................................... 35

CHAPTER 4 CONCLUSION AND FURTHER RESEARCH .................... 36

BIBLIOGRAPHY ................................................................................................. 38

VITA ...................................................................................................................... 40
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>The level structure of the ER Model</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Attributes defined on entity set University</td>
<td>9</td>
</tr>
<tr>
<td>2.3</td>
<td>Attributes defined on relationship set StudentUseComputer</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>One example for ER diagram represented by UML</td>
<td>12</td>
</tr>
<tr>
<td>2.5</td>
<td>The Aggregation of the Shipment entity set</td>
<td>16</td>
</tr>
<tr>
<td>2.6</td>
<td>One example for specialization/generalization in EER diagram represented by UML</td>
<td>18</td>
</tr>
<tr>
<td>2.7</td>
<td>One example for aggregation in EER diagram represented by UML</td>
<td>18</td>
</tr>
<tr>
<td>3.1</td>
<td>Superclass Person and its subclasses</td>
<td>20</td>
</tr>
<tr>
<td>3.2</td>
<td>Entries For Person Entity</td>
<td>21</td>
</tr>
<tr>
<td>3.3</td>
<td>Distinct entities (Patient, Physician, Witness, Technician and Official</td>
<td>22</td>
</tr>
<tr>
<td>3.4</td>
<td>Person superclass with its subclasses</td>
<td>23</td>
</tr>
<tr>
<td>3.5</td>
<td>Type inheritance of weak entity set in EER</td>
<td>25</td>
</tr>
<tr>
<td>3.6</td>
<td>Specialization of the Referral entity</td>
<td>27</td>
</tr>
<tr>
<td>3.7</td>
<td>Generalization of different site programs</td>
<td>29</td>
</tr>
<tr>
<td>3.8</td>
<td>Specialization of Company</td>
<td>32</td>
</tr>
<tr>
<td>3.9</td>
<td>Aggregation of medical examination</td>
<td>34</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

Thanks to Jeffrey Coombs for help with \LaTeX.

I would like to thank all of the staffs at the Information Science Research Institute for their discussions about this work as it progressed. Special thanks go to Julie Borsack and Jeffrey Coombs for instructing me both the project and the thesis preparation, and most of all to Dr. Kazem Taghva for introducing me to the field of database design.
CHAPTER 1

INTRODUCTION

The health care industry is one of the largest industries in the world. Health care is also the most information intensive industry in the world[15]. However, the healthcare industry has been one of the most backward industries in the adoption of information technology to underpin the huge task of healthcare information management. The "medical chart" has always served as the repository and the assemblage of medical information for over a century. It has been the well-accepted method for tracking a person's medical history throughout their lifetime[21]. The need for the electronic medical record arises from the need to be able to record patient health information more accurately and in greater detail than is possible with paper. And it is possible to be able to process it in a more efficient and error-free way. Electronically stored health record information has greater long-term utility than its paper predecessor, to the patient, clinicians, and to other parties aiming to improve health care in general.

There are several projects that are working to develop models that will be comprehensive, portable, and robust and at the same time, benefit both the physician and the patient for the ultimate goal of better health care. This is not a simple task. The electronic medical record needs to be medico-legally acceptable, comprehensive and secure. Whole records, or parts thereof, need to be transmitted between health care facilities for clinical purposes or when a patient moves, and to government and insurance information systems for financial or administrative processing. Clinicians, researchers, educators and administrators need to be able to create, modify and query electrical medical records using diverse tools. The Good Electronic Health Record
(GEHR) project has been working on electronic medical record representation for nearly fifteen years.

Although very much related to other medical record research projects, our project has a more narrow focus[21]. Our work is specific to occupational medicine (occ-med) which represents a concentration within a much larger arena. What's more, the core of our research centers around the conversion of existing medical charts to electronic form.

The Information Science Research Institute (ISRI)'s involvement in this project began with a government agency's need to review past employee medical records that span over 50 years[21]. The records are warehoused in several storage facilities across the country so the process of just locating the correct patient file can take several weeks. The next step in the process is to manually review the patient file to locate the sought after information — a pain-staking and time-consuming task. So an automated way of accessing this medical data is required.

The goal of the above project is to design a system specific to occ-med that takes as input hard copy medical data and produces correct, queryable medical information. Occ-med is a medical specialty dedicated to the prevention, diagnosis, treatment and rehabilitation of illnesses and injuries arising from work-related activity[15]. It identifies and supports outside medical consultation when an injury or illness is outside its scope of responsibility. Specific applications in occ-med include: identifying potential on-site risks, assessing fitness for work, communicating with primary care physicians and other clinical colleagues, promoting health, responding to medical emergencies, and monitoring employees for possible side-effects caused from their work environment. The key discriminate of occ-med is its direct relationship to the workplace.

Scanning technology and OCR was used to convert hard copy pages to electronic form. OCR typically refers to the recognition of machine printed characters which may or may not be a component of a particular form. Forms recognition is broader.
It uses several recognition modules to recognize the data, whether they be textual fill-in fields or a checkbox.

The database will be used to store the data recognized from the forms. We discovered early in our analysis that the complexity of the data's relationships required more than just a flat relational representation. Although the basic entity relationship (ER) concepts can model most features of an Occupational Medical Record (OMR) database, our model needed to embody superclass/subclass relationships, inheritance and other features that are not easily described in a traditional ER diagram. So it is necessary to describe some features of the OMR database using an Extended Entity Relationship (EER) diagram. Our work is to use EER diagram modeling the OMR database using Unified Modeling Language (UML) notation.

The ER data model is based on a perception of the real world that consists of a set of basic objects called entities and of relationships among these objects. It was developed to facilitate database design by allowing the specification of an enterprise schema, which represents the overall logical structure of a database. The ER model supported with additional semantic concepts is called the EER model[3]. Most features in our OMR database can be captured using the ER model. However, some aspects of the OMR database may not be expressed easily using the ER diagram. We used EER features of specialization, generalization, aggregation in our OMR database design phase.

Chapter 2 provides an introduction to the concept of semantic modeling and covers the concept of ER and EER. ER and EER diagrams represented by UML is explained. Chapter 3 describes the EER feature with the OMR database. Finally, chapter 4 states the conclusion of the study and offers prospects for further research.
CHAPTER 2

DESIGN DATABASE

The design of the database is one of the most important steps in the development of a computerized information system.

It is commonly accepted[4] [19] that the entire design of a database requires at least four separate but interdependent design steps[10].

1. In the requirements analysis, the part of the world that is to be modeled must be thoroughly analyzed to take into account the requests of potential database users.
2. In the conceptual design, the structure and behavior of the database have to be specified formally using information gained in the previous step.
3. In logical design, the resulting system-independent conceptual schema is mapped into a schema of an implemented data model like the relational one. This step is necessary to bridge the gap between rich conceptual structures used for conceptual modeling and processable structures for effective system implementation.
4. In physical design, questions of actual database implementation are treated.

Conceptual Design of Database

Among the above steps during the design of database, conceptual design plays a central role. The concept model represents a global view of data. It is the basis for the identification and description of the main data objects, avoiding details. The concept model is independent of hardware or software constraints. Rather than trying to represent the data as a database would see it, the concept model focuses on representing the data as the user sees it in the real world.
The goal of the concept model is to make sure that all data objects required by the database are completely and accurately represented. Because the concept model uses easily understood notations and natural language, it can be reviewed and verified as correct by the end users.

The concept model is also detailed enough to be used by the database developers to use as “blueprint” for building the physical database[8]. The information contained in the data model will be used to define the relational tables, primary and foreign keys, stored procedures, and triggers.

The conceptual database design is a very difficult problem, especially when the database is very large or complex. If a database is not designed properly, it may have a serious impact on the operations of the organization using the database. A poorly designed database will require more time in the long term. Without careful planning the database created will omit data required to create critical reports, produces results that are incorrect or inconsistent, and is unable to accommodate changes in the user’s requirements. Therefore database design is an important problem for data processing specialists as well as users and managers.

To describe the requirements of database users in a formal and complete manner, semantic data models are needed. But the notion of semantics must be regarded with caution in this context, since only few data models possess a proper formal mathematical semantics, such as Entity Relationship model[14], T AXIS[7], I FO[17] or the algebraic approach of Sernadas et al.[2]. The well-tried and widely accepted ER model is often considered as the most appropriate data model.

Entity Relationship Model

The entity relationship model is a high level data model or “conceptual” model. Since its creation by P.P.Chen in 1976[13], the ER model has played an important role in the fields of database design, information systems analysis, and object orientation.
The ER model adopts a natural view that the real world consists of entities and relationships. It includes some important semantic information about the real world. The model satisfies a high degree of data independence and is based on set theory and relation theory. It was developed to facilitate database design by allowing the specification of an enterprise schema, which represents the overall logical structure of a database.

The semantic aspect of the model lies in the attempt to represent the meaning of the data[20]. The ER model is extremely useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema.

In figure 2.1 the underlying philosophy of the ER model is represented. The database world is structured into levels[8]. The first level deals with entities, attributes and relationships, the second with a "graphical" representation of entities, attributes and relationships. The first level could be seen to correspond to the abstract world model and the second level to the logical schema.

1. Entity Relationship Model

At level one there are three basic notions that E-R data model employs: entity sets, relationship sets, and attributes.

(1) Entity sets

An entity is a (real or conceptual) object or an event in the real world that is distinguishable from all other objects and events. For example, each university is an entity. An entity has a set of properties. And the values for some set of properties

<table>
<thead>
<tr>
<th>Level 1:</th>
<th>Entities</th>
<th>Relationships</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Entity Sets</td>
<td>Relationship Sets</td>
<td>Value Sets</td>
</tr>
<tr>
<td></td>
<td>Attributes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level 2:</th>
<th>ER Diagram</th>
</tr>
</thead>
</table>

Figure 2.1: The level structure of the ER Model

<table>
<thead>
<tr>
<th>Level 1:</th>
<th>Entities</th>
<th>Relationships</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Entity Sets</td>
<td>Relationship Sets</td>
<td>Value Sets</td>
</tr>
<tr>
<td></td>
<td>Attributes</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level 2:</th>
<th>ER Diagram</th>
</tr>
</thead>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
may uniquely identify an entity. For example, a university name uniquely identifies one particular university in the enterprise. An entity set is a set of entities of the same type that share the same properties or attributes. The entities are classified into different entity sets \( E_i \) such as “employee”, “project” and “department”\[8\]. There is a predicate associated with each entity set that tests whether an arbitrary entity belongs to the set. Entities may belong to more than one entity set. That means entity sets are not mutually disjoint. For example, an entity that belongs to the entity set “female faculty” also belongs to the entity set “person”. An entity type is used to represent both a type of entity and the entity set that exists in the database.

(2) Relationship sets

A relationship set is a mathematical relation on \( n \geq 2 \) entity sets. If \( E_1, E_2, \ldots, E_n \) are entity sets, then a relationship set \( R \) is a subset of

\[
\left\{ [e_1, e_2, \ldots, e_n] \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n \right\}
\]

where \(< e_1, e_2, \ldots, e_n >\) is a relationship. The \( E_i \)'s and \( e_i \)'s in the above definition need not be different. A relationship is an association among several entities. For example, a relationship that associates a professor with a department can be defined. This relationship specifies that a professor works for a particular department. The role of an entity in a relationship expresses the function the entity performs in the relationship\[8\]. In a relationship set “marriage” defined between entities from the entity set “person”, e.g

\[
"marriage" = \{ [e_1, e_2] \mid e_1 \in "person", e_2 \in "person" \}
\]

The first element in the relationship appears in the role “husband”, the second in
the role "wife". A relationship set is a set of relationships of the same type.

(3) Attributes

Information about an entity can be expressed by a set of attribute-value pairs associated with the entity[8]. Examples of values are "Peter", "Bell", "yellow", "35" etc., and they are classified into mutually disjoint value sets such as "first name", "last name", "color", "inches" etc., There is a predicate associated with each value set which tests whether a value belongs to the set. A value in one set may be equivalent (in a real world sense) to a value in a different set. For example, "1" in value set "feet" is equivalent to "12" in value set "inches". Attributes are descriptive properties possessed by each member of an entity set.

An attribute of an entity set is a function that maps from the entity set into a value set. A set of attributes represents an entity[8].

\[
\text{attr}_1 : E_i \rightarrow V_{i1} \times V_{i2} \times V_{in}
\]
\[
\text{attr}_2 : R_i \rightarrow V_{i1} \times V_{i2} \times V_{in}
\]

In figure 2.2 the attributes defined on the entity set University are illustrated. The attribute name maps university entities into elements of the value set University Name. The attribute address maps from the entity set University into a pair City Name, Street Name of value sets. Tuition and fund both map from the entity set University into the value set Dollar. An attribute is always defined as a function. Therefore, it maps a given entity to a single tuple if a value set product is identified.
Relationships also may have attributes. In figure 2.3 the relationship StudentUseComputer is illustrated. The attribute usage which defines the number of hours a specific student $e_i$ uses a machine $e_j$ is an attribute of the corresponding relationship. It is neither an attribute of the Student nor the Computer entity set since its meaning depends on the relationship between the two, i.e., a pair $[e_i, e_j]$.

(4) Relationship constraints

Relationship constraints express the number of entities to which another entity can be associated via a relationship set[20]. Relationship constraints are most useful in describing binary relationship sets. Although occasionally they contribute to the description of relationship sets that involve more than two entity sets.

For a binary relationship set $R$ between entity sets $A$ and $B$, the relationship constraints must be one of the following: (1) One to one. An entity in $A$ is associated with at most one entity in $B$, and an entity in $B$ is associated with at most one entity

Figure 2.2: Attributes defined on entity set University
in A. (2) One to many. An entity in A is associated with any number of entities in B. An entity in B, however, can be associated with at most one entity in A. (3) Many to one. An entity in A is associated with at most one entity in B. An entity in B, however, can be associated with any number of entities in A. (4) Many to many. An entity in A is associated with any number of entities in B, and an entity in B is associated with any number of entities in A.

The appropriate relationship constraints for a particular relationship set is dependent on the real world situation that is being modeled by the relationship set.

2. ER Diagram Represented By UML

The Entity-Relationship Diagram is a diagrammatic technique associated with Entity-Relationship Model. An ER diagram can be represented by the Unified Modeling Language (UML).

The object data model includes many of the concepts proposed for semantic modeling[9]. As an object modeling methodology, UML is becoming increasingly popular in software design and engineering. Although it was developed mainly for software design, a major part of software design involves designing the databases that
will be accessed by the software module. Hence as an important part of UML, class diagrams are similar in many ways to ER diagrams. However the terminology often differs.

In UML class diagrams, a class is displayed as a box that includes three sections: the top section gives the class name; the middle section includes the attributes for individual objects of the class; and the last section includes operations that can be applied to these objects. Operations are not specified in ER diagrams. A composite attribute is modeled as a structured domain. A multivalued attribute will generally be modeled as a separate class.

Relationship types are called associations in UML terminology, and relationship instances are called links[9]. A binary association (binary relationship type) is represented as a line connecting the participating classes(entity types), may have a name. A relationship attribute, called a link attribute, is placed in a box that is connected to the association’s line by a dashed line. The (min, max) notation is used to specify relationship constraints, which are called multiplicities in UML terminology. Multiplicities are specified in the form min..max, and an asterisk (*) indicates no maximum limit on participation.

In figure2.4, each entity is shown as the upper part of the rectangle labeled with the name of the entity, which is normally a singular noun[3]. In UML, the first letter of each word in the entity name is upper case. illustrates the diagrammatic representation of the Staff and Branch entity types. Each relationship type is shown as a line connecting the associated entity types, labeled with the name of the relationship. Normally , a relationship is named using a verb or a short phrase including a verb. The first letter of each word in the relationship name is shown in upper case. A relationship is only labeled in one direction, which normally means that the name of the relationship only makes sense in one direction. So once the relationship name is chosen, an arrow symbol is placed beside the name indicating the correct direction.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
for a reader to interpret the relationship name. For example, the relationship named Manages as shown in Figure 2.4. The middle part of rectangle lists the name of the attributes associated with an entity. The name(s) of the primary key attribute(s) can be labeled with the tag {PK}. In UML, the name of an attribute is displayed with the first letter in lower case, if the name has more than one word, with the first letter of each subsequent word in upper case. The ER diagram in Figure 2.4 also shows the relationship constraints. To represent that a member of staff can manage zero or one branch, we place a 0..1 beside the Branch entity. To represent that a branch always has one manager, we place a 1..1 beside the Staff entity.

Extended Entity Relationship Model

Since the late 1970s there has been a rapid increase in the development of many new database applications that have more demanding database requirement than those of the traditional applications. As the basic concepts of ER modeling are often not sufficient to represent the requirements of the newer, more complex applications, this stimulated the need to develop additional 'semantic' modeling concepts. A number of new concepts have also been introduced into the ER model by various researchers, as in [12] [18] [5] [16], giving rise to the notion of the EER models. Smith and Smith [6] present the concepts of generalization and aggregation. The semantic data model of Hammer and Mcleod [11] introduced the concepts of class/subclass lattices.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
1. Generalization/Specialization

An entity type is used to represent both a type of entity, and the entity set that exist in the database. In many cases an entity type has numerous subgroupings of its entities[8]. Those subgroups of the entity are meaningful and they need to be represented explicitly because of their significance to the database. For example, the entities that are members of employee entity type may be grouped further into supervisor, hourlyEmployee, salariedEmployee, engineer, secretary, and so on. The set of entities in each of latter groupings is a subset of the entities that belong to the employee entity set, meaning that every entity that is a member of one of these subgroupings is also an employee. We call each of these subgroupings a subclass of the Employee entity type, and the Employee entity type is called the superclass for each of these subclasses. The relationship between a superclass C and any one of its subclasses S is called as a superclass/subclass relationship[9]. The superclass/subclass relationship is donated by:

\[ S \subseteq C \]

A Generalization is denoted by \( T_1 \mid T_2 \mid \cdots \mid T_n \) if \( T_1, T_2, \ldots, T_n \) are (generalized) entity sets.

And defines a new entity set \( T \) with the meaning

\[ t \in T \iff \exists T_i (1 \leq i \leq n \land t \in T_i) \]

That is, there exists for every entity in \( T \) at least one \( T_i \) which contains that entity. Generalization is a process of abstraction in which we suppress the difference among several entity types, identify their common features, and generalize them into
a single superclass of which the original entity types are special subclasses. For example, consider the entity types car and truck, they can be generalization into the entity type vehicle. Generalization is a term which is used to refer to the process of defining a generalized entity type from the given entity types[9].

A specialization \( Z = \{S_1, S_2, \ldots, S_n\} \) is a set of subclasses that have the same superclass \( G \); that is[8], \( G/S_i \) is a superclass/subclass relationship for \( i = 1, 2, \ldots, n \). \( Z \) is said to be mandatory if we always have

\[
\bigcup_{i=1}^{n} S_i = G
\]

Otherwise, \( Z \) is said to be optional.

\( Z \) is said to be disjoint if we always have

\[
S_i \cap S_j = \emptyset \text{ for } i \neq j
\]

Otherwise, \( Z \) is said to be overlapping.

Specialization process can be viewed as being functionally the inverse of the generalization process[9]. If we have an entity set Employee and want to use the specialization, we have to specify in the model roles, that define when an employee entity belongs to one or the other component entity set.

The process of generalization and specialization characterize entities by their similarities and differences. For example, suppose an organization categorizes the work it does into internal and external projects. Internal projects are done on behalf of some unit within the organization. External projects are done for entities outside of the organization. We can recognize that both types of projects are similar in that each involves work done by employees of the organization within a given schedule.
Yet we also recognize that there are differences between them. External projects have unique attributes, such as a customer identifier and the fee charged to the customer. In practice, it is likely that neither the generalization process nor the specialization process is followed strictly, but a combination of the two processes is employed.

In a Generalization/Specialization the attributes and relationships of the superclass are inherited by the subclasses.

2. Aggregation

Aggregation represents a 'has a' or 'is-part-of' relationship between entity types, where one represents the 'whole' and the other the 'part'][8].

The Aggregation is defined by: If \( T_1, T_2, \ldots, T_n \) are (generalized) entity sets, an Aggregation is denoted by \( < T_1, T_2, \ldots, T_n > \) or by \( < s_1 : T_1, s_2 : T_2, \ldots, s_n : T_n > \)

where \( s_1, s_2, s_n \) are called selectors which extract one of the component entity sets. The operation defines a new entity set \( T \) with the meaning

\[
t \in T \iff \exists t_1, t_2, \ldots, t_n \\
(t_1 \in T_1 \land t_2 \in T_2 \land \ldots \land t_n \in T_n \land < t_1, t_2, t_3, \ldots, t_n > = t)
\]

That is, the new entities are formed as tuples of entities from the component entity sets. To be meaningful the entity sets \( T_1, T_2, \ldots, T_n \) have to be part of some relationship, and this relationship will always be included in the representation of the generated entity set.

Attribute-value set pairs can be attached to the new entity set. It also can take part in any relationship. One example of an Aggregation operation is given in Figure 2.5. The new entity set Shipment is defined as an aggregation of the three entity sets Supplier, Part and Project with the new attributes shipDate and shippedQuantity.
There is an important difference between these two attributes, however. While shipmentDate cannot be thought of belonging to any component entity set, the shippedQuantity attribute clearly refers to 'parts'.

Composition is a specific form of aggregation that represents an association between entities, where there is a strong ownership and coincidental lifetime between the 'whole' and the 'part'.

The options to use aggregation and composition are subjective decisions. Aggregation and composition should only be used when there is a requirement to emphasize special relationships between entity types such as 'has-a' or 'is-part-of', which has implications on the creation, update, and deletion of these closely related entities.

3. EER Diagram Represented By UML

An EER diagram can be represented by the Unified Modeling Language (UML)[9]. (1) UML has a special notation for representing specialization/generalization[3]. For example, consider the specialization/generalization of the Staff entity into subclasses that represent job roles. The Staff superclass and the Manager, SalesPersonnel, and Secretary subclasses can be represented in an Enhanced Entity Relationship (EER) diagram as illustrated in Figure 2.6. The Staff superclass and the subclasses, being entities, are represented as rectangles. The subclasses are attached by lines to a triangle.
that points toward the superclass. The label below the specialization/generalization triangle, shown as `{Optional, Overlapping}`, describes the constraints on the relationship between the superclass and its subclasses. These constraints will be discussed in Chapter 3.

Attributes that are specific to a given subclass are listed in the lower section of the rectangle representing that subclass. For example, salesArea and carAllowance attributes are only associated with the SalesPersonnel subclass, and are not applicable to the Manager or Secretary subclasses. Similarly, we show attributes that are specific to the Manager (mgrStartDate and bonus) and Secretary (typingSpeed) subclasses.

Attributes that are common to all subclasses are listed in the lower section of the rectangle representing the superclass. For example, staffNo, name, position, and salary attributes are common to all members of staff and are associated with the Staff superclass. We can also show relationships that are only applicable to specific subclasses. For example, in Figure 2.6, the Manager subclass is related to the Branch entity through the Manages relationship, whereas the Staff superclass is related to the Branch entity through the Has relationship.

(2) UML represents aggregation by placing an open diamond shape at one end of the relationship line, next to the entity that represents the 'whole'. In Figure 2.7 this EER diagram displays two examples of aggregation, namely Branch Has Staff and Branch Offers PropertyForRent. In both relationships, the Branch entity represents the 'whole' and therefore the open diamond shape is placed beside this entity.

UML represents composition by placing a filled-in diamond shape at one end of the relationship line next to the entity that represents the 'whole' in the relationship. For example, to represent the Newspaper Displays Advert composition, the filled in diamond shape is placed next to the Newspaper entity, which is the 'whole' in this relationship.
Figure 2.6: One example for specialization/generalization in EER diagram represented by UML

Figure 2.7: One example for aggregation in EER diagram represented by UML
CHAPTER 3

OMR DATABASE DESIGN USING EER

The OMR database will be used to store the data recognized from the forms. Based on the complexity of the data's relationships in the forms, ER modeling concepts can not easily represent all the relationships in the OMR database. Besides ER modeling concepts, we use EER modeling concepts to describe other features such as the superclass/subclass relationship, type inheritance, and aggregation in the OMR database.

Generalization and Specialization in the OMR Database

In the OMR database we used generalization and specialization to capture the features such as superclass/subclass relationship and type inheritance.

1. Superclass and Subclass

A superclass is an entity type that includes one or more distinct subgroupings of its occurrences. A subclass is a distinct subgrouping of occurrences of an entity type. The relationship between a superclass and any one of its subclasses is called a superclass/subclass relationship.

An entity in a subclass represents the same 'real-world' object as in the superclass. For example in figure 3.1 a patient "John English" is also the person "John English". Hence the subclass member is the same as the entity in the superclass, but in a distinct, specific role.

An entity cannot exist in the database merely by being a member of a subclass. It must also be a member of the superclass. Such an entity can be included optionally
as a member of any number of the subclass. For example, a patient who is also an official belongs to the two subclasses Patient and Official of the Person type. On the other hand, not every member of a superclass need be a member of a subclass. For example, a person "Jim Bell" may not belong to any subclass under superclass Person. However in our OMR Database, every entity in a superclass is a member of some subclasses based on their role in database.

In the OMR database, there are many different types of persons in actual forms. We have three options as to how we best model members of Person.

(1) The first option is to represent all members of Person as a generalized Person Entity. In this way, we try to describe different types of Person with possibly different attributes within a single entity.

This option will cause two problems. The first problem is if all Person attributes and those specific to a particular role are described by a single Person entity, this
<table>
<thead>
<tr>
<th>personID</th>
<th>lastName</th>
<th>firstName</th>
<th>personRole</th>
<th>patientDOB</th>
<th>patientSex</th>
<th>physicianDegree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>English</td>
<td>Tim</td>
<td>patient</td>
<td>02/23/62</td>
<td>male</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Bell</td>
<td>Jason</td>
<td>patient</td>
<td>04/15/58</td>
<td>male</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Algebe</td>
<td>Alex</td>
<td>physician</td>
<td></td>
<td></td>
<td>M.D.</td>
</tr>
<tr>
<td>4</td>
<td>Harron</td>
<td>Tina</td>
<td>technician</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Jeffery</td>
<td>Tom</td>
<td>official</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.2: Entries For Person Entity

will result in a lot of nulls for the role specific attributes. For example, in figure 3.2 the Patient entity has special attributes patientDOB, patientSex and so on. These attributes are not shared by other members in the Person entity. Therefore in the actual database, the values for these special attributes for those non patient members in the Person entity will be null. On the other hand, the Physician entity has special attributes physicianDegree and physicianType. The values for attribute physicianDegree and physicianType for non physician members in the Person entity will also be null. It will cause a lot of waste in memory space when the database is implemented and used. This option will also cause another problem. Some members in the Person entity may have distinct relationships that are not appropriate for all members in the Person entity. For example, a member of Patient may be required to attend some kind of physical examination. But other members in the Person entity may not need to attend that kind of examination. In this model, we can not represent that only a subset of the members in the Person entity have a relationship with other entities. So it is not appropriate to represent different types of persons using a single Person entity.

(2) The second option is to create distinct entities Patient, Physician, Witness, Technician and Official. This option will overcome the two disadvantages of the first option. It is obvious that these distinct entities can be described by common attributes among them and special attributes associated with each entity. The special attributes associated with each distinct entity will not be empty any more when the database is
Figure 3.3: Distinct entities (Patient, Physician, Witness, Technician and Official implemented and used. And it can also represent those relationships associated with each distinct entity with other entities in database. However at the same time it will cause new problems. These distinct entities have similar concepts that each member in an entity is some kind of person. They possess some common attributes such as PersonID, lastName, firstName and middleInitial. And they have some similar relationships with the Address entity, the Phone entity and so on. However this option does not represent the commonality of attributes and relationships associated with each entity. Figure 3.3 does not show a clear picture of the relationship among these distinct entities.

(3) The third option is to represent the Patient, Physician, Witness, Technician and Official entities as subclasses of a Person entity. This option will overcome the disadvantages in the first two options. It is based on the commonality of attributes and relationships associated with each entity. All attributes of the Person entity are represented in the subclasses Patient, Physician, Witness, Technician and Official, including the primary key personID. The Patient entity includes those attributes associated with a patient. On the other hand, this entity does not include those attributes associated with a physician such as physicianDegree. And the Patient entity is associated with a distinct relationship, namely RelatedTo which a relationship between patient and correspondence, as are other subclasses. For example, physician
Figure 3.4: Person superclass with its subclasses

gets involves in a distinct relationship, namely GivenBy, which is a relationship between physician and Treatment. This option also adds more semantic information to the design. A superclass/subclass relationship is often called an IS A (or IS AN) relationship because of the way we refer to the concept. So in this design the assertions that say “a PATIENT IS A PERSON”, “A PHYSICIAN IS A PERSON” add significant semantic content in a concise form. Therefore the third option is the best option to model the information of the Person in OMR. See figure 3.4.
2. Type Inheritance

Because an entity in a subclass represents the same 'real-world' object as the superclass, entities that are members of subclasses inherit all the attributes of the superclass including the primary key. The entity also inherits the relationship of the superclass. But a subclass can have its own unique attributes and relationship. The type of an entity is defined by the attributes it possesses and the relationship types in which it participates. Therefore a subclass, with its own specific attributes and relationships together with all the attributes and relationships it inherits from the superclass, can be considered an entity type in its own right. For example, a member of the Patient subclass inherits all the attributes of the Person superclass such as personID, lastName, firstName and middleInitial.

There is one issue about type inheritance associated with the weak entity. In the OMR database, HearingConservationProgram is a SiteProgram and at the same time it is also the superclass of HearingConservationData. OccupationalNoiseExposure is a weak entity associated with the strong entity HearingConservationProgram. Since in actual forms, OccupationalNoiseExposure is part of HearingConservationProgram and it is a list of employer, workLocation, jobTitle, workYears, noiseSources and noiseSourcesTimePercentage (see Figure 3.5). OccupationalNoiseExposure can not exist without the program. And OccupationalNoiseExposure does not have sufficient attributes to form a primary key. Although each OccupationalNoiseExposure entity is distinct, each entity in OccupationalNoiseExposure may be shared by different HearingConservationPrograms. So we represent OccupationalNoiseExposure as the weak entity of HearingConservationProgram. For HearingConservationData, it is a HearingConservationProgram and it has special attributes and a relationship with Audiogram. This entity also possesses the same weak entity as its superclass HearingConservationProgram. The relationship between a strong entity and weak entity is one-to-many since we know a subclass inherits all the relationships in which the
superclass participates. As the subclass of HearingConservationProgram, HearingConservationData inherits the weak entity of HearingConservationProgram. Therefore OccupationalNoiseExposure is also the weak entity of HearingConservationData. And the relationship between HearingConservationData and OccupationalNoiseExposure is also one-to-many. In figure 3.5 we connect the subclass to the weak entity set directly using UML notation.

3. Specialization and Generalization Process

In the design phase of the OMR database a combination of the generalization process and specialization process is employed.

Specialization is a top-down approach to defining a set of superclasses and their related subclasses. When we apply the process of specialization on an entity, we attempt to identify the difference between the members of this entity such as members
with distinctive attributes and/or relationships. The specialization process allows us to do the following: (1) Define a set of subclasses of an entity type. (2) Establish additional specific attributes with each subclass. (3) Establish additional specific relationship types between each subclass and other entity types or other subclasses.

In the OMR database, there is an entity type called Referral used to describe the medical referral information. Referral with the entity types of ExternalReferral and InternalReferral has distinct attributes and relationships. Therefore we identify ExternalReferral and InternalReferral as the subclasses of Referral.

The set of subclasses is defined on the basis of some distinguishing characteristics of the entities in the superclass. Consider the entity set Referral with attributes referralID, referralType, referralFrom, referralTo, referralDate, treatment, and diagnosis. A referral is further classified as being one of the following: (1) ExternalReferral (2) InternalReferral

In this specialization we determine exactly the entities that will become members of each subclass by placing a condition on the value of some attribute of the superclass. Because the Referral entity type has an attribute referralType as shown in figure 3.6, we can specify the condition of membership in the ExternalReferral subclass by the predicate (referralType = 'ExternalReferral'), which we call the defining predicate of the subclass. ExternalReferral and InternalReferral subclasses are called predicate defined (or condition defined) subclasses.

There is another approach for determining membership in a subclass when we don't have a defining predicate of the subclass. In this case, membership is specified individually for each entity by the database user, not by any condition which may be evaluated automatically. In the OMR database, we never took this approach.

We may have several specializations of the same entity type based on different distinguishing characteristics. For example, one specialization of the Injury entity type has the set of subclasses {OccupationalInjury, NonOccupationalInjury} based
on type of injury. Another specialization of the Injury entity type may yield the set of subclasses \{InternalInjury, ExternalInjury\}; this specialization distinguishes among injuries based on the injured body part. In the OMR database, there are two actual forms which are related to Injury entity, one is about Occupational Injury, another is about nonOccupational injury. Besides the common attributes and relationships, each form has distinct attributes and relationships. So we choose the specialization based on type of injury. It is not necessary to have specialization of the Injury entity based on injured body part since there are no distinct attributes and relationships associated with either InternalInjury or ExternalInjury.
The process of generalization is a bottom up approach, which results in the identification of a generalized superclass from the original entity types. The generalization process can be viewed as being functionally the inverse of the specialization process. In the OMR database, there are seven different site programs RespiratorMedicalProgram, NuclearEmergencyProgram, IonizingRadiationProgram, DrugAlcoholScreeningProgram, FormaldehydeScreeningProgram, LeadScreeningProgram and HearingConservationProgram. They are initially represented as distinct entity types. If we apply the process of generalization on these entities, we attempt to identify common features of these entities. These entities share the same attributes programID and date. And more importantly they share two common relationships: Monitors and SurveillanceRelatedTo. Therefore we generalized the above seven programs into a single superclass in which the original entities are now subclasses as in Figure 3.7.
Figure 3.7: Generalization of different site programs
4. Specialization/Generalization Hierarchy

A subclass itself may have further subclasses specified on it forming a hierarchy or a lattice of specializations. A specialization hierarchy requires that every subclass participate in one superclass/subclass relationship. It has single inheritance. In contrast, a specialization lattice allows a subclass to participate in more than one superclass/subclass relationship. In the OMR database, the specialization of the Person entity has the set of subclasses {Patient, Physician, Witness, Technician and Official}. Official is a subclass of Person. In the real world there are two types of officials which are involved in the OMR database. Therefore Official is also a superclass of ClaimsPersonnel and Supervisor; this represents the real world constraint. In such a specialization hierarchy, a subclass inherits the attributes and relationships not only of its direct superclass but also of all its predecessor superclasses, all the way to the root the hierarchy. In this case, an entity in Supervisor inherits all the attributes and relationships of the Official and Person entities. It is possible to arrive at the same hierarchy from the other direction using the generalization process.

In the OMR database, there is no lattice of specialization included in the EER model since it is not necessary to capture the concept of multiple inheritance. The situation that a subclass with more than one superclass never happens in the OMR database.

5. Constraints on Specialization/Generalization

In general, we may have several specializations defined on the same supertype. In such a case, entities may belong to subclasses in each of the specializations. However, a specialization may also consist of a single subclass only, such as the specialization{LabMemo} of the superclass Correspondence.

There are two constraints that may apply to a specialization/generalization.

(1)The first one is the disjointness constraint, which indicates whether it is possible for a member of a superclass to be a member of one, or more than one, subclass. There
are two cases with regard to disjointness constraints. If the subclasses are disjoint then
an entity can be a member of at most one of the subclasses of the specialization. In
the OMR database the specialization of Exposure into subclass InternalExposure and
ExternalExposure is disjoint, which means that a member of Exposure must belong
to InternalExposure or ExternalExposure, but not both. If the subclasses are not
constrained to be disjoint, their sets of entities may overlap for the specialization of
Person entity, the subclasses overlap. The same real world entity may be a member of
more than one subclass of the specialization. For example, a person can be a member
of the Patient entity, at the same time, this person is also a member of the Official
entity.

(2) The second constraint on specialization is called the participation constraint, which
determines whether every member in the superclass must participate as a member of
a subclass. It may be mandatory or optional. Since every member of the Exposure
dentity must be either one kind of internal exposure or external exposure, the speciali-
zation has mandatory participation, which specifies that every entity in the superclass
must be a member of some subclass in the specialization. A superclass/subclass re-
lationship with optional participation specifies that a member of a superclass need
not belong to any of its subclasses. In the OMR database, if a specialization consists
of more than one subclass, then the specialization has mandatory participation. If
a specialization only consists of one subclass then this specialization has to be an
optional participation. Otherwise there is no need to classify the subclass as the
subclass of a superclass. For example, in fig3.8 in the specialization of Company, a
company need not belong to one member of the Contractor entity. If this specializa-
tion has mandatory participation, it means every member of Company entity must be
a member of the Contractor entity. The relationship between superclass{Company}
and subclass{Contractor} is 1:1. And in the real world the entity in the subclass
represents the same entity in the superclass. In this case, the superclass and subclass
have the same attributes and relationships. A member of Company may not be a member of the subgroup Contractor. Therefore the specialization must have optional participation when the specialization consists of only one subclass.

The disjointness and participation constraints of specialization and generalization are independent. Hence, we have the following four possible constraints on specialization: (1) mandatory and disjoint (2) optional and disjoint (3) mandatory and overlapping (4) optional and overlapping.

The correct constraint is determined from the real world meaning that applies to each specialization. However, a superclass that was identified through the generalization process usually is mandatory, because the superclass is derived from the subclasses and hence contains only the entities that are in the subclasses.

Certain insertion and deletion rules apply to specialization (and generalization) as a sequence of the constraints specified earlier. Some of these rules are as follows: (1) Deleting an entity from a superclass implies that it is automatically deleted from all the subclasses to which it belongs. (2) Inserting an entity in a superclass implies that the entity is mandatorily inserted in all predicate-defined subclasses for which
the entity satisfies the defining predicate. (3) Inserting an entity in a superclass of a mandatory specialization implies that the entity is mandatorily inserted in at least one of the subclasses of the specialization.

Aggregation and Composition in the OMR Database

In the OMR database we used aggregation to capture the feature of ‘is-part-of’ relationship.

1. Aggregation

During the design phase of the OMR database sometimes we need to model a ‘has-a’ or ‘is-part-of’ relationship, in which one entity represents a larger entity (the ‘whole’), consisting of smaller entities (the ‘part’). This special kind of relationship is called an aggregation. Aggregation does not change the meaning of navigation across the relationship between the whole and its parts, nor does it link the lifetimes of the whole and its parts. In the OMR database Medical Examination plays an important role. When a patient takes a medical examination, they need to take different kinds of examinations such as a vision test, a hearing test and so on. The relationship between MedicalExamination and the different special examinations is therefore aggregation. The MedicalExamination entity represents the ‘whole’ and the different special examinations represent the ‘parts’. The relationship we want to represent is that MedicalExamination has a vision test or MedicalExamination has a hearing test. It is different from the superclass/subclass relationship. In superclass/subclass relationship, the information captured was that a vision test is a medicalExamination. Because we want to explain that a patient needs to take different special examinations when he or she takes a medical examination, the relationship should be the ‘whole’ and ‘part’ relationship. See figure3.9
Figure 3.9: Aggregation of medical examination
2. Composition

Aggregation is entirely conceptual and does nothing more than distinguish a ‘whole’ from a ‘part’. Composition is a stronger form of aggregation in which a ‘part’ belongs to only one ‘whole’ and exists only as part of the ‘whole’. In a composite, the ‘whole’ is responsible for the disposition of the ‘parts’, which means that the composition must manage the creation and destruction of its ‘parts’. In other words, an object may only be part of one composite at a time. In the OMR database, there is no information which should be captured using composition.

In the OMR database, we used most of the concepts of EER to model the features that can not be represented easily by ER model. We used generalization and specialization to represent the different persons which play different roles in the database, and other superclass/subclass relationships such as the superclass Authorization entity with its subclasses MedicalInformationAuthorization, and SurgicalTreatmentAuthorization and so on. We used aggregation to represent the relationships between a ‘whole’ medical examination and different special examinations.
CHAPTER 4

CONCLUSION AND FURTHER RESEARCH

The various features of the EER model offer us numerous choices in how best to represent the enterprise being modeled. Concepts and objects may be represented by entities, relationships or attributes. Aspects of the overall structure of the enterprise may be best described using weak entity sets, generalization, specialization or aggregation[1]. We finished the EER diagram using the features of the EER model for the conceptual design of the OMR database. And we found that the features of the EER model helped us to build a more semantic model for the OMR database.

The EER model we established for the OMR database is a conceptual representation of the data structures that are required by a database. The data model focuses on representing the data as the user sees it in the real world. It serves as a bridge between the concepts that make up real world events and objects and the physical representation of those concepts in a database.

The EER model and relational database design are abstract, logical representations of real world enterprises. Because they employ similar design principles, we can convert an EER design into a relational design. A database that conforms to an EER diagram can be represented by a collection of tables[20]. Converting a database representation from an EER diagram to a table format is the basis for deriving a relational database design from an EER diagram. For each entity set and for each relationship set in the database, there is a unique table that is assigned the name of the corresponding entity set or relationship set. Each table has multiple columns, each of which has a unique name.
The relational database design was formally introduced by E. F. Codd in 1970 and has evolved since then[1]. The design provides a simple, yet rigorously defined, concept of how users perceive data. The relational database design represents data in the form of two dimensional tables. Each table represents some real world person, place, thing, or event about which information is collected. The organization of data into relational tables is known as the logical view of the database. That is, the form in which a relational database presents data to the user and the programmer.

A basic understanding of the relational database is necessary to effectively use relational database software such as Oracle, Microsoft SQL Server, or even personal database systems such as Access or Fox, which are based on the relational database[1].

The EER model for the OMR database will be represented by a set of tables. The goal of the relational database design is to generate a set of relation schemas that allows users to store information without unnecessary redundancy yet also allows users to retrieve information easily. One approach is to design schemas that are in an appropriate normal form. Normalization is the process of efficiently organizing data in a database[20]. There are two goals of the normalization process: eliminate redundant data (for example, storing the same data in more than one table) and ensures data dependencies make sense (only storing related data in a table). Both of these are worthy goals as they reduce the amount of space a database consumes and ensure that data is logically stored. The normalization process for our OMR database only needs 1NF, 2NF and 3NF.
BIBLIOGRAPHY


VITA

Graduate College
University of Nevada, Las Vegas

Tong Feng

Local Address:
1600E. University Ave #148
Las Vegas, NV 89119

Degrees:
Bachelor of Science, Microbiology, 1996
Nankai University, P.R.China

Thesis Title:
Large Scale Database Modeling: Extended ER Diagrams and UML

Thesis Examination Committee:
Chairperson, Kazem Taghva, Ph. D.
Committee Member, Dr. Thomas Nartker, Ph. D.
Committee Member, Dr. Wolfgang W. Bein, Ph. D.
Graduate Faculty Representative, Dr. Henry Selvaraj, Ph. D.