Large-scale database modeling: Discovering attributes, entities, and relationships

Ying Yang
University of Nevada, Las Vegas
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
LARGE SCALE DATABASE MODELING:
DISCOVERING ATTRIBUTES, ENTITIES, AND RELATIONSHIPS

by

Ying Yang

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
The Thesis prepared by

YING YANG

Entitled

LARGE SCALE DATABASE MODELING: DISCOVERING ATTRIBUTES, ENTITIES, AND RELATIONSHIPS.

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

Graduate College Faculty Representative
ABSTRACT

Large Scale Database Modeling: Discovering Attributes, Entities, and Relationships

by

Ying Yang

Dr. Kazem Taghva. 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. Entity relational model approach is taken to study an extensive set of forms for structure discovery. This approach has led to thousands of attributes and hundreds of entities and relationships. A meta-database is used to manipulate this data for further design.
# TABLE OF CONTENTS

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

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

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

CHAPTER 2 ENTITY RELATIONSHIP DATA MODEL ........................... 5
  What is A Data Model .................................................................................... 5
  Methodology ..................................................................................................... 6
  Database Design .............................................................................................. 6
  Data Modeling as Part of Database Design ............................................... 7
  Requirements Analysis .................................................................................... 7
  Components of A Data Model ........................................................................ 8
  Why is Data Modeling Important? ............................................................ 9
  ER Modeling ..................................................................................................... 9
  Basic Constructs of ER Modeling ............................................................ 10
  Summarize ........................................................................................................... 17

CHAPTER 3 DATA PLANNING, ANALYSIS AND MODELING . . . . 18
  Sample Set ........................................................................................................ 18
  Sample Classification ....................................................................................... 20
  Attributes Extraction ....................................................................................... 22
  Entity Generation .............................................................................................. 25
  Relationship Establishment .......................................................................... 29
  UML Notation ................................................................................................. 34

CHAPTER 4 CONCLUSION AND FURTHER WORK ......................... 36

BIBLIOGRAPHY .................................................................................................... 37

VITA ............................................................................................................................ 39

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
<table>
<thead>
<tr>
<th>FIGURE</th>
<th>DESCRIPTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>The procedures of the ER model design</td>
<td>19</td>
</tr>
<tr>
<td>3.2</td>
<td>The Attribute Record Form</td>
<td>26</td>
</tr>
<tr>
<td>3.3</td>
<td>Attributes grouping table</td>
<td>27</td>
</tr>
<tr>
<td>3.4</td>
<td>Entity document table</td>
<td>29</td>
</tr>
<tr>
<td>3.5</td>
<td>Fan trap problem in ER model</td>
<td>31</td>
</tr>
<tr>
<td>3.6</td>
<td>Solution to fan trap problem in ER model</td>
<td>32</td>
</tr>
<tr>
<td>3.7</td>
<td>Chasm trap problem in ER model</td>
<td>33</td>
</tr>
<tr>
<td>3.8</td>
<td>Solution to Chasm trap problem in ER model</td>
<td>33</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

First of all, I would like to express my hearty thanks to Dr. Kazem Taghva for his invaluable guidance, advice and encouragement throughout my thesis research work. His suggestions throughout this work are of great help.

I would like to thank Dr. Tomas Nartker, Dr. Wolfgan Bein, and Dr. Henry Selvaraj for serving on the committee and spending their valuable time to evaluate this work.

I am grateful to Julie Borsack and Jeffrey Coombs for help with my thesis structure and Latex. I would like to thank all of the staff at the Information Science Research Institute for their support during my work.

Finally, my thanks go to my husband Mr. De Chen. It is impossible to finish this thesis without him.
CHAPTER 1

INTRODUCTION

Until recently, keeping medical records in paper files was the only way to create a medical record repository. Since a person's medical records has to be kept at least for his/her lifetime, the total number of personal medical record in each institute can easily grow to be very large. Locating a patient's file is very inefficient and time-consuming. With the advent of electronic means for sharing information, it becomes feasible to store medical data electronically, which could be much more accessible, more useful, and more timely to the physicians and medical professionals.

The creation of this electronic medical data system requires a large database system that can create, add, revise and retrieve a specific file. The main goal of the occupational medical records project is to design a system specific to occupational medicine that takes as input hard copy medical data and produces correct, queryable medical information. The technologies involved include optical character recognition (OCR), image processing, form recognition, database and interface display. The process of this medical capture and retrieval system is:

1. The information from the forms is extracted using form and OCR recognition and save it in a database.

2. Modeling the database requires the use of object relational concepts modeled using extended entity relationship diagram (EER).

3. Once the information is in the database, it can be queried and modified as:
electronic medical chart, patient information and image of form, editable patient record form.

The first things needed in this project is to capture all the medical record data, to examine related technologies, and to automate the data capture process. In the data capturing process, the hard copy pages of the medical record is converted to electronic form. The scanning and OCR (optical character recognition) technologies[19] are used for this purpose. The hard copies of medical record are digitized by scanners. In digitizing a hard copy page, the scanner senses the variations in light intensity, the patterns in the input page, which become analog signals. These analog signals are then digitized into digital images which are represented as matrices of 1s and 0s that replicate the hard copy. The most important thing in this step is the quality of the scanned image[21][22], it directly affects the processing steps followed, such as recognition. Close attention needs to be paid to the characteristics of the hard copy and to the scanning process so that flaws in the image are minimized.

The medical record data being dealt with in this project can be summarized as a collection of hundreds of unique forms with millions of instances. An instance can be some pre-printed information representing labels for fields in the form. An instance can also be user-filled information which can be typed, hand written, or check marked by patient, physician, witness, technician, official etc. Only instances with user-filled information are considered in this project. The initial processes in data capturing in the system are image processing and forms classification. Each form in the patients file belongs to a specific category (tabs) such as Audio, EKGs, etc.

OCR is specifically the technique that recognizes machine printed characters which may or may not be a component of a particular form. For real forms recognition, more techniques may be involved, usually they include several recognition modules: handprint recognition or intelligent character recognition (ICR), optical mark recognition (OMR), barcode recognition and possibly handwritten recognition. In such a
complete forms recognition system, zones on the form are identified and bound to the appropriate recognition module. Thus data, in almost any form, can be recognized.

The data, when recognized from the forms, is stored in the database. Because of the complexity of the data's relationships, the database requires a more complicated representation. For the medical record database system, object relational model which embodies the features that are not easily described in a traditional relationship (ER) diagram should be considered.

The form processing capability of Adobe's Portable Document Format can be used for a PDF display interface. In an editable fashion, the field data is acquired from the patient forms and stored into the database. The PDF file generated from the system is an image of the form (can be obtained from a blank form), with field values filled in its correct positions. This display system has the ability to edit all the fields in a form, which is actually a query interface that allow the user to type queries (known field values) directly into the form, and all other related information for that form can then be searched, located and displayed.

The Electronic Medical Chart of a patient should appear as they would in the hardcopy folder with some patient forms on the left and some on the right. The viewer can then determine what aspect of the patient they want to investigate from various pull-down menus, such documents dealing with X-rays, or audiograms, etc. Once a query, such as patient's name, has been submitted, all patient files relevant to that query are displayed.

For a patient information and form image query, once a form is selected, an image of the form is displayed along with the patient ID, employment location, and first and last name. This image may be enlarged and the OCR text corresponding to the image can be viewed[20].

This work is concentrated on the establishment of an ER and EER model for our system. The ER model is one of the most used techniques in database development.
It is advisable to start with an ER model for a complex database system, which naturally becomes the backbone of the real system.

In building the ER model, a subset of ninety-five distinct personal medical records in forms is used as the sample data. Based on these initial data pretreatment, relationships between entities are established. Finally, the established ER model is represented using UML (Uniform Modeling Language).

In chapter two, the theoretical background of data modeling and the ER model are expounded. The explanation of the data model, the methodology of creating a data model, the design process, the requirements analysis, the components of the data model, and the importance of data modeling along with the properties and the basic constructs of ER modeling are explained. Chapter 3 is the heart of this thesis with detailed explanation of attribute, entity and relationship discovery. Chapter 4 of this thesis is the conclusion and consideration of future work.
CHAPTER 2

ENTITY RELATIONSHIP DATA MODEL

Database design has been accomplished with a variety of approaches, including top-down, bottom-up, and combined methodologies. The traditional approach has been a low-level bottom-up activity synthesizing data elements into normalized relations using the inter-data element dependencies resulting from the requirements analysis. Although the traditional process is vital to the design of databases, its complexity can be overwhelming to the point where practical designer often do not bother to master it or even use it with any regularity. In practice, typically a few basic relations are defined by the requirements analysis process, and then a combination of the top-down and bottom-up approach is used. The combined approach has recently become much more popular, because of the introduction of a well established conceptual design tool, the entity-relationship model into this process. The entity-relationship (ER) model has been most successful as a tool for communication between the designer and the end user during the requirement analysis and conceptual design phases because of its ease of understanding and its convenience in representation[10].

What is A Data Model

A data model is a conceptual representation of the data structures that are required by a database. The data structures include the data objects, the associations between data objects, and the rules which govern operations on the objects. As the name implies, the data model focuses on what data is required and how it should be organized rather than what operations will be performed on the data. A data model
is independent of hardware or software constraints. Rather than try to represent the data as a database would see it, 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 processes and the physical representation of those concepts in a database.

Methodology

There are two major methodologies used to create a data model: the Entity-Relationship (ER) approach and the Object Model. This thesis uses the Entity-Relationship approach.

Database Design

The design process roughly follows five steps:

1. planning and analysis
2. conceptual design
3. logical design
4. physical design
5. implementation

The data model is one part of the conceptual design process.

Data modeling must be preceded by planning and analysis. Planning defines the goals of the database, explains why the goals are important, and sets out the path by which the goals will be reached. Analysis involves determining the requirements of the database. This is typically done by examining existing documentation and interviewing users.
Data Modeling as Part of Database Design

The data model is one part of the conceptual design process. The data model focuses on what data should be stored in the database. The data model is used to design the relational tables. Data modeling is proceeded by planning and analysis. The effort devoted to this stage is proportional to the scope of the database. The planning and analysis of a database intended to serve the needs of an enterprise will require more effort than one intended to serve a small workgroup.

The information needed to build a data model is gathered during the requirements analysis. Although not formally considered part of the data modeling stage by some methodologies, in reality the requirement analysis and the diagramming part of the data model are done at the same time.

Requirements Analysis

The goals of the requirements analysis are:

1. to determine the data requirements of the database in terms of primitive objects
2. to classify and describe the information about these objects
3. to identify and classify the relationships among the objects
4. to identify rules governing the integrity of the data

Information needed for the requirement analysis can be gathered in several ways:

1. review of existing documents - such documents include existing forms and reports, written guidelines, job descriptions, personal narratives, and memoranda. Paper documentation is a good way to become familiar with the organization or activity you need to model[11].

2. interviews with end users - these can be a combination of individual or group meetings. Try to keep group sessions to under five or six people. If possible,
try to have everyone with the same function in one meeting. Use a blackboard, flip charts, or overhead transparencies to record information gathered from the interviews.

3. review of existing automated systems - if the organization already has an automated system, review the system design specifications and documentation

In this thesis, the first method is used for the requirement analysis.

The requirements analysis is usually done at the same time as the data modeling. As information is collected, data objects are identified and classified as either entities, attributes, or relationships. They are assigned names, and defined using terms familiar to the end-users. The objects are then modeled and analyzed using a diagram. If the model is not correct, it is modified, which sometimes requires additional information to be collected. The review and edit cycle continues until the model is certified as correct.

An effective data model completely and accurately represents the data requirements of the end users. It is simple enough to be understood by the end user yet detailed enough to be used by a database designer to build the database. The model eliminates redundant data. It is independent of any hardware and software constraints and can be adapted to changing requirements with a minimum of effort. Data modeling is a bottom up process.

Components of A Data Model

The data model gets its inputs from the planning and analysis stage. Here the analysts collect information about the requirements of the database by reviewing existing documentation (The sample set used in this thesis is hundreds of medical record forms).

The data model has two outputs. The first is a diagram which represents the data structures in a pictorial form. Because the diagram is easily learned, it is a valuable
tool to communicate the model to the end-user. The second component is a data
document. This document describes in detail the data objects, relationships, and
rules required by the database.

Why is Data Modeling Important?

Data modeling is probably the most labor intensive and time consuming part of the
development process. Why bother especially if you are pressed for time? A common
response by practitioners who write on the subject is that you should no more build
a database without a model than you should build a house without blueprints.

The goal of the data model is to make sure that all data objects required by the
database are completely and accurately represented. Because the data model uses
easily understood notations and natural language, it can be reviewed and verified as
correct by the end-users.

The data model is also detailed enough to be used by the database developers to
use as a “blueprint” for building the physical database. The information contained in
the data model will be used to define the relational tables, primary and foreign keys,
stored procedures, and triggers. A poorly designed database will require more time in
the long-term. Without careful planning you may create a database that omits data
required to create critical reports, produces results that are incorrect or inconsistent,
and is unable to accommodate changes in the user’s requirements.

ER Modeling

Databases continue to become increasingly complex. To model, design, and main­
tain these complex databases, it is more important than ever to understand the
structure of the data that had been used.

The Entity-Relation Model (ER) is the most common method used to build d ata
models for relational databases[17].

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Entity-relationship (ER) modeling have traditionally been an essential part of the overall planning and maintenance of an organization's information resources[2].

The Entity-Relation Model (ER) is the most common method used to build data models for relational databases[17]. The Entity-Relationship (ER) model was originally proposed by Peter in 1976 as a way to unify the network and relational database views. Simply stated, the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity-Relationship diagram which is used to visually represent data objects. Since Chen wrote his paper the model has been extended and today it is commonly used for database design. For the database designer, the utility of the ER model is:

1. it maps well to the relational model. The constructs used in the ER model can easily be transformed into relational tables.

2. it is simple and easy to understand with a minimum of training. Therefore, the model can be used by the database designer to communicate the design to the end user.

3. In addition, the model can be used as a design plan by the database developer to implement a data model in a specific database management software[17].

Basic Constructs of ER Modeling

There are three basic notions that the ER data model employs: attributes, entity sets, and relationship sets[3].

Attributes:

An attribute is any detail that serves to identify, qualify, classify, quantify, or otherwise express the state of an entity occurrence or a relationship. Attributes are specific pieces of information which need to be known or held. An attribute is either required or optional. When it's required, we must have a value for it. a value must be
known for each entity occurrence. When it's optional, we could have a value for it, a value may be known for each entity occurrence. The attributes reflect the need for the information they provide. In the analysis meeting, the participants should list as many attributes as possible. Later they can weed out those that are not applicable to the application, or those the client is not prepared to spend the resources on to collect and maintain. The participants come to an agreement on which attributes belong with an entity, as well as which attributes are required or optional.

Attributes can be classified as below:

1. **simple**. A simple attribute is an attribute composed of a single component with an independent existence. Simple attributes cannot be further subdivided into smaller components.

2. **composite**. A composite attribute is an attribute composed of multiple components, each with an independent existence.

3. **single-valued**. A single-valued attributes is an attribute that holds a single value for each occurrence of an entity type.

4. **multi-valued**. A multi-valued attribute is an attribute that holds multiple values for each occurrence of an entity type.

5. **derived**. A derived attribute is an attribute that represents a value that is derivable from the value of a related attribute or set of attributes, not necessary in the same entity type.

6. **null attributes**. A null value is used when an entity does not have a value for an attribute[17].

**Keys:**

A key is an attribute or collection of attributes. Attributes can be classified as identifiers or descriptors. Identifiers, more commonly called keys, uniquely identify
an instance of an entity. A descriptor describes a non-unique characteristic of an entity instance.

Properties of key:

A key has two properties:

1. Unique identification. In each tuple of a relation the value of the key must uniquely identify that tuple.

2. Non-redundancy. No attribute in the key can be removed without destroying property 1.

What governs the selection of the primary key?

1. It should not be possible for an attribute to have an undefined value

2. The number of attributes should be a minimum.

Keys can be classified as below:

1. candidate key. A candidate key is the minimal number of attributes. whose value(s) uniquely identify each entity occurrence.

2. primary key. An entity may have more than one candidate key. Primary key is a candidate key that is selected to uniquely identify each occurrence of an entity type.

3. composite key. In some cases, the key of an entity type is composed of several attributes. whose values together are unique for each entity occurrence but not separately[5].

Entities:

The next step in modeling is to identify the entities involved in that process. An entity is a thing or object of significance to the business. whether real or imagined.
about which the business must collect and maintain data, or about which information needs to be known or held[16]

Entities are the things that hold particular interest for you in your database - you can think of them as 'subjects' to be covered - but 'entity' is a more accurate term. They are a classification of things and should have a precise definition. They can be concrete things or abstractions. It is important to identify them because they are potentially the tables in the database[18].

Entities are the principal data object about which information is to be collected. Entities are usually recognizable concepts, either concrete or abstract, such as person, address, things, or events which have relevance to the database. An entity is analogous to a table in the relational model[17].

Property of Entities:

The data in any database represents a model of an entity in the real world. All entities in the real world are unique. If they are not then they are not separate entities. When we design a table to contain data about the real world we must extract sufficient features of that entity to define each member of the table uniquely. This means that there must be enough columns to uniquely identify and discriminate between rows in a table[8].

Entity occurrence:

An entity occurrence (also called an instance) is an individual occurrence of an entity. An occurrence is analogous to a row in the relational table.

Entity classification:

Entities are classified as strong or weak (in some methodologies, the terms used are independent or dependent, respectively). A strong entity is one that does not rely on another for identification. A weak entity is one that relies on another for identification.

Special Entity Types:
Associative entities (also known as intersection entities) are entities used to associate two or more entities in order to reconcile a many-to-many relationship.

Subtypes entities are used in generalization hierarchies to represent a subset of instances of their parent entity, called the supertype, but which have attributes or relationships that apply only to the subset. Associative entities and generalization hierarchies are discussed in more detail in EER model.

Relationships:

After two or more entities are identified, the participants determine if a relationship exists between the entities. A relationship is any association, linkage, or connection between the entities of interest to the business; it is a significant association between two entities, or between an entity and itself or between a group of entities. Each relationship has a name, an optionality (optional or mandatory), and a degree (how many). A relationship is described in real terms.

Rarely will there be a relationship between every entity and every other entity in an application. If there are only two or three entities, then perhaps there will be relationships between them all. In a larger application, there are not always relationships between one entity and all of the others. Assigning a name, an optionality, and a degree to a relationship helps confirm the validity of that relationship. If you cannot give a relationship all these things, then perhaps there really is no relationship at all[16].

Classifying Relationships:

Relationships are classified by their degree, connectivity, cardinality, direction, type, and existence. Not all modeling methodologies use all these classifications.

Degree of a Relationship:

The degree of a relationship is the number of entities associated with the relationship. The n-ary relationship is the general form for degree n. Special cases are the binary and ternary, where the degree is 2 and 3, respectively. Binary relationships.
the association between two entities is the most common type in the real world. A recursive binary relationship occurs when an entity is related to itself. An example might be "some employees are married to other employees".

A ternary relationship involves three entities and is used when a binary relationship is inadequate. Many modeling approaches recognize only binary relationships. Ternary or n-ary relationships are decomposed into two or more binary relationships.

Mapping Cardinalities:

Mapping cardinalities, or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set.

Mapping cardinalities are most useful in describing binary relationship sets. For a binary relationship set R between entity sets A and B, the mapping cardinality 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 B is associated with at most one entity in A. Denoted as: 1..1.
2. one to many. An entity in A is associated with any number of entities in B. An entity B, however, can be associated with at most one entity in A. Denoted as: 1..*.
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. Denoted as: *..1.
4. many to many. An entity in A is associated with any number of entities in B, and entity in B is associated with any number of entities in A. Denoted as: *..*.

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

Participation:
A constraint represents whether all entity occurrences are involved in a particular relationship (referred to as mandatory participation) or only some (referred to as optional participation)[7].

Direction:

The direction of a relationship indicates the originating entity of a binary relationship. The entity from which a relationship originates is the parent entity; the entity where the relationship terminates is the child entity. The direction of a relationship is determined by its connectivity. In a one-to-one relationship the direction is from the independent entity to a dependent entity. If both entities are independent, the direction is arbitrary. With one-to-many relationships, the entity occurring once is the parent. The direction of many-to-many relationships is arbitrary[18].

Entity relationship model problems:

In each case a problem arises because of a misinterpretation of the meaning of certain relationships. The term connection trap will be used to describe any such interpretation error: the terms fan trap and chasm trap will be applied to particular cases to indicate the cause of the problem. Any conceptual model will contain potential connection traps. Their existence must be recognized so that they can be eliminated[14].

1. Fan trap. The fan trap can arise when three entity types are related to each other, but their relationship are represented by a linear structure[6]. That is: A fan trap may exist where two or more (1..*) relationship fan out from the same entity. The pathway between certain entities is ambiguous. This fan trap can be resolved by restructuring the original entity relation model to represent the correct association these entities.

2. Chasm trap. The chasm trap can arise when three entity types are related to each other by partly optional relationships. That is: A chasm traps may occur where there are one or more relationship with a minimum multiplicity of zero
(that is optional participation) forming part of the pathway between related entities. The pathway does not exist between certain entity occurrences. To solve this problem, the missing relationship needs to be identified.

Summerize

Basic concepts of data modeling has been discussed in this chapter along with entity relationship modeling. What is a data model, components of a data model and the importance of data model are discussed in detail as well.

For ER modeling, the basic constructs of ER modeling, such as attributes, keys, entities and relationships as well as their classifications and properties are expounded within the chapter. Also some relationship modeling problems are mentioned and analyzed.
CHAPTER 3

DATA PLANNING, ANALYSIS AND MODELING

For building a database system, it is crucial to establish an accurate and clear data model that can describe the information completely. In order to reach this goal, the procedures for data planning and analysis become extremely important. For the purpose of capturing the information from forms as completely as possible, attributes are extracted before any entity is generated, and then relationships are defined based on the associations among entities. Some special care must be taken to guarantee the completeness of information collection. The procedures for the ER model design are shown in figure 3.1.

Sample Set

The sample of this project is a set of hard copies of forms which contain a variety of information about patients medical record.

The total number of forms is 477 of which 95 are distinct. Some forms are similar, but are a different version. All the forms are only for 5 patients:

patient 1 has 4 forms: (3 left, 1 right)
patient 2 has 12 forms: (6 left, 6 right)
patient 3 has 111 forms: (56 left, 55 right)
patient 4 has 62 forms: (33 left, 29 right)
patient 5 has 288 forms: (186 left, 102 right)

(* left and right indicate the position of the document in the original folder)

Consider the number of forms of hundreds of thousands of people, it will be
Figure 3.1: The procedures of the ER model design
incredibly huge. Forms in hard copies have problems of accessibility. An electronic
database is inevitable and necessarily in the future.

Sample Classification

Data analysis starts from data classification. Since the sample set is very large,
data analysis becomes a very difficult task. Classification is a necessary step, which
is not only helpful in analyzing and capturing information from these forms, but also
useful for attribute extraction and later entity generation. Classification groups the
forms into different categories.

Content based classification:

Content based classification is done according to the contents of each form. In
this approach, forms of similar events, are classified into the same category, such as
medical history, examination laboratory work, etc.

Some of the forms in the same category can be combined into one entity because
they are related to the same event. This will be discussed later in detail in the entity
generation section.

In practice, if the content of two forms are related to the same event, they are put
into the same category. For example: the form called Accident Injury Report and the
form called Employers Report of Industrial Injury both are injury reports. They can
be put into the same category.

In this project, the 95 distinct forms are classified into seventeen categories. Ex­
ample forms are listed below for some categories:

1. Medical history

   *Statement of interim medical history (pat3 R36)*
   
   *Report of medical history (pat3 R51)*

2. Examination
3. Radiology

Radiographic report (pat3 L24)
Radiological safety investigation report (pat3 L28)
Radiologic consultation request/report (pat5 L186)

4. Lead

Lead Screening Program Occupational and Medical History (pat5 L37)
Lead workers (pat5 L45)
Blood lead analysis results* (pat5 L50)

5. Drug/alcohol

Drug screen and/or alcohol analysis request authorization (pat4 L17)
Alcohol and drug screen urine analysis request (pat4 L21)
Federal drug testing custody and control form (pat4 L18)

6. Treatment

Occupational treatment record (pat2 R3)
Non-occupational treatment record (pat1 L2)

7. Vaccine/immunization

Record of hepatitis B vaccination (pat4 L9)
Schedule of immunization (pat4 L5)
Immunization record (pat4 R3)

Format based classification:
Format based classification is another approach. In this classification method, the format or appearance of a form is the criteria for classification. The following is the rough classification result of the 95 distinct forms with this method.

1. Some of the forms are in table format. They are the tables that let a patient or physician or someone else such as technician, nurse or witness fill out the information needed.

2. Some records are in text form, such as: questionnaires, memorandums, letters, emails etc.

3. Some records are in image form, such as: ECG images, audiogram images, and respirograph images.

4. And some records are a mix of text, tables and images.

Attributes Extraction

Attributes naming convention:

For the consistency of the attributes extracted by different team members, the following name conventions for our modeling are specified:

1. For attribute names of more than one word, the first word should start with a lower case letter and all words following the first start with an upper case letter. For one word attribute, it starts with a lower case letter. e.g. labDataReceivedTime

2. Attribute names must be nouns and always be singular (not plural). e.g. collectionFacility

3. Attribute names must be descriptive and understandable to others not looking at the forms. e.g. jobChangedThisYearFromLastYearDescription

4. Attributes should clearly communicate what the entity represents as well as distinguish it from other attributes with similar names. e.g. prescriptionMedicationStatus

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
and \textit{prescriptionMedicationDescription}. Although these two attributes have similar names, they can be distinguished each other by their name. The first one is a Boolean attribute. The value of it is yes or no to the question whether or not the patient took the prescription medication. The second one is a string attribute. It may give a list of prescription medications taken by the patient.

\textbf{Attribute capture from forms:}

Since the number of forms is large, capturing attributes from forms becomes very tedious and time consuming. Each form should be examined very carefully and all the information in the form must be captured, except for what is pre-printed on the form that is not filled out by the person or which give machine settings for a certain images. In this aspect, there is no uniform approach to this problem. Different methods need to be used to capture attributes case by case. These different ways to capture attributes from forms are summarized below:

1. Usually each item in a form represents an attribute, most of them are simple attributes. Such as: \textit{patientSSN}, \textit{patientDOB}, \textit{patientSex} etc.

2. In a form that contains images, one image represents one attribute. Such as: \textit{ECGImage}, \textit{audiogramImage}, \textit{respirographImage}.

3. Sometimes several extracted attributes are combined into one new attribute and the extracted attribute becomes the value of the new attribute reducing the number of attributes. For example: two attributes called \textit{homePhone} and \textit{workPhone} which are the extracted attributes in the form are combined into one new attribute called \textit{phoneType}: the homePhone and the workPhone become the value of the attribute phoneType. Each combination is recorded in the document to prevent information loss.

4. Some items in a form with a check box are of enumeration type. For example:
maritalStatus for a patient is an enumeration type, the value of it can be single, married, divorced, or widowed.

5. There are some multi-valued attributes, such as address and phone. Because a person can have zero or more addresses and phone numbers, thus there can be a list of addresses or phone numbers for a certain person. This kind of attribute become entities. In our database model, address and phone are entities. There exists a relationships directed person to address and from person to phone. and the cardinality of that relationship can be zero to many. (Entity and relationship will be discussed later.)

6. For some questionnaires, there are many questions under a certain categories. For each question, there is a value attached to it. We deal with this kind of situation as composite attributes and multi-valued attributes. For example, there are eighteen categories in the entity RespiratorMedicalProgram. There are at most 14 questions belonging to each category. The value of each question could be yes or no. Sometimes a description needs to be provided if a yes answer is chosen. The attributes in the RespiratorMedicalProgram are:

respiratorQuestionnaireCategory[0-18]
  respiratorCategoryList[0-14]
    respiratorQuestionnaireType
    respiratorMedicalQuestionnaireValue
    respiratorMedicalQuestionnaireDescription
    respiratorCategoryComment
respiratorReview
  physicianReview
  physicianAction
  physicianComment
7. Some attributes can be derived from other attributes in the same entity; these attributes are derived attributes. For example: \texttt{patientAge} can be derived from \texttt{patientDOB}. Thus, \texttt{patientAge} is a derived attribute.

\textbf{Attribute archive:}

An attribute description form is designed to record all information about a specific attribute. This description form contains twenty five items including the source, characteristics, and access limit of an attribute. This form is filled out for each attribute extracted, which generates the necessary archive for information tracing. Figure 3.2 is a description form to be filled out for the attributes archive. There are a total of approx 3000 attributes of which 1600 are distinct ones in this archive.

\textbf{Entity Generation}

\textbf{Attribute grouping:}

The first step of entity generation in this project is attribute grouping. A proper name is attached to the attribute group afterward. For each form, tens or even hundreds of attributes may be used to describe the information contained in it. Because of the diversity of attributes a form can have, it is not feasible to represent a whole form by one entity. Usually, a single form is represented by several entities. To put these attributes into groups helps to define the entities: some of the groups may become an entity for the whole database. Example groups of attributes in a certain form are shown figure 3.3.

\textbf{Entity identification:}

To identify entities, groups of attributes from various forms that are related to the same event are combined. For example, almost every form has an attribute group called \texttt{Patient}. Thus all groups related to Patient can be combined into one entity.
Figure 3.2: The Attribute Record Form
<table>
<thead>
<tr>
<th>Entities</th>
<th>Attributes</th>
</tr>
</thead>
</table>
| Patient                   | patientLastName  
                           | patientFirstName  
                           | patientMiddleInitial  
                           | patientSSN  
                           | patientDOB  
                           | patientSex  |
| Address (addressType: patient) | patientAddressStreet  
                           | patientAddressCity  
                           | patientAddressState  
                           | patientAddressZipCode  
                           | patientAddressRFD  |
| CurrentOccupation          | agency (companyName)  
                           | organizationUnit  
                           | positionTitle  |
| Examination                | examinationPurpose  
                           | examinationDate  |
| ExaminingFacility          | examiningFacility  
                           | examiningFacilityStreet  
                           | examiningFacilityCity  
                           | examiningFacilityState  
                           | examiningFacilityZipcode  |
| HealthCondition            | presentOrPastHealthCondition  
                           | presentOrPastHealthConditionResponse  
                           | presentHealthConditionQuestion  
                           | presentHealthConditionResponse  
                           | everHadHealthConditionQuestion  
                           | everHadHealthConditionResponse  |

Figure 3.3: Attributes grouping table
The total number of entities generated is 82.

All entities are generated from grouping by the following criteria:

1. A singular noun is used for the entity name. The first letter of each word in the entity name is upper case. e.g. MedicalHistory.

2. Entities are divided into weak or strong entities[13]. For example, the entity Dependent is a weak entity of the entity Patient; Patient here is a strong entity. Since a dependent won't exist if a patient does not exist and Patient can exist on its own.

3. Entities are also classified as supertypes or subtypes (generalization or specification). Among the entities that are grouped from the attributes extracted from the forms, there may be some person type entities, such as Patient, Physician, Technician, Witness, etc. Some common attributes may exist among these subtype entities, such as lastName, firstName and middleInitial, etc. Thus a supertype entity Person is formed, containing these common attributes from the subtype. (EER model)

**Key attribute definition:**

If an attribute or several attributes uniquely define an entity, they can be selected as the primary key. For example: the attribute classCode in entity WageData can uniquely define the entity, thus it can be selected as a primary key. Sometimes all the attributes in an entity are selected as keys. In this case, an artificial key needs to be created instead of selecting all the attributes as keys. Here is an example: in the entity VisionTest, there are three attributes: visionTestIdType, visionResultOrScore and leftOrRightEye. all three attributes should be selected as keys to uniquely define the entity VisionTest, therefore an artificial attributes visionTestId is created as a primary key to uniquely defined the entity VisionTest. Thus, a composite key is
Entities | Attributes | Synonyms
--- | --- | ---
VisionTest | visionTestID PK visionTestType | distantVision
 |  | correctedDistantVision
 |  | nearVision
 |  | correctedNearVision
 |  | colorVisionTest
 |  | nightVisionTest
 |  | depthPerceptionTest
 |  | correctedRefractionForDistantVision
 |  | correctedRefractionForNearVision
 |  | eyeAccommodation
 |  | redLensTest
 |  | intraocularTension
 |  | fieldOfVision

Figure 3.4: Entity document table

Documentation:

The database model contains two parts: the document (or a data dictionary) and the diagram. In order to prevent information loss in the diagram, every entity defined is kept in a document. These documents record each attribute of the entity as well as the labels and synonyms of the attributes. Figure 3.4 is an example of a document.

Relationship Establishment

We now deal with objects that are not classified as entities or attributes, but represent associations among entities. When entities are defined, relationships among these entities need to be established.

Relationship assignment:

For every relationship between entities, the following are specified:

1. Name of the relationship. Normally, a relationship is named using a verb or
a short phrase including a verb. Again, the first letter of each word in the relationship name is in upper case. Whenever possible, a relationship name should be unique in our data model. Such as: LeadsTo between entities Injury and InjuryComplication. TreatBy between entities Injury and Treatment[9].

2. Degree of the relationship. In this model, most of the relationships are defined as binary. A ternary relationships are relationships among three entities. Ternary relationships bring great complexity to the implementation. In defining a ternary relationship, great care needs to be taken. They are used as infrequently as possible in our modeling. An example ternary relationship. PatientReferredBy, is used between Patient, Physician and Referral, since a patient is referred by a physician with a referral. There are relationships between Patient and Physician, Patient and Referral, Physician and Referral. These three relationships exist at the same time. That is why a ternary relationship is used between these three entities.

3. Direction of the relationship. The relationship is associated in a direction that indicates that the relationship only has one direction. For example, the relationship IsIdentifiedBy between Person and Identification is from Person to Identification. It makes sense that a Person IsIdentifiedBy Identification. In this model, directions are usually defined for binary relationships, but not for ternary relationships.

4. Cardinality of the relationship (usually for binary relationships). Relationship cardinality refers to the number of entity instances involved in the relationship. For example: Person Signs zero or more Signatures. The relationship cardinality is: 0..*. An Injury can be ReportedBy one InjuryReport. The relationship cardinality is 1..1[4].

5. Attributes of the relationship. Some relationships have their own attributes.
For example: the relationship *WorksFor* between *Person* and *Contractor* has its own attributes: *startDate* and *endDate*. It is necessary to know when a person starts and ends his or her contract. Another example is for ternary relationship *ExaminedBy* among *MedicalExamination*, *Person* and *Patient*. It also has its own attributes called *examinerRole*. This attribute makes it clear what kind of person, physician or technician, has participated in this relationship. In other words, attributes *examinerRole* specifies the person who gives the examination to the patient.

**Relationship problems:**

Some problems may arise when creating an ER model. These problems are referred to as connection traps, and normally occur due to a misinterpretation of the meaning of certain relationships. Two main types of connection traps are:

1. Fan Trap. Where a model represents a relationship between entity types but the pathway between certain entity occurrences are ambiguous. Consider the relationship in figure 3.5:

It is impossible to give a specific answer to the question: To which address
does the phone number P1 belong? The answer can be two addresses A1 or A2. Thus it is ambiguous. This fan trap can be resolved by restructuring the original model to represent the correct association between these entities as figure 3.6:

2. Chasm Trap. Where a model suggests the existence of a relationship between entity types but a pathway does not exist between certain entity occurrences[1]. Figure 3.7 shows a chasm trap problem.

If we attempt to answer the question: At which branch is the property number p2 available? we are unable to answer this question as this property is not yet allocated to a member of staff working at a branch. The inability to answer this question is considered to be a loss of information and is the result of a chasm trap. The multiplicity of both the staff and PropertyForRent entities in the Oversees relationship has a minimum value of zero which means that some properties cannot be associated with a branch through a member of staff. Therefore to solve this problem we need to identify the missing relationship which in this case is the Offers relationship between the Branch and PropertyForRent entities. Figure 3.8 shows the correct association between these entities.
Figure 3.7: Chasm trap problem in ER model

Figure 3.8: Solution to Chasm trap problem in ER model
UML Notation

In this data model, UML notation is used to represent the EER model. The EER model is beyond the scope of this thesis and it will not be discussed here in detail. But there is something that should be mentioned.

The UML notation is rich and full bodied. It is comprised of two major subdivisions. There is a notation for modeling the static elements of a design such as classes, attributes, and relationships. There is also a notation for modeling the dynamic elements of a design such as objects, messages, and finite state machines[12]. UML contains 9 different types of diagrams:

1. Use case
2. Class (static)
3. Sequence
4. Activity
5. Collaboration
6. State
7. Component
8. Deployment
9. Package

In this model we present some of the aspects of the static modeling notation. Static models are presented in diagrams called: Class Diagrams[15].

The UML notation with entities, attributes and relationships is presented below:

1. In UML, each entity is shown as rectangle labeled with the name of the entity.
2. Each relationship type is shown as a line connecting the associated entity types labeled with the name of the relationship. A relationship is only labeled in one direction. So once the relationship name is chosen, an arrow symbol is placed beside the name indicating the correct direction for a reader to interpret the relationship name.

3. The UML notation uses a diamond to represent relationships with degrees higher than binary. In this case, the directional arrow normally associated with the name is omitted.

4. Attributes are listed in the entity rectangle. The first attribute(s) to be listed is the primary key for the entity type. The name(s) of the primary key attributes(s) can be labeled with the tag PK.
CHAPTER 4

CONCLUSION AND FURTHER WORK

In this thesis, an EER model was proposed for an occupational medical information system. The EER model is an extension of the traditional ER model. In building the complete database model, thousands of attributes were extracted and hundreds of entities were generated from a set of ninety five distinct personal medical records. Relationships between these entities were defined. Finally, the complete database model was represented by a UML class diagram.

The current UML model represents the intermediary step in the design of our database. The next step is to translate this model into an actual relational or object relational schema followed by normalization steps. In order to completely describe the semantics of this database, many dependencies and constraints need to be implemented.
BIBLIOGRAPHY


VITA

Graduate College
University of Nevada, Las Vegas

Ying Yang

Local Address:
1600 E. University Ave #126
Las Vegas, NV 89119

Thesis Title:
Large Scale Database Modeling: Discovering Attributes, Entities, and Relationships

Thesis Examination Committee:
Chairperson. Kazem Taghva. Ph. D.
Committee Member. Dr. Thomas Nartker. Ph. D.
Committee Member. Dr. Wolfgan Bein. Ph. D.
Graduate Faculty Representative. Dr. Henry Salvaraj. Ph. D.