Improving Conceptual Modelling in Database Design

Tauqeer Hussain

School of Arts and Sciences

A thesis submitted to the Department of Computer Science, Lahore University of Management Sciences in partial fulfillment of the degree of Doctor of Philosophy

July 2006
I hereby recommend that the thesis submitted by Mr. Tauqeer Hussain entitled “Improving Conceptual Modelling In Database Design” be accepted in partial fulfillment of the requirements for the degree of Ph.D. in Computer Science.

Dr. Shafay Shamail
Advisor
To

my parents,

my uncle Sh. Ashique Hussain (late)

and

uncle Muhammad Aslam (late)
Acknowledgements

Proofs of theorems in this thesis are formal but acknowledgments are not. I mean every word in here!

Undoubtedly, I would like to acknowledge the extensive help and support provided by my advisor Prof. Dr. Shafay Shamail throughout the course of this research. He was a great learning source for shaping up ideas into structured boundaries and getting my research acknowledged in reputed conferences and journals. At the same time, I cannot forget the technical support from Prof. Dr. Mian M. Awais who was always available to me for discussions and advice. It was due to him that the research became a “continuous process improvement” for me. Both Dr. Shamail and Dr. Awais contributed many useful comments to earlier drafts and revisions of the thesis which improved its quality. I am also thankful to Prof. Dr. Shahab Baqai and Prof. Dr. Asim Karim who encouraged me and provided valuable feedback on the research from time to time. Since this thesis has now been revised in the light of suggestions and comments received by the international reviewers on this research, I am indebted to the reviewers who spared their precious time and gave valuable comments to enhance the quality of the thesis.

Had it not been Prof. Dr. Muhammad Ali Maud and Prof. Dr. Ashraf Iqbal, the Ph.D. would have been a dream otherwise. It was Dr. Maud who, like always, pushed me into the deep water and let me learn how to swim. He actually motivated rather persuaded me to join LUMS as a Ph.D. student. Dr. Ashraf Iqbal was kind enough to allow me working at my previous status as adjunct faculty at LUMS while proceeding with the Ph.D. program. This helped me a lot in terms of financial support. And, yes, one course
“Advanced Algorithms” that he taught equipped me to analyze my ER diagrams in terms of multi-color graphs.

I am grateful to Prof. Dr. Syed Zahoor Hassan and Prof. Dr. Anwar Khurshid for their kind support and encouragement during the course of study and research. During my stay at LUMS, I had the privilege of working with several faculty and students with whom interaction was always a pleasure and source of learning. It is all due to faculty of Department of Computer Science who buoyed my spirits to complete my work in time. I feel indebted to Prof. Dr. Shahid Masud who asked a “stupid” question (he used this word) during my proposal defence which opened another window of quantitative analysis in our research. One part of this thesis is in response to his “stupid” question.

I am extremely thankful to LUMS for providing full funding to participate in an international conference and to present my research. I am also sincerely thankful to Pakistan Science Foundation, Govt. of Pakistan for providing a travel grant to participate in another international conference. Without this help, it was not possible to present my research at international conferences like OOPSLA (Object Oriented Programming, Systems, Languages and Applications).

I must say I could not reach this stage without all that I gained at Dawood Hercules Chemicals Limited as a software engineer. It was that experience and experimentation when I identified some of the problems in the existing database design process. Collecting memories of those days, I can never forget Dr. Humayun Mian who really
inspired me to study computer science and operations research. He inculcated in our in-house development team the real logical thought process required for computer science discipline which paid me throughout the rest of my life. I am grateful to Mr. Hussain Dawood, the Managing Director, who always encouraged me and provided every kind of logistic and financial support to implement innovative ideas. Throughout my career in that organization, I find myself fortunate to have Kh. Ahmad Arsalan and Kh. Ahmad Usman who never behaved as my bosses but colleagues who were fond of involving in technical discussions, suggesting new ideas, motivating and getting the job done.

I have a high regard for my wife and my children who very patiently tolerated my continual absence from home. I can understand how desperately they are now looking forward to getting me back soon.

Last but not least, I am extremely thankful to all the support staff at LUMS especially Asif Karim and Zulfiqar Ali who kept me waking while writing this thesis. In the end, I would like to state that I am proud of completing this research, by the Grace of Almighty, leading towards award of the highest degree, in my own country and further being the first Ph.D. graduate of LUMS – one of the most prestigious universities of Pakistan.
Abstract

Conceptual Modeling is one of the most important stages of the database (DB) design methodology. A number of approaches for conceptual modeling have been devised in the literature amongst which the Entity-Relationship (ER) modeling technique is extensively used. Since the quality of a conceptual model impacts the quality of the end product, our research focuses on how the quality of an ER model can be improved. We have identified modeling problems in the existing ER modeling technique and have suggested an approach which solves these problems. The result is an improved ER model which closely represents the real-world problem thereby improving the semantic representation. Our proposed approach incorporates real-world constraints that can be described in the form of functional dependencies. This approach applies schema transformations iteratively for which a new set of rules has also been defined. New constructs namely single-valued relationship attribute and multi-valued relationship attribute have also been proposed for improving semantics of the relationship types in an ER model. The impact of the proposed approach on later stages of the database design methodology has also been studied which shows that the resulting relational database satisfies higher normal forms as compared to the existing technique. Quantitative aspect of measuring improvement in the quality of a conceptual model is also an integral part of the research. For this purpose, we have proposed new metrics called completeness index, normalization index, and overall quality index. Completeness index is further refined by applying fuzzy logic and thus a fuzzy completeness index is proposed. We have also defined quantitative metrics for the structural complexity of an ER model in terms of correctness and modifiability. These metrics help us compare the quality of two ER models quantitatively and objectively. We have shown with several examples the efficacy
of our approach and proposed metrics. The ultimate result is a better database design and improved database designers’ productivity.
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1NF</td>
<td>First Normal Form</td>
</tr>
<tr>
<td>2NF</td>
<td>Second Normal Form</td>
</tr>
<tr>
<td>3NF</td>
<td>Third Normal Form</td>
</tr>
<tr>
<td>BCNF</td>
<td>Boyce-Codd Normal Form</td>
</tr>
<tr>
<td>BRC</td>
<td>Business Requirements and Constraints</td>
</tr>
<tr>
<td>CA</td>
<td>Complex Attribute</td>
</tr>
<tr>
<td>CI</td>
<td>Completeness Index</td>
</tr>
<tr>
<td>DB</td>
<td>Database</td>
</tr>
<tr>
<td>DBMS</td>
<td>Database Management System</td>
</tr>
<tr>
<td>DFD</td>
<td>Data Flow Diagram</td>
</tr>
<tr>
<td>EER</td>
<td>Extended Entity-Relationship</td>
</tr>
<tr>
<td>ER</td>
<td>Entity-Relationship</td>
</tr>
<tr>
<td>ERD</td>
<td>Entity-Relationship Diagram</td>
</tr>
<tr>
<td>FCI</td>
<td>Fuzzy Completeness Index</td>
</tr>
<tr>
<td>FD</td>
<td>Functional Dependency</td>
</tr>
<tr>
<td>HERM</td>
<td>Higher-order Entity-Relationship Model</td>
</tr>
<tr>
<td>MVRA</td>
<td>Multi-Valued Relationship Attribute</td>
</tr>
<tr>
<td>NI</td>
<td>Normalization Index</td>
</tr>
<tr>
<td>NK</td>
<td>Non-key</td>
</tr>
<tr>
<td>ORM</td>
<td>Object-Role Model</td>
</tr>
<tr>
<td>QI</td>
<td>Overall Quality Index</td>
</tr>
<tr>
<td>R&lt;sub&gt;step&lt;/sub&gt;</td>
<td>Rise in the (stair) step</td>
</tr>
<tr>
<td>RAD</td>
<td>Rapid Application Development</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Database Management System</td>
</tr>
<tr>
<td>SDLC</td>
<td>Systems Development Life Cycle</td>
</tr>
<tr>
<td>SDM</td>
<td>Semantic Database Model</td>
</tr>
<tr>
<td>SVRA</td>
<td>Single-Valued Relationship Attribute</td>
</tr>
<tr>
<td>TAS</td>
<td>Tauqueer-Awais-Shamail</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language</td>
</tr>
</tbody>
</table>
# List of Figures

<table>
<thead>
<tr>
<th>Fig. 1.1: The Systems Development Life Cycle (SDLC)</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 1.2: An Overview of DB Design Process (Source: [Elmasri and Navathe 2004])</td>
<td>7</td>
</tr>
<tr>
<td>Fig. 1.3: Traditional Database Design Approach</td>
<td>10</td>
</tr>
<tr>
<td>Fig. 1.4: Proposed Database Design Approach</td>
<td>13</td>
</tr>
<tr>
<td>Fig. 2.1: Chen’s ER Diagram (Source: [Chen 1976])</td>
<td>32</td>
</tr>
<tr>
<td>Fig. 2.2: A Composite Attribute (Name)</td>
<td>34</td>
</tr>
<tr>
<td>Fig. 2.3: A Multi-valued Attribute (Phone)</td>
<td>35</td>
</tr>
<tr>
<td>Fig. 2.4: An Entity Type Designed without Hierarchy</td>
<td>36</td>
</tr>
<tr>
<td>Fig. 2.5: An Entity Type Designed with Hierarchy</td>
<td>37</td>
</tr>
<tr>
<td>Fig. 2.6: Total and Partial Relationship Constraints</td>
<td>40</td>
</tr>
<tr>
<td>Fig. 2.7: A Weak Relationship</td>
<td>41</td>
</tr>
<tr>
<td>Fig. 2.8: Participation Constraints and Relationship Attribute</td>
<td>44</td>
</tr>
<tr>
<td>Fig. 2.9: A Specialization with Disjoint Constraint</td>
<td>45</td>
</tr>
<tr>
<td>Fig. 2.10: A Shared Subclass</td>
<td>46</td>
</tr>
<tr>
<td>Fig. 2.11: A Category</td>
<td>46</td>
</tr>
<tr>
<td>Fig. 2.12: A BRC Notation</td>
<td>53</td>
</tr>
<tr>
<td>Fig. 3.1: Transforming Partial Attributes(Source:[Assenova and Johannesson 1996])</td>
<td>73</td>
</tr>
<tr>
<td>Fig. 4.1: An Entity Type: Sale_Order</td>
<td>91</td>
</tr>
<tr>
<td>Fig. 4.2: An Alternate ERD for Fig. 4.1</td>
<td>92</td>
</tr>
<tr>
<td>Fig. 4.3: An Entity Type having a Composite Key</td>
<td>98</td>
</tr>
<tr>
<td>Fig. 4.4: Improving Fig. 4.3 when Part of a Key Determines a Non-Key Attribute</td>
<td>99</td>
</tr>
<tr>
<td>Fig. 4.5: Attribute Dependencies in a Composite Attribute</td>
<td>101</td>
</tr>
<tr>
<td>Fig. 4.6: An Improvement in Fig. 4.5</td>
<td>102</td>
</tr>
<tr>
<td>Fig. 4.7: Part of a Composite Attribute Determines Another Part</td>
<td>103</td>
</tr>
<tr>
<td>Fig. 4.8: An Improvement in Fig. 4.7</td>
<td>103</td>
</tr>
<tr>
<td>Fig. 4.9: A Single NK Attribute Determines Another NK Attribute</td>
<td>106</td>
</tr>
<tr>
<td>Fig. 4.10: An Improvement in Fig. 4.9</td>
<td>106</td>
</tr>
<tr>
<td>Fig. 5.1: ERD Causing Violation of 2NF</td>
<td>120</td>
</tr>
<tr>
<td>Fig. 5.2: Modified ERD Eliminating Violation of 2NF</td>
<td>121</td>
</tr>
</tbody>
</table>
List of Tables

Table 2.1  Summary of ER Notations ................................................................. 31
Table 2.2  Summary of EER Notations ............................................................... 47
Table 2.3  Evolution of ER Model ............................................................... 54
Table 3.1  Moody’s Quality Factors ............................................................... 77
Table 3.2  Structural Complexity Measures ..................................................... 82
Table 5.1  Nomenclature for Algorithm 5.1 ...................................................... 116
Table 7.1  Summary of Quality Metrics for the Given Case Study ............... 163
Table 8.1: Decision Rules for applying FCI .................................................... 189
Table 8.2: Studying Effect of Transformations on CI .................................... 191
Table 8.3: Studying Effect of Transformations on FCI .................................... 192
Table 9.1  Proposed Weights for a Change .................................................... 200
Table 9.2  Computing Understandability ...................................................... 203
Table 9.3  Understandability of Modified ERDs ............................................. 206
Table 9.4  Effort Required For Fig. 9.1 ........................................................... 207
Table 9.5  Effort Required For Fig. 9.2 ........................................................... 208
Table 10.1  Proposed Approach and its Impact on Database Design .......... 218
# Contents

ACKNOWLEDGEMENTS ....................................................................................................................... IV  
ABSTRACT .............................................................................................................................................. VII  
LIST OF ABBREVIATIONS ................................................................................................................ IX  
LIST OF FIGURES .................................................................................................................................... X  
LIST OF TABLES .................................................................................................................................. XII  
CONTENTS ........................................................................................................................................ XIII  
CHAPTER 1 .......................................................................................................................................... 1  
  1.1 CONTEXT AND MOTIVATION ....................................................................................................... 3  
  1.2 TRADITIONAL DATABASE DESIGN APPROACH ......................................................................... 6  
  1.3 PROPOSED APPROACH FOR DATABASE DESIGN ....................................................................... 11  
  1.4 OUR CONTRIBUTION ................................................................................................................ 18  
  1.5 ORGANIZATION OF THE THESIS .......................................................................................... 20  
PART I .............................................................................................................................................. 25  
EXISTING DATABASE DESIGN METHODOLOGY ........................................................................... 25  
CHAPTER 2 ...................................................................................................................................... 26  
  2.1 INTRODUCTION ........................................................................................................................ 26  
  2.2 THE CLASSIC ER MODEL ......................................................................................................... 28  
  2.3 EXTENSIONS TO THE CLASSIC ER MODEL ........................................................................... 32  
  2.4 SUMMARIZING EVOLUTION OF ER MODEL ............................................................................ 52  
  2.5 MAPPING ER DIAGRAM TO RELATIONAL SCHEMA ............................................................... 55  
  2.6 NORMALIZATION THEORY .................................................................................................... 57  
  2.7 PROBLEMS IDENTIFIED ....................................................................................................... 63  
  2.8 SUMMARY .............................................................................................................................. 64  
CHAPTER 3 .................................................................................................................................... 66  
QUALITY ASPECT OF CONCEPTUAL MODELS ............................................................................... 66  
  3.1 A QUALITY FRAMEWORK .................................................................................................... 67  
  3.2 QUALITY THROUGH SCHEMA TRANSFORMATIONS ............................................................. 70
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3 ANOTHER QUALITY CRITERIA</td>
<td>75</td>
</tr>
<tr>
<td>3.4 QUALITY METRICS OF ER MODELS</td>
<td>76</td>
</tr>
<tr>
<td>3.5 STRUCTURAL COMPLEXITY OF ER MODELS</td>
<td>80</td>
</tr>
<tr>
<td>3.6 SOME OTHER APPROACHES</td>
<td>84</td>
</tr>
<tr>
<td>3.7 PROBLEMS IDENTIFIED</td>
<td>85</td>
</tr>
<tr>
<td>3.8 SUMMARY</td>
<td>87</td>
</tr>
<tr>
<td>PART II</td>
<td>88</td>
</tr>
<tr>
<td>IMPROVING CONCEPTUAL DESIGN</td>
<td>89</td>
</tr>
<tr>
<td>CHAPTER 4</td>
<td>89</td>
</tr>
<tr>
<td>IMPROVING CONCEPTUAL DESIGN WITH FUNCTIONAL DEPENDENCIES</td>
<td>94</td>
</tr>
<tr>
<td>4.1 INTRODUCTION</td>
<td>89</td>
</tr>
<tr>
<td>4.2 DEFINITIONS</td>
<td>94</td>
</tr>
<tr>
<td>4.3 SCHEMA TRANSFORMATIONS BASED ON COMPOSITE ATTRIBUTES</td>
<td>97</td>
</tr>
<tr>
<td>4.4 SCHEMA TRANSFORMATIONS BASED ON NON-KEY ATTRIBUTES</td>
<td>104</td>
</tr>
<tr>
<td>4.5 SCHEMA TRANSFORMATIONS AND NORMALIZATION</td>
<td>108</td>
</tr>
<tr>
<td>4.6 SUMMARY</td>
<td>112</td>
</tr>
<tr>
<td>CHAPTER 5</td>
<td>113</td>
</tr>
<tr>
<td>IMPACT ON NORMALIZATION</td>
<td>113</td>
</tr>
<tr>
<td>5.1 INTRODUCTION</td>
<td>113</td>
</tr>
<tr>
<td>5.2 ER-TO-RELATIONAL SCHEMA MAPPING</td>
<td>116</td>
</tr>
<tr>
<td>5.3 VIOLATION OF FIRST NORMAL FORM (1NF)</td>
<td>118</td>
</tr>
<tr>
<td>5.4 VIOLATION OF SECOND NORMAL FORM (2NF)</td>
<td>118</td>
</tr>
<tr>
<td>5.5 VIOLATION OF THIRD NORMAL FORM (3NF)</td>
<td>121</td>
</tr>
<tr>
<td>5.6 VIOLATION OF BOYCE-CODD NORMAL FORM (BCNF)</td>
<td>124</td>
</tr>
<tr>
<td>5.7 COMPOSITE KEYS, BCNF AND DEPENDENCY PRESERVATION</td>
<td>129</td>
</tr>
<tr>
<td>5.8 SUMMARY</td>
<td>135</td>
</tr>
<tr>
<td>CHAPTER 6</td>
<td>137</td>
</tr>
<tr>
<td>IMPROVING SEMANTIC REPRESENTATION</td>
<td>137</td>
</tr>
<tr>
<td>6.1 INTRODUCTION</td>
<td>137</td>
</tr>
<tr>
<td>6.2 PROBLEM WITH RELATIONSHIP ATTRIBUTES</td>
<td>138</td>
</tr>
<tr>
<td>6.3 MULTI-VALUED RELATIONSHIP ATTRIBUTES (MVRA)</td>
<td>140</td>
</tr>
<tr>
<td>6.4 SUMMARY</td>
<td>148</td>
</tr>
<tr>
<td>PART III</td>
<td>150</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Developing software systems is a complex task. Since 1950s, various efforts have been made to introduce methodologies to address problems in different dimensions which could make this task simpler. For example, a structured approach of systems development, known as Systems Development Life Cycle (SDLC), was introduced in late 1970s [Avison and Fitzgerald 2003] which became very popular and is still being practiced in one form or the other. Fig. 1.1 shows four phases of the systems development life cycle – systems analysis, systems design, systems development and systems implementation. At the same time, it was realized that a software engineering process should be taken as something different from SDLC. Due to this realization, various models [Pressman 2004] evolved over time, for example, waterfall model, prototyping model, Rapid Application Development (RAD) model, incremental model and Boehm’s spiral model. However, irrespective of which software process model is considered appropriate for development of a system, all the phases of SDLC have to be followed. In other words, despite the fact that process models differ in defining the sequence and parallelism of SDLC phases, each phase is an integral part of every process model.

Though the methodologies mentioned above have contributed significantly in successful systems development, one of the major reasons of system failures even today is a wide gap between user needs or requirements which are captured in systems analysis phase of the SDLC and the software which is ultimately delivered to the user as the output of
systems implementation phase of the SDLC. That is why, research is still being done in devising methodologies that could narrow down this gap. In the last decade, object oriented methodology [Rumbaugh et al. 1991] was introduced which became dominant not only in systems development (primarily programming) but also in systems analysis and systems design phases. Further, within analysis and design phases, conceptual modeling is a major activity which is responsible for translating user requirements to an abstract, high level model independent of any implementation details.

![Diagram of the Systems Development Life Cycle (SDLC)](image)

**Fig. 1.1: The Systems Development Life Cycle (SDLC)**

It has long been realized in the research [Krogstie et al. 1995, Moody 1998, Genero et al. 2001, Moody 2005, Siau and Tan 2005] that improving the quality of conceptual models is a major step towards improving the quality of the end product. Consequently, a very rich research material has been published which introduced various modeling techniques and methodologies from time to time. These techniques include Entity-Relationship (ER) Models [Chen 1976], Natural language Information Analysis Method (NIAM) [Verheijen and Bekkum 1882], Data Flow Diagrams (DFD) [Yourdon 1989], Object-Role Modeling
(ORM) [Halpin 1990], Unified Modeling Language (UML) [Rumbaugh et al. 1998], and State-Entity-Activity-Model (SEAM) [Bajaj and Ram 2001] to mention a few. Data flow diagrams are conceptual representation of business processes and their interactions whereas ER and ORM/NIAM models are data-oriented. UML, a modeling language for object-oriented environment, combines both process models and data models. Comparing ER model and NIAM, [Laender and Flynn 1993] observe that despite the fact the two models differ significantly in terms of their diagrammatic representation, they are very similar in their semantic representation or in their modeling capability. SEAM is an attempt to define a conceptual model which can provide a convenient interface to automate the design and management of workflows. This model represents time in addition to both data and processes. Nevertheless, all these techniques and methodologies attempt to focus on a better representation of the real-world concepts related to a problem domain.

1.1 Context and Motivation

Database design is considered to be a miniature version of systems analysis and design [Thalheim 2000]. Databases are an integral part of almost every software system. Designing a good database is the most important responsibility, which takes considerable time, of a database designer. A poor database design can result in the failure of a software system. A poor design is usually caused by an improper conceptual model or inadequate conceptual modeling technique. Halpin says that effective database applications, business rules management, data warehousing and enterprise modeling all depend on the quality of the underlying data model [Halpin 2000]. This is the reason, various improvements in the
conceptual modeling technique for databases have been presented in the literature, for instance, ER model [Chen 1976], Enhanced Entity-Relationship (EER) models [Elmasri and Navathe 2004], and Higher-Order Entity Relationship Model (HERM) [Thalheim 2000].

We observe that every technique of conceptual modeling has its own strengths and weaknesses. As a working strategy, one can think of one of the following two possibilities:

1. devise a new methodology altogether, or
2. improve an existing methodology which is already in use.

We, in this research, have adopted the latter strategy expecting that it will save considerable time of database community in learning a new methodology right from scratch and converting design of their existing applications to a new approach. For this purpose, we have selected the ER modeling technique because:

2. comparing it with other data models,
   a) generally, it is considered to be more expressive and simpler than NIAM/ORM [Laender and Flynn 1993], and
b) it has been demonstrated in the literature [Shoval and Shiran 1997] that the entity-relationship model is even superior to the object oriented model in its representation and design quality for the database conceptual design.

Hence, we intend to study the ER model and its limitations in successful representation of the real world. For the limitations observed, we intend to propose improvements in the ER modeling technique. At the same time, we want to use some measure which can evaluate the extent of improvement. In other words, comparing two models for the same mini-world problem (also known as Universe of Discourse (UoD)) – one formulated with the existing approach and the other by our proposed approach, we should be able to evaluate which model is better one and how much. This, as a matter of fact, measures the quality of a given conceptual model (that is, the ER model). The study of existing literature, which is given in chapter 3, defines some quality measures and metrics [Lindland et al. 1994, Assenova and Johannesson 1996, Moody 1998], but these measures suffer from the following problems:

- In literature, quality of a conceptual model is mainly defined in terms of lists of desirable characteristics. Mostly, these lists are unstructured, lack quantitative and objective measures, use imprecise and/or complicated definitions and usually overlap [Lindland et al. 1994, Genero et al. 2001, Moody 2005, Nelson et al. 2005],

- the measures are usually not quantitative, so the quality of any two conceptual models for the same problem cannot be compared objectively, and
The measures are in the form of an extensive list which makes the computations more complex practically. Also, the comparison of two models becomes very difficult due to non-convergence of these measures to a single numerical value representing the overall quality of a model.

The existing measures and their efficacy are discussed in detail at the end of this part (chapter 3). This particular aspect of conceptual modeling gives us the motivation to define measures which are effective and easy to apply. Without such measures, it will be just a theoretical exercise to devise a methodology for improvement of a conceptual model. Therefore, in our study, we try to identify an appropriate quality criterion and then propose measures which can be practiced easily.

In the following sections, first the traditional database design approach is described (section 1.2). Then, we present an outline of our proposed approach and the objectives set forth (section 1.3). It is then followed by a section giving a summary of our research contribution in this thesis (section 1.4). In the last section of this chapter, section 1.5, we discuss how this thesis is organized into various parts and chapters, and their contents.

### 1.2 Traditional Database Design Approach

Fig. 1.2 gives an overview of how database design methodology goes hand in hand with application design methodology. Literature [Teorey et al. 1986, Engels et al. 1992, Thalheim 2000] shows that a database design methodology is defined by three steps:
1. Conceptual design in which the database requirements, identified from min-world, are conceptualized to a high-level conceptual schema. This schema represents semantics of the real world. It is considered to be a high-level schema because it is independent of any specific database management system (DBMS).
2. Logical design in which the conceptual schema is transformed to a logical schema which is based upon a specific DBMS model, for instance, a hierarchical model or a relational model.

3. Physical design in which the logical design of the preceding step is transformed into physical layout of files and indexes for the specific DBMS model.

The significance of step 1, that is the conceptual design, is that a database designed without this step or designed based on an incorrect conceptual model is often unreliable, contains redundant and/or inaccurate data, performs poorly, be inflexible, or be characterized by any number of other standard anomalies [Thalheim 2000]. Thalheim classifies common mistakes in the database design amongst thirteen categories [Thalheim 2000] where important ones are listed below:

- **The Spreadsheet Design:** Frustrated by too much documentation, a database designer puts everything in one big table.

- **Mega Attributes:** Attributes should contain the smallest units of meaningful information, rather than combining different attributes together.

- **Missing Keys:** Refers to tables without candidate/primary keys.

- **Bad Keys:** Keys are selected which are not unique.

- **Ivory Tower Syndrome:** The database should be designed according to customer’s needs and not according to the designer’s experience.

- **No Associations:** For related entities in the real world, data in different tables should be related together.

- **Wrong/Too Many Associations:** Refers to associations amongst wrong tables.
➢ **Too Much Redundancy:** A well designed database should not have any redundant information other than required by associations.

➢ **Many Null Values:** A well-designed database should have minimum null values.

Assuming the underlying DBMS model is relational, the stage of logical design, that is step 2 in Fig. 1.2, can further be broken into two processes: Mapping [Chen 1976, Engels et al. 1992, Elmasri and Navathe 2004] and Normalization [Codd 1972, Codd 1974, Ullman 1988, Date and Fagin 1992, Date 2003], as shown in Fig. 1.3. The process of mapping basically transforms an ER model to a relational schema. Normalization can be considered as improving the logical model in order to eliminate/minimize data anomalies and redundancies. This process requires a set of functional dependencies [Codd 1970] (and other dependencies, depending upon what normal form the relational schema is required to be normalized) as its input. These functional dependencies are identified and defined from the mini-world. This is to be noted that the formal concept of a functional dependency is defined only at the logical level for a relational schema. However, though informally, the concept helps in the conceptual design stage as well for identification of key attributes as indicated in Fig. 1.3.

Step 3 of the database design approach, namely physical design, is not related to our work. Hence, for our purpose, the three steps or stages of the traditional approach are taken as:
Fig. 1.3: Traditional Database Design Approach
1. Conceptual Design,  
2. Mapping, and  
3. Normalization

As stated in section 1.1, for conceptual modeling, we have selected the ER model due to a number of reasons. Therefore, in this thesis, we shall be using the terms conceptual model, ER model, ER schema and conceptual schema interchangeably.

In the next section, we present our proposed approach and the objectives set forth for research in this regard.

### 1.3 Proposed Approach for Database Design

Although every step defined in the 3-step database design methodology (ignoring the physical design, as stated earlier) has its own significance, the most demanding and challenging step is the conceptual design because later steps are merely transformations. Due to its effectiveness in successful systems development, conceptual design has also been applied to the more general area of requirements analysis for information systems, and is now a feature of several methods and CASE tools [Rochfeld 1987, Laender and Flynn 1993, Thalheim and Yigitbasi 1996, Detienne and Hainaut 2001]. Further, the challenge within conceptual design is to develop a conceptual schema which closely represents the real world. As we show in the next chapters of this research, if a conceptual schema fails to do so, then either the resulting logical schema are incorrect
and/or the maintainability of the system suffers. It gives us the motivation to lay down the following objectives:

1. A conceptual schema should closely represent the mini world, and

2. if it is not so, then schema transformations should be defined and applied to improve the model.

In order to meet these objectives, we propose a methodology which is shown in Fig. 1.4. The proposed methodology suggests that a new stage in conceptual design should be introduced after the stage of conceptual modeling in order to improve the schema systematically. This, in turn, requires devising rules to be used in this new stage which can describe how to improve a given conceptual schema. Though the complete proposed methodology is explained in subsequent chapters, one can see here a promising difference in the approaches of Fig. 1.3 and Fig 1.4, as functional dependencies are directly used in Fig. 1.4 for the conceptual design stage.

Before we proceed further, it is worth emphasizing the need of improving a conceptual model as it provides the basic motivation of the whole work presented in this thesis. One can argue that as long as improvement can be done at any stage of the database design approach, it is immaterial where the improvement is done, or in other words, it is just the quality of the end product which matters.
Fig. 1.4: Proposed Database Design Approach
Firstly, this argument negates the importance and usefulness of the conceptual modeling stage. Had it been just the quality of the end product that matters, there was no need of developing a conceptual model in the first place (not to mention a continual effort for about two decades introducing many new constructs to improve the ER model or introducing other models). One could start with a universal relation and normalize it to get the end product. As a matter of fact, practitioners did try this approach historically but had to abandon it later. It was soon realized, as stated earlier, that understanding user requirements at a conceptual level is a pre-requisite of developing a good database design [Wiederhold and Elmasri 1979]. This realization led to an ongoing research in this particular area being presented in a series of international conferences (since 1979) exclusively on the ER approach [ER, APCCM].

Secondly, according to the fundamental software engineering principles:

- quality of a software system is not measured just by the end product; rather the whole process of software development determines the quality [Kan 2003, Pressman 2004, Moody 2005].
- the relative costs to find and repair an error increase dramatically as we go from requirements phase (1 unit of cost) to design (3-6 times) to field operation (40-1000 times). Hence, the sooner a design change is made, the less impact it has on cost and maintenance effort [Boehm 1981, Genero et al. 2001, Pressman 2004, Moody 2005]. A clearer conceptual model has higher probability of getting its errors corrected at an earlier stage than that of a normalized schema otherwise
which usually loses the real-world correspondence. This very reason also improves the maintainability of the systems developed from better conceptual models as changes can be easily accommodated in them.

Thirdly, the following two precepts from well-known researchers are worth quoting:

- In their classic paper on understanding quality in conceptual modeling, [Lindland et al. 1994] note that “Most developers are recognizing that the quality of the end product depends greatly on the accuracy of the requirements specification and are concentrating more on improving the early stages of development.”

- In his recent paper [Moody 2005], Moody says: “The traditional focus of software quality has been on evaluating the final product. However empirical studies show that more than half the errors which occur during systems development are requirements errors. Requirements errors are also the most common cause of failure in systems development projects. … … Improving quality of conceptual models is also likely to improve the quality of delivered systems […]. While a good conceptual model can be poorly implemented and a poor conceptual model can be improved at later stages, all things being equal, a higher quality conceptual model will lead to a higher quality information system. The conceptual model quality may affect both the efficiency (time, cost, effort) and effectiveness (quality of results) of IS development. For example, a poor quality conceptual model may increase development effort (as a consequence of detecting and correcting defects) …”
It should be noted that, in practice, a conceptual model always goes through a process of improvement, but that is done in an unstructured and subjective manner. In the absence of well-defined structured rules, the improvement process is based on sheer experience of the database designer. Major part of our research focuses on defining rules which are structured and well-defined so that subjectivity in the conceptual modeling can be minimized.

We understand that the task of improving conceptual models has a vast scope. We, in our thesis, are therefore addressing its following dimensions only:

1. **ER Schema Improvement**: Along this dimension, we need to answer questions like:
   - How can we know that an ER schema for a given problem domain needs improvement?
   - How that improvement can be made?
   - What should be the criteria to evaluate the effectiveness of an improvement?

2. **Semantic Representation**: Along this dimension, we need to answer questions like:
   - What are the real-world concepts and/or constraints which the existing ER models are unable to represent at all or to represent clearly?
   - How to represent such concepts and/or constraints appropriately?
3. **Quality Perspective:** Along this dimension, we need to answer questions like:

- Can we understand the notion of the quality of a conceptual model?
- Can the quality aspect of an ER model be addressed quantitatively?
- What should be the metrics and/or methodology to compute the quality of a conceptual model?

For the first dimension, we formally define the concepts of a functional dependency and the associated terms which can be used at the conceptual design stage. Based on these concepts, we devise rules which help us understand when a given schema needs improvement. Further, the proposed rules transform a conceptual schema to an improved schema, as well, therefore meeting the objectives of this research.

For the second dimension, we identify problems of semantic representation in the existing ER model and then propose solutions for these problems. We elaborate our proposed solutions with real world examples and demonstrate their effectiveness on semantic representation and on later stages of the database design methodology.

Along the third dimension, we identify the need of defining appropriate metrics to evaluate quality of a conceptual schema. We thus propose quantitative and easy-to-apply metrics which can compute quality of a given ER schema. After defining these metrics, we verify the effectiveness of the proposed metrics for the rules defined along first two dimensions described above. Using fuzzy based approach, we further refine our proposed
metrics to take into account membership values between 0 and 1 as well instead of just these two discrete values.

Recently, researchers [Piattini et al. 2001, Genero et al. 2002] have identified structural complexity as an important characteristic to evaluate quality of an ER model. Therefore, we also study this particular aspect of the quality and introduce metrics to measure structural complexity of the conceptual model.

1.4 Our Contribution

Our research on the topic of improving conceptual models in database design has led us to formulation of an improved methodology. Part I of the thesis, comprising of chapter 2 and chapter 3, is basically a survey of the existing literature in the related area. The rest of the parts, Part II and Part III, comprising of six chapters (Chapters 4 – 9), all are based on our contribution to this thesis which is categorized and summarized below. To the best of our knowledge, this work has not been done before.

1. We have studied in depth the existing database design methodology and have proposed addition of a new step in the conceptual design stage. Our proposed methodology makes conceptual modeling an iterative process where each iteration, called schema transformation, semantically improves the conceptual model gradually.

2. We have given formal definitions for new concepts which are necessarily required at conceptual modeling stage in our proposed methodology. These include functional dependency, key of an entity type, a key attribute, a non-key attribute. It is to be noted
that these terms already exist but for a relational model. Our contribution is defining these terms at the conceptual level without being committed to any lower level database model achieving abstraction at higher level.

3. We have provided theoretical background necessary to identify that a given conceptual schema needs improvement. We have also proposed a set of schema transformation rules and have demonstrated that consequently a conceptual schema gets improved semantically.

4. We have translated the mapping algorithm given in [Chen 1976, Engels et al. 1992, Elmasri and Navathe 2004], which is used in the transformation of a conceptual schema to a relational schema, into formal terms. This formal algorithm is then used to transform the proposed constructs of ER model to neatly extend the existing mapping algorithm.

5. We have studied the impact of applying our proposed schema transformation rules on normalization of the resulting relational schema. We have proved that the resulting relations satisfy Boyce-Codd Normal Form (BCNF) [Codd 1974]. In order to understand BCNF transformation thoroughly, we have formally proved some interesting properties which had informal definitions only in the literature.

6. We have proposed concepts for some missing constructs in the existing EER model. These are defined as single-valued relationship attributes and multi-valued relationship attributes. We have identified various situations which demonstrate that in the absence of these concepts, a conceptual schema does not correctly represent the real world and also results in an un-normalized relational schema. We have also
provided mapping rules for these proposed constructs. The proposed mapping rules lead us to modify the mapping algorithm.

7. In order to study the quality aspects of a conceptual model, we have gone through the existing literature for this purpose. We have identified that there is a lack of easy-to-apply quality metrics in this area. We have thus proposed new quality metrics and have demonstrated their effectiveness using practical scenarios.

8. We have applied fuzzy logic concepts to further refine our quality metrics. We have introduced a graph notation and algorithms to map a given ERD and a set of functional dependencies to a graph which can then be used to compute the proposed fuzzified quality metric.

1.5 Organization of the Thesis

The rest of the thesis is organized into four parts. The first part introduces existing database design methodology, its terms and concepts. We also summarize in this part the research done in this area so far. The second part describes thoroughly our proposed methodology to improve the conceptual model. The third part is devoted to our study and research on quality aspects of the conceptual model. Finally, the fourth part presents conclusions by giving a summary of results that we arrived at during our research. It also discusses some ideas for future work. These parts of the thesis are structured as follows:
Part I: Existing Database Design Methodology

This part is primarily aimed at study of the existing literature and identification of the associated problems in the area of database design. This part is divided into two chapters which are described below.

Chapter 2 discusses main steps of the database design methodology – conceptual models, the extensions suggested in the literature, mapping rules to transform a conceptual schema to a logical schema, and the process of normalization. Concepts discussed for these steps are frequently used in our thesis. In this chapter, we specifically review terms and concepts of the entity-relationship model and the evolution of ER model over a period of time. In normalization theory, we cover concepts of functional dependency, and normal forms.

Chapter 3 looks into the quality aspects of a conceptual model. It studies quality measures and metrics proposed in the literature for a conceptual or ER model. This study provides us an insight about the conceptual modeling stage with a quality perspective. Only with appropriate measures, we are in a position to evaluate the quality of a conceptual model or compare the quality of two given models. This leads us to validate any proposed metrics and the proposed improvements suggested in the conceptual modeling process.
Part II: Improving Conceptual Design

This part comprises of chapters 4 – 6. It describes in detail our proposed methodology to improve a given conceptual model and its impact on the relational schema with respect to the normalization theory.

In chapter 4, we define concepts of functional dependency and key attributes for a conceptual model and then present various scenarios which provide the basis for formulation of our proposed rules to improve the conceptual model. These rules are also stated in the same chapter.

In chapter 5, we investigate characteristics of a relation schema which comes to a lower normal form after its transformation from a conceptual schema. We identify problems with the conceptual model due to which it results in a relation schema of lower normal form. We then propose rules which should be considered and applied while developing the conceptual schema in order to obtain a relation schema which is normalized up to BCNF. We also relate these rules with those developed in chapter 2.

In chapter 5, we also state and prove some important properties of a relation schema which is in 3NF but not in BCNF. These properties help us understand that no improvement in conceptual model can be done whereby the schema generated satisfies BCNF without loss of some functional dependencies. It is to be noted that the existing approach of normalization is also unable to transform a relational schema from 3NF to BCNF without loss of some functional dependencies. So, in this respect, our proposed
approach is no worse than existing methodology. In this chapter, we present a formal proof for this observation given in the form of a theorem. It should also be noted that our research shows stronger results than the informal observation which states that the conceptual schema cannot be improved in such a way to satisfy BCNF.

Chapter 6 elaborates semantic ambiguities arising from relationship attributes on an ER model. In the presence of these ambiguities, a conceptual model may not represent the mini-world correctly. In this chapter, we demonstrate that it may also lead to a relation schema which is in lower normal form. Concepts of single-valued and multi-valued relationship attributes are therefore proposed to resolve the semantic ambiguities. For these attributes, proper notations and rules for mapping to relational schema are also proposed.

**Part III: Quality of a Conceptual Model**

This part comprises of chapters 7 – 9. In Part II, rules have been defined to improve a conceptual model and their application has also been studied. This part of the thesis focuses on defining the quality of a conceptual model and devising appropriate metrics for its measurement. Chapter 7 discusses limitations of existing quality metrics and then proposes easy-to-apply quantitative metrics to understand quality aspects of a conceptual model. This chapter establishes the need of these metrics and introduces their basic concept; however, these metrics need further refinement. This refinement is required because a quality measure should not take on discrete values (which is the case with measures given in chapter 7 due to simplicity), and so a continuous scalability in quality
measures should be provided. Chapter 8 presents the proposed refinement using fuzzy logic based approach. The last chapter of this part, chapter 9, introduces another dimension of quality of a conceptual model called structural complexity. We, in this chapter, propose metrics for this measure, too. A hypothetical case study is presented where these metrics are applied to compute quality and to verify that our proposed rules for schema transformations actually improve quality with these metrics.

**Part IV: Conclusions**

This part consists of a single chapter, chapter 10. In this chapter, we summarize our research work and outline main findings. We also specify directions for future research.
Part I

EXISTING DATABASE DESIGN

METHODOLOGY
Chapter 2

The Database Design Process As Today

2.1 Introduction

As discussed in chapter 1, the database design approach primarily consists of three stages: 1) conceptual design in which database requirements are represented by a conceptual schema, ER model in our case, 2) mapping to logical model in which the ER model from previous stage is transformed to a specific database model, relational model in our case, and 3) normalization in which the relational schema resulting from the second stage is improved by eliminating data anomalies. Therefore, in this chapter, we review the existing literature related to all these three stages, one by one. The first three sections including this one (sections 2.1–2.3) present a review on conceptual models in database domain, starting from the classic ER model to various extensions proposed by researchers. We also discuss problems associated with these models so that we have a better understanding of the evolution of the ER model. Section 2.4 is a review of the existing mapping rules and section 2.5 provides a summary of normalization theory as practiced today.

The ER model has been widely used in the database community for its ease of use, ease of representation and ease of understanding [Engels et al. 1992]. Thalheim notes that, using this simple graphical representation technique, even complex database schemes can be understood [Thalheim 2000]. He further adds that this model has been so successful
that it is used at present in many branches of computer science, even in software engineering. Therefore, the ER model and its associated entity-relationship diagram (ERD) are focus of our research for suggesting improvements in the conceptual model. The quality of an ER model can be determined by how close and how well it represents the real world (that is, the problem domain). Entity relationship modeling is however subjective in representing real world information. Sometimes it is not easy especially for a novice to decide whether a concept be modeled as an entity, an attribute or a relationship. Efforts have been made continually to reduce the gap between representation of an ER model and the real world situation. In this chapter, we discuss that the existing ER model has actually evolved over time through an ongoing process of gradual improvements and extensions to the basic model initially proposed by [Chen 1976]. The evolution of the ER model was primarily caused by two different reasons:

- Either, the prevailing ER model was unable to represent a real world concept existing at that time, or
- the growing complexity of the applications demanded introducing new constructs or concepts in the ER model.

In any case, these reasons refer to semantic representation problems of an ER model. To fill in the semantic gap, various researchers adopted the following approaches to address these issues:

1. Introduce new constructs in the then prevailing ER model,
2. Extend the prevailing ER model by amalgamating new concepts and constructs to define an extended ER (EER) model, or
3. Devise new models altogether which are primarily different from the ER model.

In the following sections, starting with the Chen’s model, we discuss the need of improvements and extensions leading to introduction of new constructs and the extended models.

2.2 The Classic ER Model

Chen notes that the early 70s era had competing forces in the database area [Chen 2002]. He categorizes these into competing forces in the industry and competing forces in the academic world. Most of the organizations in the industry at that time were using file systems and a few were using database systems. Amongst the data models that had been implemented as commercial products were the hierarchical model (such as IBM’s IMS) and the network model (such as Honeywell’s IDS). In the academic world, the relational model had been proposed, and it had generated considerable interest in the database community. Most people in the industry and in academia were working on relational database management system (RDBMS) prototypes whereas definitions and algorithms for normal forms of relations were being investigated by the academic world. Since the stage of conceptual design was not introduced by that time, the database design approach was to develop a logical model directly from database requirements. Chen, at that time, proposed his ER model which was presented as a unification of different views of data – the network model, the relational model and the entity set model [Chen 1976]. He also introduced a proper diagrammatic technique to represent the ER model which he called the ER diagram.
The concepts of entity, entity set, relationship, relationship set, value and value set, attribute, entity key, relationship mappings, and relationship arity as defined by [Chen 1976] are as follows:

- **Entity**: A “thing” which can be distinctly identified. For example, a specific person, company or an event are entities.

- **Entity Set**: Entities are categorized into entity sets based upon common properties or a common predicate which tests whether an entity belongs to a specific entity set or not. For example, EMPLOYEE, PROJECT and DEPARTMENT are entity sets.

- **Relationship**: An association among entities. For example, “father-son” is a relationship between two entities.

- **Relationship Set**: A relationship set, \( R_n \), is a mathematical relation among \( n \) entities, each taken from an entity set:

\[
\{ [e_1, e_2, ..., e_n] \mid e_1 \in E_1, e_2 \in E_2, ..., e_n \in E_n \},
\]

and each tuple of entities, \([e_1, e_2, ..., e_n] \), is a relationship.

- **Value and Value Set**: The information about an entity or a relationship is obtained by measurement or observation and is expressed by a set of attribute-value pairs. “red”, “Peter”, and “Johnson” are values. Values are classified into different value sets, such as COLOR, FIRST-NAME and LAST-NAME.

- **Attribute**: A function which maps from an entity set or a relationship set into a value set or a Cartesian product of value sets:
\[ f: E_i \text{ or } R_i \rightarrow V_1 \text{ or } V_{i1} \times V_{i2} \times \ldots \times V_{in}. \]

- **Entity Key:** A group of attributes to identify entities in an entity set.
- **Relationship Mappings:** One instances of an entity set \( E_1 \) may have one or \( n \) \((n > 1)\) instances of entity set \( E_2 \) defined in a relationship set. This categorizes relationship mappings into 1:1, 1:n or m:n.
- **Relationship Arity:** A relationship set may be defined on one or more entity sets. If it is defined on two entity sets, then it is a binary relationship set; on three entity sets, it is a ternary relationship set, and so on.

The notations proposed by Chen for developing an ER diagram are summarized in Table 2.1 and are demonstrated in Fig. 2.1.

Chen’s ER model, due to its ease of use and semantic representation, laid the foundation for developing higher level conceptual models prior to development of the logical model. However, Chen’s model was just a beginning. There were certain concepts which were not defined clearly in his paper. For example, though he discusses weak entity relations and regular entity relations, the term relation is in the context of the relational model, that is, weak entity relations and regular entity relations are relations of the relational model. This required a clear definition for weak and regular entity sets within the entity relationship model without any pre-commitment to the relational model. Moreover, it was later observed that the weakness is a property of a relationship and not that of an entity [Scheuermann et al. 1979]; therefore, leading to the concept of weak relationships as discussed in section 2.3.4. At the same time, Chen’s ER model was devoid of some
concepts required for more complex requirements. For instance, it did not define composite attributes (discussed in section 2.3.1), multi-valued attributes (discussed in section 2.3.2), relationship attributes (discussed in section 2.3.5), and hierarchies (as discussed in section 2.3.3).

Table 2.1 Summary of ER Notations

<table>
<thead>
<tr>
<th>Concept</th>
<th>Symbol</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity Set</td>
<td>Rectangle</td>
<td>EMPLOYEE</td>
</tr>
<tr>
<td>Relationship Set</td>
<td>Diamond</td>
<td>DEPT HAS EMP</td>
</tr>
<tr>
<td>Value Set</td>
<td>Circle</td>
<td>EMP E# emp#</td>
</tr>
<tr>
<td>Attribute</td>
<td>Mapping</td>
<td>E# is an attribute in the above example</td>
</tr>
<tr>
<td>Weak Entity Set</td>
<td>Double-lined Rectangle</td>
<td>DEPEND HAS EMP</td>
</tr>
</tbody>
</table>
2.3 Extensions to the Classic ER Model

In this section, we discuss the following enhancements and extensions to the original ER model as proposed from time to time:

- The concept of composite attributes
- The concept of multi-valued attributes
- The concept of generalization
- Incorporating relationship constraints
Extended Entity Relationship (EER) model

Other enhancements
  - Engels’ EER model
  - Higher-order ER model
  - Incorporating Business Requirements and Constraints (BRC)

2.3.1 Composite Attributes

In Chen’s ER model, an attribute was a function which maps from an entity set into a value set. Value sets could be called COLOR and NAME, for instance. It was later observed that an attribute may need to be further broken down into simpler attributes which represent independent meanings. For example, in a certain problem domain, an attribute NAME need to be decomposed into FIRST_NAME and LAST_NAME. This decomposition may be required because some queries refer to FIRST_NAME only whereas some refer just to LAST_NAME. In Chen’s model, there was no such notion as a decomposition of an attribute. Elmasri et al. introduced the concept of composite attributes [Elmasri et al. 1985] to solve this problem. They called NAME a composite attribute whereas FIRST_NAME and LAST_NAME were called simple (or atomic) attributes. This distinction was important in many respects. First, it provided a clearer semantics to the conceptual model. Second, it facilitated queries dig out basic information from the database without any internal processing. Third, the violation of first normal form (1NF) was avoided when an ER model containing composite attributes was transformed to a relational schema.
The diagrammatic notation for this concept was to branch out more ovals (simple attributes) from a parent oval (the composite attribute) as represented in Fig. 2.2.

![Diagram](image)

**Fig. 2.2: A Composite Attribute (Name)**

### 2.3.2 Multi-valued Attributes

Referring again to the Chen’s definition of an attribute, an attribute is a function and so it can map an entity (or an attribute of an entity) to a single value only. This concept was not clear enough to model real world requirements in which an attribute need to be mapped to more than one value. For instance, an employee may have two or more phones which are to be recorded in the database. Elmasri et al. [Elmasri et al. 1985] observed this distinction between an attribute which can always take on a single value for every entity belonging to an entity set and an attribute which may have more than one value for some entity or entities belonging to an entity set. They called attributes of former type as *single-valued* attributes and of the latter type as *multi-valued* attributes. The diagrammatic notation for a multi-valued attribute is a double-lined oval as represented in Fig. 2.3. Here, PHONE is a multi-valued attribute of EMPLOYEE. On the other hand, no employee has more than one name (supposedly) that is to be kept track of in the database,
so NAME is a single-valued attribute. This distinction was important again for the same three reasons stated in the previous sub-section.

Fig. 2.3: A Multi-valued Attribute (Phone)

2.3.3 Generalization (Hierarchy)

Referring to the database of employees designed in Fig. 2.1, say, the EMPLOYEE entities have attributes ID, NAME, ADDRESS and SALARY. Further, if an employee is an engineer, then the system wants to keep track of some additional information about its engineering discipline, the year of graduation and the respective college. Likewise, if an employee is a worker, we want to know his/her experience (number of years) and his/her skill. In classic approach, the additional attributes GRAD_YEAR, COLLEGE, EXP and SKILL could be defined for the EMPLOYEE entity type, in addition to the attributes already specified, as shown in Fig. 2.4. However, this solution introduces many null values in the database due to the following reasons:

1. If an employee is an engineer, EXP and SKILL are assigned null values for this entity,

2. If an employee is a worker, GRAD_YEAR and COLLEGE are assigned null, and
3. If an employee is neither an engineer nor a worker, GRAD_YEAR, COLLEGE, EXP and SKILL all would be null for this particular entity.

Another problem with this solution is of abstraction. Under the given scenario, employees are classified into engineers, workers and others in the real world (that is, with respect to this problem domain) whereas the ER diagram is unable to represent this fact. Third problem with this solution is that exclusive relationships of engineers and/or workers with some other entity-sets cannot be specified. For instance, say, we have an additional constraint in the problem domain stating that only an engineer can be a manager of the project. Very clearly, Fig. 2.4 is unable to represent this exclusive relationship.

![Fig. 2.4: An Entity Type Designed without Hierarchy](image)

[Smith and Smith 1977] presented a more elegant solution to this problem. They suggested that ENGINEER and WORKER should be two different entity types created from a generalized EMPLOYEE entity type. They called this concept “Generalization” or a “Hierarchy” [Smith and Smith 1977]. Using this concept, the common attributes are defined for EMPLOYEE whereas the exclusive attributes of ENGINEER and WORKER are defined with respective “sub-entity-types”, as shown in Fig. 2.5. According to [Smith...
and Smith 1977], generalization is perhaps the most important mechanism we have for conceptualizing the real world and the main advantage of this approach is that the model becomes a more precise representation of reality. This makes the model more understandable and thus less prone to erroneous access and manipulation. Their research further indicates that the real world constraints on the database can be implemented more effectively to avoid data integrity problems.

![Fig. 2.5: An Entity Type Designed with Hierarchy](image)

Hammer and McLeod further refined the concept of generalization in their Semantic Database Model (SDM) [Hammer and McLeod 1981] which introduces class/subclass relationships and associated constraints through the concepts of classes, types of classes, interclass connections, class membership predicates, and attribute inheritance. Although Hammer and McLeod didn’t introduce any diagrammatic notations, these concepts improved the semantics represented in the ER model considerably. Important features of SDM, which are related to our research, are listed below:
The concept of a class is similar to an entity-set. In other words, an SDM database is a collection of entities that are organized into classes.

The class has a collection of members. The concept of a member of a class is analogous to the concept of an entity of an entity-set.

The class is either a base class or a non-base class. A base class is one that is defined independently of all other classes in the database; whereas, a non-base class does not have independent existence. Referring to our scenario described above, EMPLOYEE is a base class whereas ENGINEER and WORKER are non-base classes.

If the class is a base class, it must have a set of attributes which uniquely identifies the members of that class. SDM calls this set an identifier. This is the same concept as that of entity-key in the classic ER model.

A non-base class has an associated interclass connection which defines subclass/parent class relationship. A subclass S of a class C (called the parent class) is a class that contains some, but not necessarily all, of the members of C. By this definition, ENGINEER and WORKER are subclasses and EMPLOYEE is a parent class. It is worth mentioning that the same entity can be a member of many classes, for example, a given entity can be simultaneously member of ENGINEER and SUPERVISOR where SUPERVISOR is another subclass of EMPLOYEE. This concept of “subtype” was missing in previous database models [Hammer and McLeod 1981].

In SDM, a subclass S is defined by specifying a class C and a predicate P on the members of C, that is, S consists of just those members of C that satisfy P. If P is
defined on the member attributes of C, then the subclass defined this way is called *attribute-defined subclass*. If a subclass is not predicate defined, then its members are “manually” defined by users. Such a subclass is called *user-controllable subclass*.

- A subclass can be defined which has two or more parent classes, that is, a member of subclass is always a member of two or more parent classes. Such a subclass is called an *intersection subclass*.
- In addition to the intersection capability, a subclass can be defined by class union and difference capability. A *union subclass* contains those members which are present in either C₁ or C₂, being the two subclasses from C. A *difference subclass* contains those members of C that are not in C₁.

The intersection, union and difference subclass definition primitives are provided because they often represent the most natural means of defining a subclass.

### 2.3.4 Relationship Constraints

Although the classic ER model introduced relationship constraints in the form of relationship mappings, there was no way to specify constraints like “every employee has to work on some project (or every employee may not work on some project)”. Scheuermann et al. identified these constraints as *total* and *partial* relationships [Scheuermann et al. 1979]. They defined such constraints formally as given below:

> “An arbitrary relationship set, Rᵦ, is a relation among n entities, each taken from
> an entity set, i.e.
\[ R_i = \{[e_1, e_2, \ldots, e_n] \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n \}, \text{ where} \]

each tuple \([e_1, e_2, \ldots, e_n]\) constitutes a relationship.

A relationship \(R_i\) is total on the entity set \(E_k\) if:

\[ \forall e_k \in E_k \Rightarrow \exists [e_1, e_2, \ldots, e_n] \in R_i \]

Informally, this means that all instances of the entity set \(E_k\) occur in some tuple of \(R_i\).

The notation proposed for total relationship was a dot on the edge of the relationship connected with the total entity set. This notation is shown in Fig. 2.6. It is to be noted that a relationship constraint which is not total on any entity-set was called partial.

![Fig. 2.6: Total and Partial Relationship Constraints](image)

Scheuermann et al. [Scheuermann et al. 1979] also introduced weak relationships in the ER model. A weak relationship was defined as a relationship in which the existence of an entity in one entity set (the weak one) depends upon the existence of a specific entity in the other entity set (the strong one). It is to be noted that Chen’s definition [Chen 1976] was for a weak entity whereas Scheuermann et al.’s definition was for a weak relationship. Scheuermann et al. thus observe that weakness is a property of a relationship, and not that of an entity. The notation for a weak relationship ("Has") is a double-lined diamond shape as given in Fig. 2.7.
Elmasri et al. proposed EER model [Elmasri et al. 1985] which further built upon the above approach of extending the classic ER model and introduced these concepts more formally. Their work and the diagrammatic notations are summarized in the following sub-section.

### 2.3.5 Extended Entity Relationship (EER) Model

The EER model by Elmasri et al. [Elmasri et al. 1985, Elmasri and Navathe 2004] is an extension to the classical ER model which incorporates the modifications suggested by that time. This EER model has defined clearer concepts in more formal terms as these definitions do not mix in the RDBMS terminology. Key concepts of this model are defined below:

- **Entity Type:** Same as entity-set in the classic ER model. Represented by a rectangle.

- **Entity:** Same concept as in the classic ER model. Not represented in the EER model.

- **Attribute:** A characteristic or property about entities belonging to an entity type which describes some information about the respective entity. For example,
EmployeeID, DepartmentName. It is represented by an oval connected with the rectangle.

- **Key Attribute:** Same as entity-key in the classic ER model. Represented by underlining the attribute. Every entity type on EER diagram must have at least one key attribute. Later, we define this term formally.

- **Composite Attribute:** Same concept as given in sub-section 2.3.1. Represented by branching out ovals from an oval. For example, Address can be further divided into Apartment#, street and city. If an attribute is not a complex attribute, then it is a simple attribute.

- **Multi-valued attribute:** Same concept as given in sub-section 2.3.1. Represented by an oval in double-lined border.

- **Weak Entity Type:** An entity type that does not have a key attribute of its own is called a weak entity type. Entities belonging to a weak entity type are identified by being related to specific entities from another entity type with respect to one of their attributes. Such other entity type is known as owner entity type and the set of attributes of weak entity type which participates in unique identification of weak entities is known as partial key. A weak entity type is shown as a rectangle on an ER diagram but with double border.

- **Relationship Type:** An association amongst a set of entity types. It is denoted by a diamond shape on ERD. For example, EMPLOYEE WORKS-IN DEPARTMENT, the two entity types are associated in the relationship type WORKS-IN.
Identifying Relationship Type: A relationship type that relates a weak entity type to its owner entity type is known as identifying relationship type. It is represented as a diamond shape on an ER diagram but with double border.

Cardinality of Relationship Type: It defines a relationship constraint which specifies how many relationship instances an entity can participate. Usually, the cardinality is categorized as one-to-one (1:1), one-to-many (1:M) or many-to-many (M:N).

Participation Constraints and Existence Dependency: Participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type. This constraint is categorized into total participation and partial participation. If every entity in an entity type E₁ must be related to some entity (or entities) in another entity type E₂ through a relationship type R, then the participation of E₁ in R is total (or E₁ has an existence dependency on E₂); otherwise it is partial. Total participation is represented by a double line connecting the respective entity type to the relationship type, as shown in Fig. 2.8. A single line (as usually drawn) represents partial participation.

Relationship Attributes: Some attributes can be specified on the relationship types. For example, to record the number of hours an employee works on a particular project, an attribute HOURS can neither be placed on EMPLOYEE entity type nor on PROJECT entity type because none of these entity types can define a value of HOURS attribute for any of their entities without the relationship type PRJ-WORK (Fig. 2.6) between these two entity types. Hence,
HOURS can be defined only on PRJ-WORK, as shown in Fig. 2.8. (Represented the same way by an oval as a regular attribute.)

![Diagram of Employee, Project, and PRJ-WORK relationships with Hours](image)

**Fig. 2.8: Participation Constraints and Relationship Attribute**

- **Specialization:** Specialization (or generalization) is the process of defining a set of subclasses of an entity type. The entity type itself is called a superclass of the specialization. Representation is given in Fig. 2.9.

- **Predicate-defined Subclass:** Same concept as attribute-defined subclass [Smith and Smith 1977] given in sub-section 2.3.3. If a subclass is not predicate defined, then it is called user-defined (same concept as user-controllable subclass [Smith and Smith 1977], as discussed in sub-section 2.3.3).

- **Disjointness Constraint:** If an entity can be a member of at most one subclass of the specialization, then the specialization is called disjoint specialization; otherwise, it is overlapping. A disjoint specialization is represented by putting a small ‘d’ in the circle connecting superclass and its subclasses; whereas, an ‘o’ in the circle indicates overlapping specialization, as illustrated in Fig. 2.9.
Fig. 2.9: A Specialization with Disjoint Constraint

- **Completeness Constraint**: This constraint on a specialization is categorized into total specialization and partial specialization. A total specialization constraint specifies that every entity in the superclass must be a member of at least one of the subclasses in a specialization. Represented by a double line connecting superclass to the circle (as shown in Table 2.2). If a specialization is not total, then it is partial and is represented by the usual single line connecting the superclass to the circle, as illustrated in Fig. 2.9.

- **Shared Subclass**: Same as intersection subclass discussed in section 2.3.3. Represented as given in Fig. 2.10.

- **Category**: Same as union subclass discussed in section 2.3.3. Represented as given in Fig. 2.11.
Fig. 2.10: A Shared Subclass

Fig. 2.11: A Category
Table 2.2  Summary of EER Notations

<table>
<thead>
<tr>
<th>Concept</th>
<th>Symbol</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entity Type</strong></td>
<td>Rectangle</td>
<td><img src="emp.png" alt="Employee" /></td>
</tr>
<tr>
<td><strong>Relationship Type</strong></td>
<td>Diamond</td>
<td><img src="dept_emp.png" alt="Department to Employee Relationship" /></td>
</tr>
<tr>
<td><strong>Attribute</strong></td>
<td>Oval</td>
<td><img src="emp.png" alt="Employee Attribute" /></td>
</tr>
<tr>
<td><strong>Key Attribute</strong></td>
<td>Attribute underlined</td>
<td><img src="emp.png" alt="Employee Key Attribute" /></td>
</tr>
<tr>
<td><strong>Composite Attribute</strong></td>
<td>Oval branched-out</td>
<td><img src="emp.png" alt="Employee Composite Attribute" /></td>
</tr>
<tr>
<td><strong>Multi-valued Attribute</strong></td>
<td>Double-lined oval</td>
<td><img src="emp.png" alt="Employee Multi-valued Attribute" /></td>
</tr>
<tr>
<td><strong>Weak Entity Type</strong></td>
<td>Double-lined Rectangle</td>
<td><img src="section.png" alt="Section Entity Type" /></td>
</tr>
<tr>
<td><strong>Identifying Relationship Type</strong></td>
<td>Double-lined diamond</td>
<td><img src="course_section.png" alt="Course to Section Relationship" /></td>
</tr>
<tr>
<td><strong>Cardinality Constraints</strong></td>
<td>1 : 1 1 : M M : N</td>
<td><img src="dept_emp.png" alt="Department to Employee" /></td>
</tr>
<tr>
<td>Participation Constraints (Total)</td>
<td>Double-lined edge</td>
<td><img src="image1" alt="Diagram" /></td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-----------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Participation Constraints (Partial)</td>
<td>Single-lined edge</td>
<td>Participation of DEPT in HAS is partial in example above</td>
</tr>
<tr>
<td>Relationship Attributes</td>
<td>Oval in diamond</td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
<tr>
<td>Specialization</td>
<td>Rectangles connected with circle</td>
<td><img src="image3" alt="Diagram" /></td>
</tr>
<tr>
<td>Disjoint Constraint</td>
<td>‘d’ in circle</td>
<td><img src="image4" alt="Diagram" /></td>
</tr>
<tr>
<td>Overlapping Constraint</td>
<td>‘o’ in circle</td>
<td><img src="image5" alt="Diagram" /></td>
</tr>
<tr>
<td>Completeness Constraint (Total)</td>
<td>Double-lined edge from super class to circle</td>
<td><img src="image6" alt="Diagram" /></td>
</tr>
<tr>
<td>Shared Subclass</td>
<td>Edges connecting a single subclass to two other classes</td>
<td><img src="image7" alt="Diagram" /></td>
</tr>
<tr>
<td>Category</td>
<td>Edges connecting rectangles to a circle ‘U’</td>
<td><img src="image8" alt="Diagram" /></td>
</tr>
</tbody>
</table>
2.3.6 Other Enhancements

2.3.6.1 EER Model by Engels et al.

Following an object-oriented paradigm, Engels et al. proposed an Extended ER (EER) model for conceptual design covering both structural and behavioral aspects of a database application [Engels et al. 1992]. Since our research is not related to behavioral aspects, we restrict our discussion to the structural aspect only. Engels et al. has divided the specification of structural aspects into two categories – modeling of object structures and modeling of data types used for describing object properties. Though an object in their EER model can be an entity of an ER model, it permits complex objects as well (as in object oriented design methodology). Primary contribution of Engels’ EER model (restricting ourselves to static part of modeling of object structures) is extending the concept of attributes. Due to the object-oriented approach, attributes in their EER model are also categorized into data-valued attributes and object-valued attributes based upon their domains. The domain of a data-valued attribute is a data type (the case in ER model). On the other hand, the domain of an object-valued attribute is an entity type. Consequently, value of such an attribute is an instance of an entity type (that is, an entity). This approach uses pointer semantics to relate entities through object-valued attributes. Both data- or object-oriented attributes can be multi-valued – value being a set or a list of values from the corresponding domains. As opposed to a set, an element may occur more than once in a list. Further, a list has its elements enumerated so that those elements can be referenced by their position numbers.
2.3.6.2 **Higher-order Entity-Relationship Model (HERM)**

Thalheim emphasizes the need of an effective database design methodology which can provide theoretical basis for ER model in addition to its ability to represent semantic information [Thalheim 2000]. On one hand, well-defined semantics are the basis for the maintenance of the database through its entire life cycle; and on the other hand, in his opinion, “the theory is the basis upon which a designer can build a consistent schema that can be understood by other designers and consistently rebuilt during redesign or schema development. Based on this theory, a consistent design methodology and several design strategies can be developed”. However, he suggests that the desired methodology should not require the user to understand the theory or the implementation issues in order to design a database scheme. He proposes a Higher-order Entity-Relationship Model (HERM) [Thalheim 1989, Thalheim 2000] to meet these objectives.

Key features of HERM are:

1. HERM is an extension of the EER model which supports relationships of relationships and relationships of higher degrees.
2. It discusses efficient use of Is-A relationships (or generalization / specialization).
3. Weak entity types can be avoided.
4. A normal form theory is developed for HERM. Using this normal form theory, one can obtain normalized schemes as in the classical theory.
HERM has the following basic and extended modeling constructs:

- **Simple Attribute:** As in the EER model.
- **Nested Attribute:** Complex attributes are supported with the help of the following constructors:
  - **Tuple Constructor:** Using nested attributes, a new nested attribute can be defined by the Cartesian aggregation.
  - **Set Constructor:** Using a nested attribute, a new nested attribute can be defined by the set aggregation.
  - **Other Constructors:** Bags and Lists can also be used.
- **Entity:** Same as in the EER model.
- **Cluster:** A disjoint union of types whose identification type is domain compatible.
- **First-order Relationships:** These are defined as associations between single entity types or clusters of entity types. They can also be characterized by attributes.
- **Other Constructs:** Include higher-order relationships, integrity constraints and operations. Considered irrelevant, these are not discussed here.

2.3.6.3 **Business Requirements and Constraints (BRC)**

Khan et al. noted that the EER model does not have sufficient support to represent business constraints [Khan et al. 2004]. These constraints were used to be specified at the implementation level, as opposed to the conceptual level, in the form of domain
constraints, assertions and/or triggers in relational database management systems. This left a gap between requirements elicitation and database implementation. To bridge this gap, Khan et al. proposed an approach of specifying a wide range of business constraints in EER model. The objective was to enable the database designers specify business requirements and constraints at the conceptual modeling stage with minimal effort. In their approach, called attribute-oriented business requirements and constraints (BRCs), they classify five different categories of attribute oriented BRCs and introduce a construct to enhance the modeling capabilities and expressiveness of the EER model accordingly. The five categories identified in their research were based upon which attributes are involved in business constraints. These categories are summarized below:

- **Category (i) BRC**: Attributes of only one entity
- **Category (ii) BRC**: Attributes of both entities (participating in a relationship)
- **Category (iii) BRC**: Attributes of a relationship
- **Category (iv) BRC**: Attributes of a relationship and one entity
- **Category (v) BRC**: Attributes of a relationship and at least one attribute from both entities.

Fig. 2.12 demonstrates the proposed construct for the given business constraint.

### 2.4 Summarizing Evolution of ER Model

We have studied in previous sections that the ER model is a very convenient way of representing a real world problem. It has become quite popular among database community and so it is used for conceptual design stage prior to the logical design stage.
However, the ER model that we use today is a result of many modifications suggested in the original ER model at various times. All these modifications improved the semantic representation of the real-world problem. Table 2.3 gives a chronological summary of its evolution. Some of the work specified in the table is not directly related to our research and therefore is not discussed in these sections.

*Constraint:* Students with degree type ‘research’ must lease non-shared accommodation for at least six months

**Fig. 2.12: A BRC Notation**

It is worth mentioning here that conceptual models and their improvements are not restricted to just ER models. Some other data models or conceptual models have also been introduced, for instance, Natural language Information Analysis Method (NIAM) [Verheijen and Bakkum 1982], Object-role Modeling [Halpin 1990], and Unified Modeling Language (UML) [Rumbaugh et al. 1998]. Since these models are primarily not based upon constructs of ER models, we do not discuss these here.
### Table 2.3 Evolution of ER Model

<table>
<thead>
<tr>
<th>Work Done</th>
<th>By</th>
<th>When</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original ER model</td>
<td>Chen</td>
<td>1976</td>
</tr>
<tr>
<td>Generalization</td>
<td>Smith &amp; Smith</td>
<td>1977</td>
</tr>
<tr>
<td>Relationship Constraints</td>
<td>Scheuermann et al.</td>
<td>1979</td>
</tr>
<tr>
<td>Semantic Database Model (SDM)</td>
<td>Hammer &amp; McLeod</td>
<td>1981</td>
</tr>
<tr>
<td>Composite Attribute</td>
<td>Elmasri et. al.</td>
<td>1985</td>
</tr>
<tr>
<td>Multi-valued Attribute</td>
<td>Elmasri et al.</td>
<td>1985</td>
</tr>
<tr>
<td>EER Model</td>
<td>Elmasri et al.</td>
<td>1985</td>
</tr>
<tr>
<td>Higher-order Entity-Relationship Model (HERM)</td>
<td>Thalheim</td>
<td>1989</td>
</tr>
<tr>
<td>EER Model</td>
<td>Engels et al.</td>
<td>1992</td>
</tr>
<tr>
<td>Schema Transformations of ER Models</td>
<td>Assenova et al.</td>
<td>1996</td>
</tr>
<tr>
<td>Structural Validity of Relationship Constraints</td>
<td>Dullea et al.</td>
<td>1997-99</td>
</tr>
<tr>
<td>Fuzzy Constraints in EER Model</td>
<td>Galindo et al.</td>
<td>2001</td>
</tr>
<tr>
<td>BRC</td>
<td>Khan et al.</td>
<td>2004</td>
</tr>
</tbody>
</table>

In the following section, we study the next stage of database design methodology which is mapping a given ER diagram to a relational schema. This mapping or transformation is defined by a set of rules that we describe there.
2.5 Mapping ER Diagram to Relational Schema

As discussed earlier, the database design approach comprises of three stages:

1. Conceptual Design,
2. Mapping to Logical Model,

We have discussed in detail the conceptual design stage in previous sections. The output of that stage is a conceptual model, typically an ER model or an ER diagram. In the mapping stage, the ER diagram is taken as an input and it is transformed to a relational schema (assuming relational database is required). This transformation follows certain rules which are given in the literature [Chen 1976, Engels et al. 1992, Elmasri and Navathe 2004] and are described below.

Rule 2.1: Mapping Regular Entity Types – For each regular, that is strong, entity type in the ER diagram, create a relation R. All simple attributes of the entity type are added in R. One of the key attributes of the entity type is marked as primary key of the relation R.

Rule 2.2: Mapping Weak Entity Types – For each weak entity type in the ER diagram, create a relation R which contains all simple attributes of the weak entity type. Also add in R as foreign key the primary key of the relation corresponding to the owner entity type of the weak entity type being mapped. The primary key of R is the foreign key added in it together with the partial key attribute of the weak entity type.
**Rule 2.3:** Mapping 1:1 Relationship Type – For each binary 1:1 relationship type in the ER diagram, say S is the relation corresponding to one participating entity type and T is the relation corresponding to the other participating entity type in the relationship type under consideration. Add in T as foreign key the primary key of S. Also include in T simple attributes of the relationship type, if any.

**Rule 2.4:** Mapping 1:N Relationship Type – For each 1:N relationship type in the ER diagram, say S is the relation corresponding to the participating entity type on 1-side of the relationship type and T is the relation corresponding to the other participating entity type. Add in T as foreign key the primary key of S. Also include in T simple attributes of the relationship type, if any.

**Rule 2.5:** Mapping M:N Relationship type – For each M:N relationship type in the ER diagram, create a new relation R. Include as foreign keys in R the primary keys of the two relations that represent the participating entity types and their combination will be the primary key of R. Also include in R simple attributes of the relationship type, if any.

**Rule 2.6:** Mapping Multi-valued Attributes – For each multi-valued attribute A on every entity type E, create a new relation R which contains the primary key of the relation representing E (as foreign key) and the multi-valued attribute A itself. The primary key of R is all its attributes together.
2.6 Normalization Theory

In the database design methodology, the objective of designing a good relational database is to define a set of relation schemas which does not have data redundancy or data anomalies (also known as update anomalies). Codd identified these anomalies as insertion anomalies, modification anomalies and deletion anomalies [Codd 1972]. Various normal forms starting from 1NF to domain key normal form (DKNF), also known as 6NF, have been defined in the literature [Codd 1972, Codd 1974, Zaniolo 1976, Nicolas 1978, Fagin 1977, Fagin 1981, Tsou and Fischer 1982, Ullman 1988, Wang 1990, Hernandez and Chan 1991] which test a given relation to be in a specific normal form. Normalization can be considered as the process which determines the “goodness” or quality of a relation schema. Like ER model, this process also evolved over time. Initially, Codd proposed the first three normal forms called first normal form (1NF), second normal form (2NF) and third normal form (3NF) [Codd 1972] and later a stronger definition of 3NF, called BCNF, was proposed by Boyce and Codd [Codd 1974]. Observing that anomalies can still be there in a relation schema which is normalized up to BCNF, higher normal forms were later defined based upon multi-valued dependencies [Zaniolo 1976, Fagin 1977] and inclusion dependencies [Cosmadakis et al. 1990].

There are two main techniques for relational database design – top-down design and bottom-up design [Elmasri and Navathe 2004]. In bottom-up design, a universal relation which consists of all the database attributes, is defined and then based upon given set of dependencies the normalization algorithms, as defined in literature [Codd 1972, Codd 1974, Ullman 1988, Elmasri and Navathe 2004] are repeatedly applied. These algorithms
decompose the universal relation into a set of relations which satisfy a given normal form. On the other hand, the top-down design is defined in three steps [Teorey et al. 1986]:

1) Conceptual modeling

2) Mapping to logical schema, and

3) Normalization

as shown in Fig. 1.3. Comparing these two approaches, the bottom-up design approach has been criticized due to the problems of a universal relation [Kent 1981, Kent 1983]; whereas, top-down design is used most extensively in commercial database design [Elmasri and Navathe 2004]. In either case, normalization process requires a set of dependencies to be defined for every problem. This set of dependencies may have redundancy, that is, redundant dependencies, in itself which is eliminated by finding a minimal cover set of these dependencies. Although polynomial algorithms are available for computing a minimal cover, finding a minimal cover is a complex task for almost all practical problems. Further, in pursuit of defining a set of dependencies that represents all real world constraints of the problem, a database designer may include dependencies that are valid but irrelevant to the problem from normalization point of view [Ullman 1990]. Presence of these dependencies makes the task of normalization more difficult and identification of these valid but irrelevant dependencies is not easy either. Ramakrishnan observes that the normalization theory cannot discriminate among decomposition alternatives and that a designer has to consider the alternatives and choose one based on the semantics of application [Ramakrishnan and Gehrke 2003].
This gives us the motivation to review the normalization process and identify the reasons why a database schema resulting from the top-down approach is not normalized. We are further interested in analyzing these reasons and devising a methodology which can minimize or eliminate the need of normalization process. We would also like to propose an approach where schema alternatives can be compared and evaluated without designer’s subjectivity.

In this section we briefly present the normalization theory as available in the literature [Ullman 1990, Date 2003].

### 2.6.1 Related Definitions and Normal Forms

- **Functional Dependency:** A functional dependency defines a constraint between two sets of attributes in a relation schema. Let R be a relation schema defined as \( R = \{A_1, A_2, \ldots, A_n\} \) and r be its instance. A functional dependency (FD), written as \( X \rightarrow Y \), where \( X,Y \subseteq R \), defines a constraint that for any two tuples \( t_1 \) and \( t_2 \) in r that have \( t_1[X] = t_2[X] \), they must also have \( t_1[Y] = t_2[Y] \).

- **Non-trivial Functional Dependency:** A FD \( X \rightarrow Y \) is trivial if \( Y \subseteq X \); otherwise, it is non-trivial.

- **Closure of F, \( F^+ \):** For a given set of functional dependencies F, its closure \( F^+ \) represents a set of all functional dependencies that represents the same constraints as
specified by F. In other words, $F^+$ represents all such functional dependencies which can be inferred from F.

- **Equivalent sets of Functional Dependencies:** Two sets of functional dependencies E and F are equivalent if $E^+ = F^+$.

- **Minimal Set of Functional Dependencies:** A set of functional dependencies F is minimal if it satisfies the following conditions:
  1. Every FD in F has a single attribute on its right-hand side.
  2. For every FD: $X \rightarrow Y$ in F, $[(F - \{X \rightarrow Y\}) \cup \{Z \rightarrow Y\}]^+ \neq F^+$ where $Z \subset X$.
  3. For any FD: $X \rightarrow Y$ in F, $[F - \{X \rightarrow Y\}]^+ \neq F^+$

- **Minimal Cover for a Set of Functional Dependencies:** It is a minimal set of functional dependencies that is equivalent to the given set of functional dependencies. An algorithm [Elmasri and Navathe 2004] to find a minimal cover is given below:

**Algorithm 2.1:** Finding a Minimal Cover $F$ for a Set of Functional Dependencies

1. Set $F_0 = E$.

2. Replace every FD: $X \rightarrow Y$ in F, where Y is not a singleton set, by

   $n$ functional dependencies $X \rightarrow A_1$, $X \rightarrow A_2$, ..., $X \rightarrow A_n$

   where $Y = \{A_1\} \cup \{A_2\} \cup \ldots \cup \{A_n\}$

3. For each FD: $X \rightarrow Y$ in F,
For each attribute $A \in X$

If $[(F – \{X \rightarrow Y\}) \cup \{(X – \{A\}) \rightarrow Y\}]^+ = F^+$, then

replace $X \rightarrow Y$ with $(X – \{A\}) \rightarrow Y$ in $F$.

4. For each FD: $X \rightarrow Y$ in current $F$,

If $[F – \{X \rightarrow Y\}]^+ = F^+$, then

Remove $X \rightarrow Y$ from $F$.

- **Superkey of a Relation:** A set of attributes $S$ is a superkey of a relation schema $R = \{A_1, A_2, \ldots, A_n\}$ if $S \subseteq R$ and for any two tuples $t_1, t_2,$ in any legal state $r$ of $R$, $t_1[S] \neq t_2[S]$.

- **Candidate key of a Relation:** A key or a candidate key of a relation schema $R = \{A_1, A_2, \ldots, A_n\}$ is a superkey $K$ such that $(K-A_i)$ is not a superkey of $R$, that is, $K$ is minimal. Here $A_i$ refers to any attribute of $R$. It is to be noted that a relation may have more than one candidate key.

- **Prime and Nonprime Attributes of a Relation:** If an attribute $A$ is a member of any candidate key of a relation schema $R$, then $A$ is called a prime attribute; otherwise, it is nonprime.

- **First Normal Form (1NF):** This normal form disallows composite attributes, multi-valued attributes and their combinations in a relation. It states that domain of an attribute in a relation schema must have a simple, single, indivisible value.
- **Full Functional Dependency:** A FD: \( X \rightarrow Y \) is a full functional dependency if \((X - \{A\}) \rightarrow Y\) where \(A \in X\).

- **Second Normal Form (2NF):** A relation schema is in 2NF if every nonprime attribute \(A\) in \(R\) is fully functionally dependent on every key of \(R\).

- **Third Normal Form (3NF):** A relation schema \(R\) is in 3NF if for every non-trivial FD: \(X \rightarrow Y\) in \(R\),
  - either \(X\) is a superkey of \(R\), or
  - \(Y\) is a prime attribute of \(R\).

- **Boyce-Codd Normal Form (BCNF):** A relation schema \(R\) is in BCNF if for every non-trivial FD: \(X \rightarrow Y\) in \(R\), \(X\) is a superkey of \(R\).

### 2.6.1 Normalization Algorithms

In literature [Elmasri and Navathe 2004], one can find normalization algorithms which normalize a given relation schema to 3NF or BCNF. The input to all such algorithms is a set of functional dependencies \(F\) and a relation schema \(R\). As first step of these algorithms, a minimal cover is found for \(F\) and then the algorithm tests \(R\) for specific conditions. If violation of a certain condition (or normal form) is found, then the original schema is decomposed into a number of relation schemas. This is an iterative process and as stated earlier is quite time consuming.
2.7 Problems Identified

The literature studied in previous sections informs us that the entity-relationship model [Chen 1976] is still in a continuous process of improvement. It has been extended [Elmasri and Wiederhold 1980, Hammer and McLeod 1981, Elmasri et al. 1985, Teorey et al. 1986, Engels et al. 1992 and Thalheim 2000] as more complex requirements come up. Furthermore, on one hand, Codd observes that the task of capturing the semantics is a never-ending one [Codd 1979]; and on the other hand, Thalheim notes the shortcomings of the ER literature [Thalheim 2000] as the use of ER concepts often:

- lack a clear statement of the intended semantics
- applies different semantics to the same concept, and
- mixes semantics of different constructs.

In our research, we have observed that semantics of relationship attributes in the existing ER model are still not clearly represented. Since there is no notion and notation of a relationship attribute which can have more than one value for a single relationship instance, an anomaly arises. This anomaly, elaborated with examples in chapter 6, results in a relation schema which does not satisfy even 2NF. A solution proposed in the literature is to introduce artificial constructs (weak entity types and more relationship types) [Thalheim 2000] which do not exist in the real world. This makes the schema less understandable [Thalheim 2000]. So, it is a problem not only of accurate semantic representation but also of generating an un-normalized database schema.
We have also observed in previous sections that the ER model or any of its extensions does not discuss “goodness” of the relational schema obtained by transforming a given ER model to the logical model. Though transformation (mapping) rules are available, as given in section 2.5, the resulting relational schema may not satisfy, in some situations, even 2NF. Therefore, the normalization process with all its complexity, as given in section 2.6, has to be applied in order to remove data redundancies and anomalies.

Finally, we have observed that the same real world problem can be modeled in a number of ways using ER diagram. This raises questions like which of these representations is a better one, and how to know that. The literature study tells us that we do not have well-defined and structured rules to identify which of the two models has a better semantic representation or which one more closely models the real world. The next chapter describes quality issues found in the literature related to conceptual models.

2.8 Summary

We now conclude this chapter by summarizing the important findings as given below:

- Various constructs have been added from time to time in the classical ER model to fill in the semantic gap between real world and the database designer’s model. However, the ER model still has issues related to semantic representation.
- The ER model does not have a construct for clean representation of relationship attributes in order to distinguish between an attribute which can have a single value and the one which can have multiple values.
In the database design approach, process of normalization is followed because the relational schema transformed from an ER model may not be normalized. This process requires a significant amount of database designer’s time.

We do not have well-defined and structured rules to identify which of the two models has a better semantic representation for the same real world problem.
Chapter 3

Quality Aspect of Conceptual Models

The quality of a conceptual model has a direct impact on the quality of database schema and, in turn, on the quality of the ultimate software product. Moody therefore notes that improving the quality of conceptual models is a major step towards improving the quality of systems development process and the software product [Moody 1998]. Unfortunately, understanding and defining the software quality is not straight forward and same is true about the quality of a conceptual model. In practice, the quality of a conceptual model is either not taken care of or is evaluated in an ad hoc manner [Moody and Shanks 1994, Assenova and Johannesson 1996, Genero et al. 2001, Nelson et al. 2005]. In literature, quality of a conceptual model is defined in terms of lists of desirable characteristics. Mostly, these lists are unstructured, lack quantitative and objective measures, use imprecise and/or complicated definitions and usually overlap [Lindland et al. 1994, Genero et al. 2001, Moody 2005, Nelson et al. 2005]. There are few guidelines for evaluating the quality of conceptual models, and little agreement on what makes a “good” model [Janiesch05, Moody and Shanks 1994, Moody and Shanks 2003]. Therefore, this area is still wide open for research and demands finding answers to the following questions:

1. Can we understand and define quality of a conceptual model?

2. Can the quality of a conceptual model be understood only in qualitative terms or can it be measured quantitatively?
3. Different database designers may develop different ER models against the same database specifications. Can we compare two supposedly semantically equivalent conceptual models objectively?

4. If answers to the above questions are in affirmative, then can a model exhibiting poor quality be improved in an objective manner?

In the following sections, we present a summary of the research efforts already done in this direction. Comparatively being a new area of research, there is not much literature available on quality aspects of the conceptual models. Therefore, we provide an overview of the following work only:

- A Quality Framework by Lindland et al.
- Schema Transformations by Assenova and Johannesson
- A Quality Criteria by Teeuw & Berg
- Moody’s Quality Metrics
- Structural Complexity of ER Models by Genero et al.

However, in section 3.6, we shall briefly describe some other approaches on understanding the quality of conceptual models.

### 3.1 A Quality Framework

[Lindland et al. 1994] proposed a quality framework based upon a systematic approach to identifying quality-improvement goals and the means to achieve them. They distinguish between goals and means by separating what we achieve in conceptual modeling from how to achieve it. Interestingly, they make their goals realistic in the light of feasibility of the effort to achieve these goals. Their framework borrows three important linguistic
concepts - *syntax, semantics* and *pragmatics* - and applies these concepts to four aspects of conceptual modeling, namely, *language, domain, model* and *audience participation*.

*Syntax* relates the model to the modeling language by describing relations among language constructs without considering their meaning. *Semantics* relates the model to the domain by considering not only syntax, but also relations among statements and their meaning. *Pragmatics* relates the model to audience (those who are involved in modeling) participation by considering not only syntax and semantics, but also how the audience will interpret them.

The modeling *language* consists of all the statements that can be made according to the syntax. The *domain* consists of all possible statements which are correct and relevant for solving the problem. The *model* is the set of statements actually made and the *audience interpretation* is the set of statements that the audience thinks the model contains.

Based upon the notions described above, the framework [Lindland et al. 1994] defines three types of model quality:

- **Syntactic Quality** describes how well the model corresponds to the language,
- **Semantic Quality** relates to how well the model corresponds to the domain, and
- **Pragmatic Quality** defines how well the model corresponds to its audience interpretation.

Using their goal-means framework, these quality types are analyzed as follows:
➢ **Syntactic Quality:** There is only one syntactic goal: *syntactic correctness*, which means that all statements in the model are according to syntax. Means for this goal are: *error prevention, error detection* and *error correction*.

➢ **Semantic Quality:** There are two semantic goals: *validity* and *completeness*. *Validity* means that all statements made by the model are correct and relevant to the problem; whereas, *completeness* means that the model contains all the statements about the domain that are correct and relevant. These goals are made realistic by considering the feasibility, that is, cost-benefit analysis of achieving a certain model quality. Therefore, making a model perfect is not considered feasible, most of the time. As far as means are concerned, most semantic quality improvements can be done by manually inspecting the model. However, consistency checking can be automated which requires that the model must be in a formal language.

➢ **Pragmatic Quality:** There is only one pragmatic goal: *comprehension*, that is, how well the model has been understood by the audience. It is to be noted that goal is comprehension and not the comprehensibility, that is, the extent to which a model can be understood. Means for this goal include visualization, explanation, filtering and a tool support like a browser to navigate through the model quickly.

The research by Lindland et al. can be considered as a major effort towards understanding the quality aspects of a conceptual model. Moody has recently conducted an empirical testing of Lindland et al.’s framework and validated it [Moody et al. 2003,
Moody and Shanks 2003]. Later research further improved the Lindland et al.’s framework. For example, [Krogstie and Sølvberg 2003] introduced the element of participant knowledge in the quality framework as an additional aspect of conceptual modeling. [Krogstie 1998, Krogstie and Sølvberg 2003] noted that without the participant’s knowledge, a valid and complete correspondence between the conceptual model and the domain cannot be built or checked directly. However, we observe that the proposed frameworks did not suggest any quantitative measures either for goals or for means which can provide basis for comparing the quality of two conceptual models which possibly define the same domain. Without such measures, the understanding about quality of a given conceptual model depends on the perception or subjectivity of a database designer which varies from person to person. Our research in this thesis addresses this important issue, that is, quantitative and objective measurement of quality for comparing two models. Considering that means suggested by Lindland et al. for syntactic quality can easily be measured using metrics similar to those already used for software process quality or for software product quality as given in [Kan 2003], our research focuses on semantic quality and pragmatic quality only.

### 3.2 Quality Through Schema Transformations

To evaluate the pragmatic quality of a conceptual model, [Assenova and Johannesson 1996] proposed a set of quality criteria. The criteria defined homogeneity, explicitness, size of conceptual schema, rule simplicity, rule uniformity, query simplicity and stability as pragmatic quality parameters. They also introduced an approach [Assenova and Johannesson 1996] for improving schema quality which is based on schema
transformations. They introduced a number of schema transformations which can incrementally restructure schemas. They used their quality criteria to evaluate the effect of proposed schema transformations on the quality of conceptual model. This quality criteria, the schema transformations rules and their impact on quality are discussed in the following sub-sections.

### 3.2.1 Quality Criteria

The basic concepts in schema transformations proposed by Assenova and Johannesson is of an object (entity) and the cardinality constraints from Universe of Discourse (UoD). They distinguish between lexical and non-lexical objects. Lexical objects are data types like strings, integers, booleans, etc. and other objects are then non-lexical. The cardinality constraints specify for each attribute if it is single-valued, injective, total or surjective. These constraints are represented by a quadruple \((x,y,z,w)\) where \(x\), if the attribute is single-valued, set to 1 otherwise \(m\); \(y\), if the relation is injective, is set to 1 otherwise \(m\); \(z\), if the relation is total, is set to \(t\) (totality) otherwise \(p\) (partiality), and \(w\) indicating the surjectivity of the relation is set to \(t\) (if surjective) otherwise \(p\).

Assenova and Johannesson argue that though quality of schemas can be evaluated along three dimensions – syntax, semantics and pragmatic [Lindland et al. 1994], schema transformations cannot improve quality along all the three dimensions. According to them, schema transformations can only be used to improve the pragmatic quality of a schema, in particular the following aspects:

- **Homogeneity**: Object types should not contain highly dissimilar objects.
➢ **Explicitness:** Information about the UoD should be represented on the type level, not on the instance level.

➢ **Size:** Size of a schema is the number of its objects and attributes.

➢ **Rule Simplicity:** As many rules as possible are expressed by simple types of constraints in the schema e.g. cardinality constraints.

➢ **Rule Uniformity:** Cardinality constraints should be uniform, that is, attributes with a non-lexical range should be (1, m, t, t), and attributes with a lexical range should be (1, 1, t, p).

➢ **Query Simplicity:** It should be possible to retrieve simple information about the UoD through simple queries on the schema.

➢ **Stability:** When small changes in the UoD occur, it should only be necessary to make small changes to the schema in order to obtain a schema of good quality that reflects the UoD.

### 3.2.2 Schema Transformations

Schema transformations proposed by Assenova and Johannesson define rules for transformations of partial attributes, non-surjective attributes, partial attributes which are total in union, non-surjective attributes which are surjective in union, m-m attributes, lexical attributes, attributes with fixed ranges, to lattice structures and non-unary attributes.
They observed that the order in which these transformations are applied is important because otherwise a different schema would result. We describe here the first three transformations just to elaborate basics of their proposed approach.

- **Transforming Partial Attributes:** If an attribute in a schema is partial, that is, it is not applicable for all entities belonging to an entity type, then the schema is transformed by introducing a new subtype and changing the domain of this attribute accordingly. As a result of this transformation, the attribute becomes total which shows that the transformation increases the rule uniformity in a schema. Also, the size of the schema increases as another entity type is introduced. This transformation is demonstrated in Fig. 3.1 where the partial attribute *salary* gives rise to the introduction of a specialization EMPLOYEE in which *salary* becomes total.

![Diagram of Transforming Partial Attributes](Source:[Assenova and Johannesson 1996])

- **Transforming Non-surjective Attributes:** This rule suggests that a non-surjective attribute which has a range of non-lexical type can result in a schema transformation by introducing a new entity type. This transformation also increases the size of the schema, affects the cardinality constraints so that the
resulting schema becomes rule uniform with respect to surjectivity, and makes the schema more homogeneous as well as more stable.

- **Transforming Partial Attributes which are Total in Union:** This transformation is applicable when an entity type is the domain of a set of partial attributes but the union of those attributes is total. If the range of this partial attribute is a lexical type, an entity type that corresponds to the lexical type is introduced and the correspondence is represented by an attribute. Also, a new (total) attribute is introduced, whose range is a new entity type that is a generalization of the ranges of partial attributes. Such a transformation increases schema size, rule simplicity, rule uniformity with respect to totality, and stability of the schema. However, this transformation has a negative effect on query simplicity.

### 3.2.3 Impact on Quality

Assenova and Johannesson studied the influence of schema transformations on the quality criteria. Some transformations have positive effect on some criteria and negative effect on other criteria; whereas, some transformations affect some criteria both positively and negatively. They also concluded that the transformations did not necessarily produce a schema which is better than the original one and that the application of a transformation, therefore, always be at the discretion of the individual user.
However, we observe that, except *size of conceptual schema* (measured in terms of number of objects and attributes), their work did not introduce any quantitative measures. Also, their proposed transformations could be used to improve only the pragmatic quality [Assenova and Johannesson 1996].

### 3.3 Another Quality Criteria

[Teeuw and Berg 1997] proposed general quality criteria for conceptual models which was based on the quality types already proposed by [Lindland et al. 1994]. As syntactic quality was not their consideration, their quality criteria were devised to capture semantic quality and pragmatic quality only. Their quality criteria are defined in the following terms:

- **Completeness:** The concepts must be expressive enough to capture all essential aspects of the problem domain.
- **Inherence:** Concepts should be relevant and focus on essential aspects only.
- **Clarity:** A designer should be able to comprehend the concepts as well as rules clearly.
- **Consistency:** Concepts must not conflict with each other in representation of aspects of the real world.
- **Orthogonality (Modularity):** Independent aspects of the real world must be captured by different concepts.
- **Generality:** Concepts should be as independent as possible from any specific application or application domain.
It can be noted that definitions of *completeness* and *clarity* in their [Teeuw and Berg 1997] quality criteria are analogous to those of *completeness* and *comprehension* respectively, proposed in [Lindland et al. 1994]. However, we again observe lack of quantitative measures in the proposed criteria.

### 3.4 Quality Metrics of ER Models

Moody proposed eight different quality factors [Moody 1998] which can be used for evaluating the quality of a conceptual model or for comparing alternative representations of requirements. These quality factors and primary stakeholders involved in their evaluation are listed in Table 3.1.

In his research, Moody identified 25 metrics for these eight quality factors which are summarized in the following sub-sections.

#### 3.4.1 Completeness

It relates to whether the ER model contains all information required to meet user requirements. Moody rates this factor as the most important because if it is not satisfied then no other quality factor matters. Completeness mismatches in an ER model are categorized as follows:

- **Type I errors:** Represent those elements included in the data model that do not correspond to any user requirement or are out of the scope of system.
- **Type II errors:** represent those user requirements which are not represented anywhere in the data model. These errors refer to gaps and omissions.
Table 3.1 Moody’s Quality Factors

<table>
<thead>
<tr>
<th>Sr#</th>
<th>Quality Factor</th>
<th>Primary Stakeholder</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Completeness</td>
<td>Business User</td>
</tr>
<tr>
<td>2.</td>
<td>Integrity</td>
<td>Business User</td>
</tr>
<tr>
<td>3.</td>
<td>Flexibility</td>
<td>Business User</td>
</tr>
<tr>
<td>4.</td>
<td>Understandability</td>
<td>Business User</td>
</tr>
<tr>
<td>5.</td>
<td>Correctness</td>
<td>Data Analyst</td>
</tr>
<tr>
<td>6.</td>
<td>Simplicity</td>
<td>Data Analyst</td>
</tr>
<tr>
<td>7.</td>
<td>Integration</td>
<td>Data Administrator</td>
</tr>
<tr>
<td>8.</td>
<td>Implementability</td>
<td>Application Developer</td>
</tr>
</tbody>
</table>

The following four metrics are proposed for completeness:

- **Metric 1**: Number of items in the data model generating Type I errors
- **Metric 2**: Number of user requirements generating Type II errors
- **Metric 3**: Number of items in the data model generating Type III errors
- **Metric 4**: Number of inconsistencies with process model. This ensures that all functional requirements can be met by the model.
3.4.2 Integrity

It is the extent to which business rules or integrity constraints are enforced by the data model. Proposed metrics for integrity are as follows:

- **Metric 5:** Number of business rules which are not enforced by the data model.
- **Metric 6:** Number of integrity constraints included in the data model that do not accurately correspond to business policies.

3.4.3 Flexibility

It is the ease with which a data model can cope with business change. Proposed metrics are:

- **Metric 7:** Number of elements in the model which are subject to change in the future.
- **Metric 8:** Estimated cost of changes
- **Metric 9:** Strategic importance of changes, expressed as a rating by business users of the need to respond quickly to the change.

3.4.4 Understandability

It is defined as the ease with which the data model can be understood. Proposed metrics are:

- **Metric 10:** User rating of understandability of the model
- **Metric 11**: Ability of users to interpret the model correctly
- **Metric 12**: Application developer rating of understandability

### 3.4.5 Correctness

It refers to whether the model conforms to the rules of the data modeling technique being used. These rules include diagramming conventions, naming rules, definition rules, and rules of composition. Proposed metrics are:

- **Metric 13**: Number of violations to data modeling conventions
- **Metric 14**: Number of normal form violations. These are further classified into 1NF violations, 2NF violations, 3NF violations and higher normal form violations
- **Metric 15**: Number of instances of redundancy between entities

### 3.4.6 Simplicity

It means that the data model should contain minimum possible constructs. Proposed metrics are:

- **Metric 16**: Number of entities
- **Metric 17**: Number of entities and relationships
- **Metric 18**: Number of constructs. It includes entities, relationships and attributes.
3.4.7 Integration

It is defined as the level of consistency of the data model with the rest of the organization’s data. Proposed integration metrics are:

- **Metric 19**: Number of data conflicts with the Corporate Data Model
- **Metric 20**: Number of data conflicts with existing systems
- **Metric 21**: Number of data elements which duplicate data elements stored in existing systems or other projects
- **Metric 22**: Rating by representatives of other business areas to whether the data has been defined in a way which meets corporate needs rather than the requirements of the application being developed.

3.4.8 Implementability

It is defined as the ease with which the data model can be implemented within the time, budget and technology constraints of the project. Proposed metrics are:

- **Metric 23**: Technical risk rating
- **Metric 24**: Schedule risk rating
- **Metric 25**: Development cost estimate

3.5 Structural Complexity of ER Models

In this section, we discuss another important aspect of the quality of an ER model called its *structural complexity*. Structural complexity is an internal characteristic of an ER model.
The ISO/IEC 9126 standard [ISO99] defines software quality as composed of six external characteristics – functionality, reliability, usability, efficiency, maintainability and portability. These quality characteristics are further categorized into sub-characteristics. A characteristic like maintainability is usually measured only when the software has been released and hence it cannot provide insight about the software quality at early stages of the systems development life cycle (SDLC). This led the research to the identification of appropriate measures which can be used before the release of the software product.

In [Genero et al. 2002], Genero et al. have observed that the practical utility of quality metrics proposed by Moody (discussed in section 3.4) has not been demonstrated, in practice, probably due to their difficulty of use. They have presented a metric-based approach in [Piattini et al. 2001] for predicting maintainability of conceptual models. According to them, measures for structural complexity (internal attribute) of ER diagrams can be used to predict their maintainability which is an external attribute. Further, maintainability can be categorized into sub-characteristics like understandability and modifiability amongst others.

In their research [Genero et al. 2002], Genero et al. have proposed measures for the structural complexity of ERDs and have validated these measures. The measures and the corresponding constructs which are counted are defined in Table 3.2. Their research has shown that structural complexity can be considered to be an indicator of external quality attributes: understandability and modifiability of an ERD. Through a controlled
experiment on forty students of their university in which students were given sample ERDs, Genero et al. noted understandability time and modifiability time. Then computing Pearson’s coefficients, it was concluded that there was a high correlation between the understandability time and the modifiability time and the measures NE, NA, NR, N1:NR, NM:NR, NBinaryR.

Table 3.2 Structural Complexity Measures

<table>
<thead>
<tr>
<th>Sr#</th>
<th>Measure</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>NE</td>
<td>Total number of entities in an ERD</td>
</tr>
<tr>
<td>2.</td>
<td>NA</td>
<td>Total number of attributes</td>
</tr>
<tr>
<td>3.</td>
<td>NDA</td>
<td>Total number of derived attributes</td>
</tr>
<tr>
<td>4.</td>
<td>NCA</td>
<td>Total number of composite attributes</td>
</tr>
<tr>
<td>5.</td>
<td>NMVA</td>
<td>Total number of multi-valued attributes</td>
</tr>
<tr>
<td>6.</td>
<td>NR</td>
<td>Total number of relationships</td>
</tr>
<tr>
<td>7.</td>
<td>NM:NR</td>
<td>Total number of M:N relationships</td>
</tr>
<tr>
<td>8.</td>
<td>N1:NR</td>
<td>Total number of 1:N and 1:1 relationships</td>
</tr>
<tr>
<td>9.</td>
<td>NBinaryR</td>
<td>Total number of binary relationships</td>
</tr>
<tr>
<td>10.</td>
<td>NIS_AR</td>
<td>Total number of IS-A relationships (specializations)</td>
</tr>
<tr>
<td>11.</td>
<td>NRefR</td>
<td>Total number of reflexive relationships</td>
</tr>
<tr>
<td>12.</td>
<td>NRR</td>
<td>Number of redundant relationships</td>
</tr>
</tbody>
</table>
Understandability can be defined as the ease with which a given conceptual model can be understood. The assumption is that the person who is “understanding” a conceptual model is experienced enough in ER terminology and in application of the associated concepts. By now, understandability is defined subjectively in the literature and any attempts to define it quantitatively are also restricted to a Likert scale where subjects are asked to rate a given ERD at, as given in [Piattini et al. 2001]. The problem with this approach is that it ignores the fact that any two ERDs cannot be compared for understandability unless:

a) the two ERDs represent the same problem domain, and
b) both ERDs have the same degree of correctness.

The first condition is quite straightforward; whereas the second condition is important to consider because if an ERD has some missing entity types, relationships and/or attributes, it appears to be simpler than the one which has appropriately and correctly represented all the necessary information from the problem domain. This implies that the understandability can only be measured if we first have a definition and a proper measure for correctness of an ERD.

Likewise, modifiability can be defined as the ease with which a given conceptual model can be modified or changed when a new requirement or change is requested. Determining the ease with which any of these changes can be introduced requires an appropriate measure. One such measure available in the literature [Piattini et al. 2001] is the time
taken by the designer to introduce the requested change in the existing model. This is comparable to the software engineering metric called mean-time-to-change (MTTC) for maintainability of a software product. However, in case of ERD, this measure is of limited practical use for two reasons – one modification time varies not only from designer to designer but also for the same designer if modifications are done at different times; two measuring the time whenever a change takes place and then predicting the quality on the basis of it is simply impractical.

Through the above discussion, we understand that the characteristics like understandability and modifiability are though identified as important attributes to measure structural complexity, there is a need to define them quantitatively as a metric which can be easily applied to practical problems.

3.6 Some Other Approaches

It is worth mentioning here that, in addition to the above mentioned approaches to understand the quality of conceptual models, some other approaches have also been studied in parallel. For example, [Siau and Tan 2005] has adopted a cognitive approach to demonstrate that human cognition plays a pivotal role in understanding and improving the quality of conceptual models. This approach may not be ignored while studying quality parameters like comprehension and understandability introduced earlier in this chapter. Likewise, some researchers [Wand et al. 1999] have been interested in ontological analysis of common conceptual modeling constructs in order to improve the model quality in terms of capturing correct knowledge about an application domain.
Within semantic quality, for the validity goal [Lindland et al. 1994], some research efforts [Dullea and Song 1997, Dullea and Song 1998a, Dullea and Song 1998b, Dullea and Song 1999, Dullea et al. 2003] have focused on structural validity of ER diagrams. The outcome of these efforts is a set of rules to validate the semantics of relationship types. Janiesch [Janiesch 2005] has explored guidelines for conceptual modeling in the domain of web-based information systems. Pfeiffer et al. [Pfeiffer and Niehaves 2005] have proposed a two-dimensional framework to deduce the structural requirements of conceptual models. They use the structural approach from philosophy of science to propose an inner structure for conceptual models and then based on their framework they evaluate the quality of conceptual models. However, these approaches do not address any quantitative issues and hence are not our area of study in this research.

3.7 Problems Identified

Quality focus of ER models relatively being new area of research, we observe different dimensions of this quality. Some researchers, for instance [Lindland et al. 1994], provide just a framework which gives some insight on the process of measuring quality but not on the measures directly. Some researchers, for example [Teeuw9 and Berg 1997], have proposed quality criteria which are quite abstract and does not discuss how to measure their proposed parameters. On the other side, some researchers, for instance [Moody 1998] have proposed an extensive (or probably an exhaustive) list of quality parameters which are not easy to apply in practice. It is therefore observed that there should be defined a set of quality parameters which can be measured quantitatively and easily. Further, these measures should be valid to provide correct quality insight of a conceptual
model. This demands validation of the proposed metrics. Without such measures, two conceptual models claimed equivalent in semantic representation cannot be compared in terms of quality.

Once a set of quality metrics becomes available, we are in a position to comment on the quality of a given conceptual model. If this quality is measured to be low, then another question arises: how to improve the quality of this particular conceptual model. In [Assenova and Johannesson 1996], a set of schema transformation rules are provided for this purpose. We have two observations on their proposed approach. First, the proposed transformation rules are very complex therefore limiting their practicability. Second, they have concluded in their own research that the proposed transformations may not necessarily produce a schema which is better than the original one. It necessitates a quality improvement approach which suggests a set of rules for schema transformations which are easy-to-apply and every transformation yields a better schema.

Finally, we have noted that there are three dimensions of quality of a conceptual model: syntactic quality, semantic quality and pragmatic quality. Although every dimension has its own importance, semantic quality and pragmatic quality require more focus as syntactic quality can be considered as just adhering to the syntax of the model which is quite trivial. In our study, we did not find any approach which addresses both of these quality dimensions except for [Moody 1998]. We have already observed that metrics and quality parameters proposed in [Moody 1998] are very complex. Schema transformations by Assennova et al. improve pragmatic quality only. On the other hand, [Lindland et al.
1994] discuss semantic quality and its goals, but there are no quantitative measures. Hence, we need a practical approach which can improve pragmatic quality as well as semantic quality. It implies that appropriate measures should be defined for both of these quality dimensions.

### 3.8 Summary

In this chapter we have studied quality aspects of a conceptual model. This study explains three dimensions of quality namely syntactic quality, semantic quality and pragmatic quality. A framework is discussed which gives insight about the process of quality measurement along these three dimensions. We have studied schema transformation rules which can improve pragmatic quality of a conceptual model. Various quality factors as completeness, integrity, flexibility, understandability, correctness, simplicity, integration and implementability are identified in the literature along with their associated metrics. Another important quality parameter is identified as structural complexity of an ER diagram in the conceptual model. Some measures for structural complexity are also studied. We have identified the need of defining an appropriate approach which can define easy-to-apply metrics for syntactic quality and pragmatic quality, and can suggest rules to improve the quality of a given conceptual schema.

Starting with the next part of this thesis, that is part II, we elaborate our proposed approach in an attempt to solve most of the problems identified in chapter 2. In part III of the thesis, we propose solutions for problems identified in this chapter.
Part II

IMPROVING CONCEPTUAL DESIGN
Chapter 4

Improving Conceptual Design With Functional Dependencies

4.1 Introduction

As stated in chapter 2, the quality of a conceptual model (an ER model, in our case) can be determined by how close and how well it represents real world information including business rules and other constraints. Ignoring these constraints may lead to an incorrect database schema which in turn may lead to failure of a software system. Hence, an ER model which incorporates most of these constraints has a better representation of the mini-world. The existing ER models represent these constraints in the form of:

- key attributes, enforcing uniqueness amongst entities,
- single-valued or multi-valued attributes, enforcing number of values an attribute can take on for an entity,
- cardinality of relationship types, enforcing how many instances of one entity type can participate in the relationship with another entity type,
- participation constraints which enforce whether every entity participates in the relationship or not, and
- weak entity types, enforcing existence dependency on other entity types.
Another important design issue is deciding a certain concept to be modeled as an attribute or an entity type, or a concept to be modeled as a composite attribute, a regular entity type or a weak entity type. In very simple situations, it may sound trivial; but in problems of practical size, these decisions are not straightforward [Navathe 1992, Silberschatz et al. 2005]. Usually, database designers apply some kind of heuristics to answer such questions. For instance, regarding the decision about an entity type versus an attribute, one common heuristic says that “anything about which you want to store information is an entity type”. In the following example, we show that the heuristic like this may fail to yield a good model.

Consider a sales scenario where customers place orders. Assuming every order is uniquely identified by its order# and for a given order#, other attributes of the order (e.g. customer name, customer phone) can be functionally determined. Two possible ER models for this scenario are shown in Fig. 4.1 and Fig. 4.2. The heuristic stated above favors the model shown in Fig. 4.2 in which two entity types ORDER and CUSTOMER are created because we want to store information about both of these “things”. However, it can be noted that Fig. 4.2 is not a good model because CUSTOMER should not be created as an entity type due to the fact that every customer may not have a unique name. Hence, Fig. 4.1 is a better representation under the given scenario. In other words, the decision about better semantic representation is based upon the concept of a functional dependency and not upon the heuristic. Note that in this scenario, we assert that the FD: Cust_Name → Cust_phone does not hold.
We therefore observe that functional dependencies play an important role in modeling real-world problems. They represent important real world constraints but in current practice they are not considered while developing an ER model. The result is an ER model which misses important information and hence is not a “good” representation of the real world problem.

Based upon the concept of a functional dependency, we can further analyze the scenario. It should be noted that Fig. 4.1 is a better model as long as we have only the following functional constraint,

\[ \text{FD: } \text{Order} \# \rightarrow \text{Cust}_\text{Name}, \text{Cust}_\text{Phone}, \text{Date} \]

![Fig. 4.1: An Entity Type: Sale_Order](image)

If in our Universe of Discourse (UoD) there exists an additional constraint,

\[ \text{FD: } \text{Cust}_\text{Name} \rightarrow \text{Cust}_\text{Phone} \]

then Fig. 4.1 is not a good model anymore as it does not represent the stated constraint. This case makes Fig. 4.2 a better representation as the constraint is properly represented through the entity type CUSTOMER in it.
Therefore, we understand that the semantic gap between a real world problem and its corresponding ER model can be reduced by incorporating functional dependencies in it. However, functional dependencies should not be incorporated in the ER model in a subjective manner. It should be studied whether a systematic well-defined approach can be followed, for this purpose, or not. This becomes the motivation of our work in this chapter in which we attempt to define a systematic approach to improve representation of a real world problem in an ER model using functional dependencies. Our proposed approach is expected to give the following benefits:

a) it narrows down the gap between real world and the ER model,

b) it helps resolve very common design issues arising while developing an ER model, for instance, a particular concept should be modeled as an entity type or an attribute,

c) the resulting ER model is easier to be understood by the database designer, as it represents most of the constraints required in database design, and

d) it makes the ER model more flexible to take care of future enhancements or modifications in the real world problem.
It is worthwhile to note that application of functional dependencies at the conceptual level is not a novel idea. In literature, we find [Jajodia et al. 1983, Ling 1985, Thalheim 2000] amongst other researchers discussing functional dependencies in ER models. However, we have the following observations on their work:

**Thalheim’s Approach**

1. it was restricted to understanding justification of weak entity types only,

2. it was informal and did not define at all any approach or structured rules towards semantic improvement of an ER model, and

3. it applied functional dependencies on ER model without defining what a functional dependency was (as a matter of fact, we have done the same thing while giving examples above). One might ask that a functional dependency is a constraint on tuples of a relation – a concept of relational theory (as discussed in section 2.5.1), so how can we use it in the context of conceptual design, one stage earlier when relations are not yet defined.

**Other Researchers’ [Jajodia et al. 1983, Ling 1985] Approach**

4. it was a rigorous and complex approach focused towards achieving a NF-ERD which would always generate relations satisfying a certain normal form,

5. the objective was not the semantic improvement of an ER model, and

6. once again, as stated above, the concept of functional dependencies was not defined for the conceptual level.
Therefore, we start the next section by first defining new concepts of *functional dependency*, *key*, *key attribute* and *non-key attribute* without involving relational model. Then, in subsequent sections, we elaborate our proposed approach for improving the conceptual model. We call this approach *schema transformations*. It is interesting to note that schema transformations also take place from one database design stage to another. For example, a conceptual schema is transformed to a relational schema which is later transformed to a physical schema. But our approach transforms all schema while staying at the conceptual level and therefore corresponds to the stage 2 of our proposed database design methodology shown in Fig. 1.4.

4.2 Definitions

In this section, we propose definitions for new concepts which are required to formally define our proposed approach of schema transformations. These definitions do not require any theory from relational model and are defined independent of it as given below:

**Definition 4.1**: There exists a functional dependency $X \rightarrow Y$ for an entity type $E$ if for every entity in $E$ the set of attributes, $X$, uniquely determines the attribute $Y$. Formally, let $e_i$ represent an $i^{th}$ entity in $E$ where $i=1,2,3, ..., n$ and $e_i[X]$ represent the value of attribute $X$ for $e_i$, then there exists a FD $X \rightarrow Y$ if for any two entities $e_i$ and $e_j$ in $E$:

$$e_i[X] = e_j[X] \text{ implies } e_i[Y] = e_j[Y]$$
**Definition 4.2:** Let $\text{Attr}(E)$ represent the set of all attributes of an entity type $E$ and there exists a set of FDs $X \rightarrow Y_i$ where $Y_i \subset \text{Attr}(E)$ such that $X U( Y_i) = \text{Attr}(E)$ and there does not exist any attribute $Z$ of $E$ such that $Z \in X$ and $X-\{Z\} \rightarrow \text{Attr}(E)$ then $X$ is a key of $E$.

**Definition 4.3:** Let $K$ be a key of an entity type $E$ and there exists an attribute $A$ such that $A \in K$ then $A$ is a key attribute of $E$.

**Definition 4.4:** Let $K$ be a key of an entity type $E$ and there exists an attribute $A$ such that $A \not\in K$ then $A$ is a non-key attribute (NK) of $E$.

These definitions are analogous to those in the relational model. At this point, one may suggest that there is no need of re-defining these concepts if they already exist in the relational model. However, it is important to realize the following:

1. As already stated, the concepts of a functional dependency and other associated terms (defined above) are defined in the literature only for the relational model. There does not exist definitions for these concepts at the conceptual level. A well-structured approach that we define in this chapter cannot be based on a concept which is not formally defined.

2. The relational model was proposed by Ted Codd in 1970 [Codd 1970]. Six years later, in 1976, Peter Chen proposed the fabulous ER model [Chen 1976] which is now widely practiced in the industry. Most of the concepts in ER model have
their equivalence in the relational model. For instance, the concept of an entity type is very similar to that of a relation; the attributes are the same as attributes (in the relational model), a key attribute is the same as a candidate key, and so on. This correspondence does not eliminate or alleviate the need or significance of the ER constructs and terminology; rather, it makes the mapping process easier and more intuitive. By virtue of this correspondence, it is very natural that a constraint which is defined on a relation (that is, a functional dependency) can be defined as a constraint on an entity type. However, it cannot be implied that the concept need not be defined at the conceptual level or it has a little significance. Hence, definitions of a functional dependency and the associated terms are required at the conceptual level and would correspond to the concepts in the relational model.

3. Major advantages of defining these concepts at the conceptual level become evident when we demonstrate in the following sections that:

   a. an approach like schema transformations can be devised to improve a conceptual model, and

   b. it provides answers to important design questions like “Should a concept be modeled as an entity type or an attribute?” which otherwise are not that simple [Silberschatz et al. 2005].

Based upon these definitions, we define rules for schema transformations in the next section. These transformations are based upon:

- Composite Attributes
- Non-key attributes
in an ER model.

4.3 Schema Transformations Based On Composite Attributes

While developing an ER model, a designer is often confronted with the problem of representing a concept as an entity type or an attribute. This is a very important design decision which needs to be addressed objectively. In this section, we propose rules which are helpful in making such a decision.

4.3.1 Case I: An Entity Type Having A Composite Key

If a composite attribute is a key then, by Definition 4.2, it determines all other attributes of the entity type. So marking it as a composite attribute (and not making it an entity type) should be preferred.

Consider a part of the sales system of an organization in which ORDER is identified as an entity type and Order#, (Order) Amount and ShipMode are its attributes. Order# is further divided into Type and Serial_Number (Sr#). Order# is a key for this entity type. ShipMode refers to the mode of shipment which can take on data values like ‘By Air’ or ‘By Sea’. A segment of ER diagram for this part of the problem is represented in Fig. 4.3.
Our discussion of this case suggests that Order# should not be marked as an entity type and it should stay as a composite attribute.

![Diagram of an Entity Type having a Composite Key](image)

**Fig. 4.3: An Entity Type having a Composite Key**

However, this case further has a special case when an attribute, which is part of this composite attribute (key as well), functionally determines another attribute (non-key). This is discussed as *Case IA* below:

**Case IA:  A KEY ATTRIBUTE DETERMINES A NON-KEY ATTRIBUTE**

Say, in the above example:

\[
\text{Type} \rightarrow \text{ShipMode}
\]

Fig. 4.3 does not represent this FD, so it should be improved. In this case, a new entity type (ORDERTYPE) is to be created which has attributes Type and ShipMode, Type being the key attribute of the new entity type. As a result, ORDER now becomes a weak entity type of ORDERTYPE with Sr# as its partial key attribute.
Fig. 4.4 now represents the given real world constraint clearly. The discussion of this case can be stated in *Schema Transformation Rule 4.1* as:

**Rule 4.1:** For every functional dependency $A \rightarrow X$ where $A \in Attr(\text{Key})$ and $X$ is a set of non-key attributes of $E$:

- **a)** there can be defined an entity type $E'$ such that $\text{Attr}(E') = \{A\} \cup X$ and $A$ is a key attribute of $E'$,
- **b)** $E$ is converted into a weak entity type which no longer holds $A$ and $X$ as attributes and where $\text{Attr}(\text{key}) - \{A\}$ is the set representing partial key of the weak entity type $E$, and
- **c)** there can be defined an identifying relationship type between $E$ and $E'$.

*Case I* discusses those scenarios in which we have a composite key. We also observe important findings when there exists a composite attribute which is not a key. This is discussed in *Case II* here.

![Fig. 4.4: Improving Fig. 4.3 when Part of a Key Determines a Non-Key Attribute](image-url)
4.3.2  **Case II: An entity type having a composite attribute which is NOT a key**

Let A be a composite attribute of entity type E which is further broken down into simple attributes $A_i$ where $i=1,2,\ldots,n$. Now one of the following three scenarios must exist:

a) $\forall i \exists A_j | A_j \rightarrow A_i$

b) $\forall i \forall j | A_j \not{}\rightarrow A_i$ (i.e. $A_j$ does not functionally determine $A_i$)

c) $\exists i \exists j \exists k | A_j \rightarrow A_k \land A_j \not{}\rightarrow A_i$

where $i = 1,2,\ldots,n; \ j = 1,2,\ldots,n; \ k = 1,2,\ldots,n$ and $i \neq j, \ j \neq k, \ i \neq k$

**Scenario (a)** implies that $A_j$ is a key attribute because it uniquely determines all other attributes. Hence A can be marked as an entity type having $A_j$ as its key attribute and A forms a relationship with E.

**Scenario (b)** implies that:

i) A cannot be modeled as an entity type, or

ii) no real world attribute distinguishes entities of type A if A is modeled as an entity type, or

iii) there may be a real world distinguishing attribute but the enterprise is not interested in keeping track of distinguished entities of type A if A is modeled as an entity type.
In any of these cases, A should not be modeled as an entity type.

*Scenario (c)* implies that:

i) A itself cannot be modeled as an entity type because \( A_j \) does not determine all other attributes, and

ii) It is converted into case III (section 4.4) if we consider all simple attributes of A as simple attributes of E.

These scenarios are elaborated with examples below:

*Scenario (a)*

Consider the ERD in Fig. 4.5.

![Fig. 4.5: Attribute Dependencies in a Composite Attribute](image)

**Fig. 4.5: Attribute Dependencies in a Composite Attribute**

Customer is a composite attribute which has Name and Phone as its simple attributes. Say,

\[ \text{Name} \rightarrow \text{Phone, Type} \]
This ERD can now be changed into Fig. 4.6. It clearly represents the additional constraint given by:

\[ FD: \text{Name} \rightarrow \text{Phone, Type} \]

\[ \text{CUSTOMER} \]

\[ \text{ORDER} \]

\[ \text{Amount} \]

\[ \text{Order#} \]

\[ \text{Places} \]

**Fig. 4.6: An Improvement in Fig. 4.5**

whereas it does not exist in Fig. 4.5. Further, newer design can be appreciated in terms of its flexibility. For example, if now we are interested in keeping track of the date a customer places a particular order, it can be easily represented as a relationship attribute of relationship type Places instead of a less clear representation when it would be marked on ORDER (in Fig. 4.5)

**Scenario (b)**

Say, this time the

\[ FD: \text{Name} \rightarrow \text{Phone, Type} \]

does not hold either. Though, customer is a real world entity but no defined attribute can uniquely distinguish customer entities. In such a situation we shall not create an entity CUSTOMER. That means Fig. 4.5 is semantically correct and should not be changed.
**Scenario (c)**

Consider the ERD in Fig. 4.7.

![Figure 4.7: Part of a Composite Attribute Determines Another Part](image)

Say, this time the FD:

\[
\text{Name} \rightarrow \text{Ph#}
\]

does not exist but the

\[
\text{FD: Ph#} \rightarrow \text{Country}
\]

does exist. It leads to the following improvement:

![Figure 4.8: An Improvement in Fig. 4.7](image)

It is to be observed that Ph# is a key of PHONE.
All the scenarios given above can be formulated in *Schema Transformations Rule 4.2* as follows:

**Rule 4.2:** For every functional dependency $A_j \rightarrow X$ where $A_j$ is a simple attribute of $A$ being the composite attribute of an entity type $E$ and $X$ is a set of all simple attributes of $A$, there can be defined:

1. an entity type $E'$ such that $\text{Attr}(E') = \{A_j\} \cup X$ and $A_j$ is a key attribute of $E'$,
2. a relationship type between $E$ and $E'$

and the composite attribute $A$ no longer holds $A_j$ and $X$.

In the next section, we discuss the case when a set of non-key attributes determines another non-key attribute of the same entity type.

### 4.4 Schema Transformations Based On Non-Key Attributes

In the previous section, we studied schema transformations for improving a conceptual model based upon the notion of composite attributes in an ER model. In this section, we observe that schema transformations can be suggested on the basis of non-key attributes present in an ER model. In this context, we identify the following two cases:

**Case III:** A single NK attribute determines another NK attribute of the same entity type
Case IV: More than one NK attribute determines another NK attribute of the same entity type

Let us discuss these cases one by one:

4.4.1 Case III: A Single NK Attribute determines another NK Attribute

Let A and B be any two NK attributes of an entity type E and there exists a FD:

\[ A \rightarrow B \]

It implies that the given ER model does not successfully represent this real world constraint (the above FD) because the entity type specified with these two non-key attributes is unable to show this inter-relationship. Hence, another entity type, say E', having A and B as its attributes should be created. Since \( A \rightarrow B \), by Definition 4.2, A is a key of E'. In addition, E' should form a relationship type with E because it is created out of the attributes of E. Now, let us see with a practical example, how these changes in an ER model improve real world representation.

Let us revisit our sales system example. Say, Order# is key this time as represented in Fig. 4.9 and Supplier is another attribute which refers to supplier’s name and the ShipMode refers to the same mode of shipment which can take on data values like ‘By Air’ or ‘By Sea’.
Let us say that the following functional dependencies exist:

\[ FD1: \text{Order}\# \rightarrow \text{Supplier, ShipMode} \]
\[ FD2: \text{Supplier} \rightarrow \text{ShipMode} \]

Here, Supplier and ShipMode both are non-key attributes. Hence, Fig. 4.9 can be modified as Fig. 4.10:

Note that \( FD2 \) implies that supplier entities can be uniquely identified by supplier (i.e. supplier name). Hence SUPPLIER can be created as an entity type. The new ERD clearly represents the additional constraint given by \( FD2 \) whereas it does not exist in Fig. 4.9.

This discussion leads to formulation of the following schema transformations rule:
**Rule 4.3:** For every functional dependency $A \rightarrow B$ of an entity type $E$ where $A$ and $B$ are non-key attributes of $E$,

a) there can be defined a new entity type $E'$ such that $\text{Attr}(E') = \{A\} \cup \{B\}$ and $A$ is a key of $E'$,

b) there can be defined a relationship type between $E$ and $E'$ with appropriate cardinality, and

c) $\text{Attr}(E) := \text{Attr}(E) – (\{A\} \cup \{B\})$

Now, we discuss the next case.

**4.4.2 Case IV: More than one NK Attribute determines another NK Attribute**

In this case, the left hand side of the functional dependency has more than one NK attribute, that is, considering:

$\text{SNK} = \{A_1, A_2, \ldots, A_n\}$ be a set of all the non-key attributes of an entity type $E$ then,

$\exists i \forall X \mid X \subseteq \text{SNK} \land X \rightarrow A_i \quad \text{where } i = 1, 2, \ldots, n$

Once again, the FD: $X \rightarrow A_i$ implies that the given ER model does not successfully represent this real world constraint because the entity type specified with these non-key attributes is unable to show this inter-relationship. Hence, another entity type having $X$ and $A_i$ as its attributes should be created. In this case, $X$ becomes a key of the new entity type, say $E'$. In addition, $E'$ should form a relationship type with $E$ for the same reason discussed in case 3.2.1.
Since this discussion allows X to be a singleton set, case III becomes a special case of this one. Hence, *Rule 4.3* can be generalized as *Rule 4.4* which is given below:

**Rule 4.4:** For every functional dependency $X \rightarrow N$ of an entity type $E$ where $N$ is a non-key attribute of $E$ and $X \subseteq S_{NK}$ where $S_{NK}$ is a set of all the non-key attributes of $E$,

a) there can be defined an entity type $E'$ such that $\text{Attr}(E') = X \cup \{N\}$ and $X$ is a key of $E'$,

b) there can be defined a relationship type between $E$ and $E'$ with appropriate cardinality, and

c) $\text{Attr}(E) := \text{Attr}(E) - (X \cup \{N\})$

This completes our discussion on various cases that are observed to improve a given ER model.

### 4.5 Schema Transformations and Normalization

In this section, we try to answer some important questions that may be raised on our proposed approach of schema transformations. For example:

1. Transformation rules seem similar to the normalization rules. Are we suggesting to apply normalization rules at the conceptual level?
2. If normalization rules are to be applied anyway, then why not at the logical level?
3. If the objective of schema transformations is to obtain a high quality ERD, then can we start with any ERD, map it to a relational schema, normalize it to get a high quality relational schema and then reverse engineer that normalized schema to generate a high quality ERD?

Due to the importance of these questions, we would like to address all necessary aspects of these questions. Since these questions are inter-dependent, these are answered collectively, as follows:

- While developing transformation rules, our prime interest is in defining a systematic approach for semantic improvement. Generating a normalized schema is amongst one of our desirable goals (from quality perspective) that may or may not be achieved through schema transformations. That is, we do not favor normalization at the cost of semantic representation. In terms of preemptive goal programming [Hillier and Lieberman 2005], semantic improvement is our primary goal of priority 1 and increasing degree of normalization is our secondary goal of priority 2. In other words, a model which generates a highly normalized database cannot be considered better (at the conceptual level) than a model which has a superior semantic representation. Therefore, we are not proposing to move the normalization process at a higher level as such. More discussion on this topic will follow in section 5.1.

- Schema transformation rules can be devised which are independent of the normalization rules; [Assenova and Johannesson 1996] is one such example discussed in section 3.2.2. Though our research is primarily focused on improving the semantic
quality; other quality attributes including the degree of normalization (as recommended by many researchers [Fahrner and Vossen 1995, Moody 1998] and as defined in later part i.e. Part III of the thesis,) are also an integral part of our quality measurement approach. We are interested in devising schema transformations that can improve semantic quality as well as the degree of normalization. Hence, this is a deliberate effort to define rules which can help achieve both of these targets.

The two approaches (one in which schema transformations are applied to the ER diagrams and the other in which normalization is exercised at the logical level) cannot be considered equivalent. This is the reason many well-known researchers [Jajodia et al. 1983, Ling 1985, Markowitz and Makowsky 1990] were interested in proposing a methodology for normalization of ER diagrams (we repeat, this is however not our prime concern). They were convinced that:

- Normalization does not necessarily result in a schema which maps to the real world (lack of semantic representation), hence making the software maintenance task more difficult, and
- Applying normalization like process at the conceptual level results in simplification of the design process.

Markowitz shows that “the assumptions underlying normalization make the normalization techniques difficult to use and unreliable. Conversely, EER-oriented design favors the realization of many normalization goals” [Markowitz and Makowsky 1990].
While developing transformation rules, although we considered the normalization rules as a guideline (for degree of normalization index is amongst one of our quality measures); the transformation rules are not the same as normalization rules. Our schema transformations have their own terminology and scope of application. It is important to note that the proposed transformation rules help us understand and resolve the conceptual differences which cannot be understood by the normalization theory. Even at conceptual level, presently, it is not easy to find answers for these basic questions [Silberschatz et al. 2005]. For example, should a given concept be marked as an attribute or as an entity type? Or, should a concept be represented as a multi-valued attribute or as an entity type? Normalization rules do not answer these questions. Transformation rules distinguish situations where an entity type should be represented as a strong entity type or as a weak entity type (section 4.3); whereas, in normalization, we only have relations without any distinction of what types of entities they are representing.

It is well-established in the research [Markowitz and Makowsky 1990, Fahrner and Vossen 1995] that reverse engineering a relational schema may not generate a good quality ERD.
4.6 Summary

It is identified in this chapter that the concept of functional dependency defines important real world constraints. Incorporating these constraints in an ER model reduces the semantic gap between real-world and the model, and therefore satisfies one major objective of carrying out this research. We present in this chapter a systematic approach which can identify that a given ER model lacks representation of functional dependencies. This approach then suggests rules to improve the ER model. These rules are called schema transformation rules. We also proposed definitions for the concepts which are required to define schema transformation rules appropriately. Our schema transformations rules are based upon two broad situations – when there is a composite attribute of an entity type and when a non-key attribute determines another non-key attribute. Analyzing the actual situation, our rules introduce a weak entity type or a strong entity type and appropriate relationships in the ER model to make it closer to the real-world.
Chapter 5

Impact On Normalization

5.1 Introduction

In the previous chapter, we have proposed an approach of schema transformations to improve an ER model. The objective was to improve the model semantically in order to reduce the gap between real-world problem and its corresponding ER model. Choosing a criterion for evaluating semantic improvement can be considered subjective but we have successfully presented functional dependencies as an appropriate objective criterion for this purpose and have demonstrated, in the previous chapter, their effectiveness with a number of examples.

In this chapter, we are interested in finding out the effectiveness of our proposed approach in later stages of the database design methodology, primarily the normalization stage. The motivation is that in spite of bringing semantic improvement, our proposed schema transformation rules should not have at least a negative impact on the normalization process. That is, relational schema obtained after mapping should not define relations which are in lower normal forms as compared to the one which is formed without applying schema transformation rules. Rather, what we are looking forward to is a better
relational schema (that is, in higher normal form) after using our proposed schema transformation rules. Ideally, our target is achieving BCNF being the most common normal form, in practice.

This aspect of our research raises two questions. Firstly, is there any relationship between an ER model and the normal form of the corresponding relational schema? Secondly, following software quality assurance approach, if a relational schema obtained after mapping an ER model does not satisfy certain normal form and it is taken as a “defect”, can we trace back this defect to the previous design stage, that is, the corresponding ER model?

As far as the first question is concerned, we find in the literature [Jajodia et al. 1983, Ling 1985] that an ER model, in general, does not guarantee any particular normal form its schema is transformed to. This particular problem has motivated some researchers to define a normalized ER diagram [Jajodia et al. 1983, Chung et al. 1983, Ling 1985] which is translated to a relational schema satisfying some specific normal form directly. We do not intend to define any such normalized ER diagram. Our motivation is to find or evaluate the impact of our proposed rules on to the normalization process.

After knowing that a relational schema translated from a conceptual schema may not satisfy BCNF (or even lower normal forms), the second question becomes more important, that is, to identify the root causes of getting such a relational schema. In order to address both of these questions, either a top-down approach or a bottom-up approach
can be proposed. In top-down approach, we start with a conceptual model, develop an ER
diagram, apply proposed schema transformation rules, map the improved ER diagram to
generate a relational schema and finally check the highest normal form all relations
satisfy. Following a bottom-up approach, one can start with a relational schema which
does not satisfy BCNF, for instance, and then backtracking we explore what was “wrong”
in our conceptual model. After identifying cause of the “defect”, we investigate that had
schema transformation rules applied, the “defect” would have been eliminated or not.

Considering the importance of second question, we have adopted the bottom-up approach
in this chapter. Therefore, we discuss how various constructs of an ERD are transformed
into relations and investigate how violations of normal forms might occur. Then we
suggest the changes in an ERD which would remove the “defect” of un-normalized
relations. As a final part of this approach, we compare our suggestions with the
suggestions already proposed in chapter 4 for semantic improvement, and observe an
important finding that getting un-normalized relations from an ER model is an indication
that semantics of the application are not properly represented in the model.

In the following section, we first convert the mapping algorithm given in section 2.4 into
formal terms for the sake of precision. The formal definition would also help us in
extending this algorithm for transformation of new constructs proposed in later chapters.
Then we investigate each normal form starting from 1NF to BCNF in a separate section
and follow our approach outlined above. While discussing the case of BCNF, we present
a theorem and its formal proof observing an important property of our proposed rules.
Towards last section of this chapter, we further explore and prove interesting characteristics about those relations which satisfy 3NF but not BCNF. These characteristics are important to study the dependency preservation problem which in turn is related to the semantics of the model and is explained in that section.

5.2 ER-To-Relational Schema Mapping

In order to transform an ER schema to a relational schema (stage 3 of our database design methodology), rules have been given in section 2.4. Here, we define these rules formally in the form of Algorithm 5.1 and the nomenclature used is given in Table 5.1.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ξ</td>
<td>ERD represented as a set of entity types and relationship types amongst these entity types</td>
</tr>
<tr>
<td>SE&lt;sub&gt;&lt;i&gt;&lt;/i&gt;&lt;/sub&gt;</td>
<td>&lt;sup&gt;i&lt;/sup&gt;th strong entity type in ξ</td>
</tr>
<tr>
<td>WE&lt;sub&gt;&lt;i&gt;&lt;/i&gt;&lt;/sub&gt;</td>
<td>&lt;sup&gt;i&lt;/sup&gt;th weak entity type in ξ</td>
</tr>
<tr>
<td>R&lt;sub&gt;&lt;i&gt;&lt;/i&gt;&lt;/sub&gt;</td>
<td>&lt;sup&gt;i&lt;/sup&gt;th relationship type in ξ</td>
</tr>
<tr>
<td>Attr(E)</td>
<td>a set of all attributes of the entity type E</td>
</tr>
<tr>
<td>SimpAttr(E)</td>
<td>a set of all simple attributes of E</td>
</tr>
<tr>
<td>CompAttr(E)</td>
<td>A set of all composite attributes of E</td>
</tr>
<tr>
<td>MVA(E)</td>
<td>a set of all multivalued attributes of E</td>
</tr>
<tr>
<td>KeyAttr(E)</td>
<td>any key attribute of E</td>
</tr>
</tbody>
</table>
PartialKeyAttr(E) | a partial key attribute of weak entity type E
---|---
\(\zeta(R:E_j,E_k)\) | the cardinality of the relationship type R between entity types E_j and E_k
SE_j \(\Theta\) WE_i | SE_j is the owner entity type of the weak entity type WE_i
Attr(S) | a set of attributes of relation S
PK(S) | the primary key of relation S
S ← E | relation S corresponds to the entity type E

**Algorithm 5.1: ER-to-Relational Schema Mapping**

Let \(\xi = \{SE_1, SE_2, ..., SE_m, WE_1, WE_2, ..., WE_n, R_1, R_2, ..., R_p\}\)

1. For every \(E_i \in \{SE_1, SE_2, ..., SE_m\}\), create a relation \(S_i\) such that:
   
   \[
   \text{Attr}(S_i) = \text{SimpAttr}(E_i), \text{ and} \\
   \text{PK}(S_i) = \text{KeyAttr}(E_i)
   \]

2. For every \(E_i \in \{WE_1, WE_2, ..., WE_n\}\), create a relation \(S_i\) such that:
   
   \[
   \text{Attr}(S_i) = \text{SimpAttr}(E_i) \cup \text{PK}(T) \quad \text{where } T \leftarrow SE_j \text{ and } SE_j \Theta E_i
   \]
   
   \[
   \text{PK}(S_i) = \text{PartialKeyAttr}(E_i) \cup \text{PK}(T)
   \]

3. For every \(R_i \in \{R_1, R_2, ..., R_p\}\),
   
   a) if \(\zeta(R_i:E_j,E_k) = 1:1\) then
      
      \[
      \text{Attr}(S_j \leftarrow E_j) = \text{Attr}(S_i) \cup \text{PK}(S_k \leftarrow E_k)
      \]
   
   b) if \(\zeta(R_i:E_j,E_k) = 1:M\) then
      
      \[
      \text{Attr}(S_k \leftarrow E_k) = \text{Attr}(S_j) \cup \text{PK}(S_j \leftarrow E_j)
      \]
   
   c) if \(\zeta(R_i:E_j,E_k) = M:N\) then create a relation \(S\) such that:
\[ \text{Attr}(S) = \text{Attr}(R_i) \cup \text{PK}(S_j \leftarrow E_j) \cup \text{PK}(S_k \leftarrow E_k), \text{ and} \]
\[ \text{PK}(S) = \text{PK}(S_j) \cup \text{PK}(S_k) \]

4. For every multivalued attribute \( A_i \in \text{MVA}(E_k) \) where \( E_k \in \{SE_1, SE_2, \ldots, SE_m, WE_1, WE_2, \ldots, WE_n\} \), create a relation \( S_i \) such that:
\[ \text{Attr}(S_i) = A_i \cup \text{PK}(S_k \leftarrow E_k), \text{ and} \]
\[ \text{PK}(S_i) = A_i \cup \text{PK}(S_k \leftarrow E_k) \]

In the following sections, we analyze violations of various normal forms one by one.

### 5.3 Violation of First Normal Form (1NF)

1NF disallows relations having a set of values or a tuple as an attribute value for a single tuple. With the introduction of concept of composite attribute [Elmasri et al. 1985], violation of 1NF is no longer possible because the mapping rules always translate a composite attribute into its simpler attributes. These simpler attributes, by definition, always have atomic values.

### 5.4 Violation of Second Normal Form (2NF)

Consider a relation schema:

\[ R(A_1, A_2, \ldots, A_k, A_{k+1}, \ldots, A_n) \]

with its primary key \( K = \{A_1, A_2, \ldots, A_k\} \) where \( k < n \), and

a set of non-prime attributes \( NK = \{A_{k+1}, \ldots, A_n\} \).
2NF is violated only when there exists a partial dependency due to the presence of at least one FD such that:

\[ X \rightarrow Y \text{ where } X \subseteq K \text{ and } Y \subseteq NK. \]

Say, R is generated by mapping an entity type E. As we show in the following example, in every such case

i) the real world constraint given by this FD is not represented in the ER model which actually causes violation, and

ii) corresponding to such FD there should exist a logical entity type E' which is not defined in the original ER model.

According to the given FD, all possible values of set of attribute(s) X define unique values of set of attribute(s) Y and so X becomes a key attribute of E'. Since E' will construct a logical relationship with E, sets of attributes X and Y should be moved on to E'. As X being the set of prime attributes of entity type E has been moved on to the entity type E', E can no longer be a strong entity type. Therefore, E is now represented as a weak entity type of E' having Z as its partial key where Z = K - X. In this improved version, we have better conceptual design and relation schemas that satisfy 2NF. This can be summarized in Rule 5.1 as given below.
**Rule 5.1:**

For every FD $X \rightarrow Y$:

where $X \subseteq \text{KeyAttr}(E)$, and $Y \subseteq \text{Attr}(E) - \text{KeyAttr}(E)$

i) create an entity type $E'$ such that:

$\text{Attr}(E') = X \cup Y$, and

$\text{KeyAttr}(E') = X$

ii) mark $E$ as a weak entity type of $E'$

iii) $\text{Attr}(E) = \text{Attr}(E) - [X \cup Y]$

This is interesting to note that it is the same rule as Rule 4.1 defined for schema transformation. To illustrate this concept, consider a relation schema:

PROJECT (Loc, P#, Dept, Cost)

generated from Fig. 5.1. This figure represents a situation where different projects are carried out at various locations. Each project is uniquely identified by a project number and its location, that is, project numbers can be repeated across locations. At every location, we have a department which controls all projects carried out at that location. We want to record every project cost, too.

![ERD Causing Violation of 2NF](image-url)
Now, the following FDs are noted:

\[ FD1: \text{P#}, \text{LOC} \rightarrow \text{Cost} \]

\[ FD2: \text{LOC} \rightarrow \text{Dept} \]

The relation PROJECT is not in 2NF due to FD2. This FD suggests that Fig. 5.1 should have a better representation as given in Fig. 5.2, using solution proposed above. It generates the following relations:

- PROJECT_LOCATION (LOC, Dept)
- PROJECT (LOC, P#, Cost)

where each relation now satisfies 2NF.

![Fig. 5.2: Modified ERD Eliminating Violation of 2NF](image)

### 5.5 Violation of Third Normal Form (3NF)

Consider a relation schema:

\[ R(A_1, A_2, \ldots, A_k, A_{k+1}, \ldots, A_n) \]
with its primary key $K = \{A_1, A_2, \ldots, A_k\}$ where $k < n$ and a set of non-prime attributes $NK = \{A_{k+1}, \ldots, A_n\}$.

3NF is violated only when there exists a transitive dependency due to the presence of at least one FD such that:

$$X \rightarrow Y \text{ where } X, Y \subseteq NK.$$

Say, $R$ is generated by mapping an entity type $E$. As we show in the following example, in every such case,

i) the real world constraint represented by this FD is not represented in the ER model which actually causes violation, and

ii) corresponding to such FD there should exist a logical entity type $E'$ which is not defined in the original ER model.

According to the given FD, all possible values of set of attribute(s) $X$ define unique values of set of attribute(s) $Y$ and so $X$ becomes a key attribute of $E'$. Since $E'$ will construct a logical relationship with $E$, sets of attributes $X$ and $Y$ should be moved on to $E'$. With this improved model, we have achieved better conceptual design and at the same time its mapping generates relation schemas that satisfy 3NF. It is to be noted that, in this case, $X$ is not a set of prime attributes of the entity type $E$ and that is why $E$ still remains a strong entity type, as opposed to the case discussed in section 5.4 where it becomes a weak entity type. This can be summarized in Rule 5.2 as given below.
**Rule 5.2:**

For every $FD \ X \rightarrow Y$:

where $X, Y \subseteq Attr(E) - KeyAttr(E)$

i) create an entity type $E'$ such that:

$Attr(E') = X \cup Y$, and

$KeyAttr(E') = X$

ii) create a relationship type $R$ between $E$ and $E'$

iii) $Attr(E) = Attr(E) - [X \cup Y]$

This is interesting to note that it is the same rule as Rule 4.3 defined for schema transformation. To illustrate this concept, consider a relation schema:

EMP (ID, Name, Dept#, DeptName)

generated from Fig. 5.3.

Fig. 5.3: ERD Causing Violation of 3NF

Now, the following FDs are noted:

*FD1*: ID $\rightarrow$ Name, Dept#, DeptName

*FD2*: Dept# $\rightarrow$ DeptName
The relation EMP is not in 3NF due to $FD_2$. This FD suggests that Fig. 5.3 should have a better representation as given in Fig. 5.4, using solution proposed above. This figure generates the following relations:

EMP (ID, Name)

DEPT (Dept#, DeptName)

where each relation now satisfies 3NF.

Fig. 5.4: Modified ERD Eliminating Violation of 3NF

5.6 Violation of Boyce-Codd Normal Form (BCNF)

In previous sections, we have analyzed un-normalized relations up to 3NF and suggested solutions to overcome these normalization problems. We assume that by this stage we have obtained a relational schema in which every relation satisfies 3NF using our proposed transformations. In order to analyze the case that a relation schema does not
satisfy BCNF, Theorem 5.1 is presented based upon the work done by [Date and Fagin 1992] and it is proved that violation of BCNF is not possible if key of the relation is not composite.

Theorem 5.1: A relation schema R(A₁, A₂, ..., Aₙ) with primary key K = {A₁} as a singleton set and a set of non-prime attributes NK = {A₂, ..., Aₙ} cannot violate BCNF if R already satisfies 1NF, 2NF and 3NF.

Proof: We prove this theorem by negation. Let there be a relation schema R which satisfies 3NF and there exist a FD X → Y due to which R violates BCNF. For the left hand side of this FD (called determinant) there are following four possibilities that can exist:

i) X = K

ii) X ⊂ K

iii) X ⊃ K

iv) X ⊄ K and K ⊄ X

These cases are discussed below.

Case (i):

Since determinant is a complete key, BCNF is satisfied negating our hypothesis.

Case (ii):
If determinant is only a part of the primary key, then 2NF is violated which negates our hypothesis.

**Case (iii):**

This case implies that the FD is a trivial functional dependency always reducible to $K \rightarrow Y$ and the rest of the proof is the same as given in case i).

**Case (iv):**

This case is discussed under two possibilities, when:

I. $X \cap K \neq \emptyset$ or

II. $X \cap K = \emptyset$

For case (ivI), let $X \cap K = Z$ and $c \in Z$. Now $c \in K$. Since $K$ is a singleton set, $K \subseteq X$ which negates the condition in case (iv). Case (ivII) implies that $X$ is a non-prime attribute. $Y$ can now have the same four possibilities:

a. $Y = K$

b. $Y \subset K$

c. $Y \supset K$

d. $Y \not\subset K$ and $K \not\subset Y$

which are discussed one by one.

**Case (ivIIa):**
X → Y and Y = K imply X → K which implies that X is a candidate key. Hence BCNF is satisfied and hypothesis is false.

*Case (ivIib):*

This case is impossible because K is a singleton set and cannot have a proper subset.

*Case (ivIic):*

X → Y and Y ⊃ K imply X → K which implies that X is a candidate key. Hence BCNF is satisfied and hypothesis is false.

*Case (ivIId):*

As discussed in *case (iv)*, two possibilities may exist: Y ∩ K ≠ ∅ or Y ∩ K = ∅. The first possibility violates K ⊆ Y whereas the second possibility X ∩ K = ∅ and Y ∩ K = ∅ imply X,Y ⊂ NK. Then, X → Y violates 3NF which makes hypothesis false.

Hence, violation of BCNF is not possible under the given conditions.

This theorem has an important implication that our proposed approach of schema transformations is quite effective as every relation in the relational schema satisfies BCNF if the case of composite keys is rest aside.

When primary key is composite, before we can analyze cause(s) of the “defects”, we need to study the inherent characteristics of the relations which do not satisfy BCNF. The
next section investigates and proves formally such characteristics presented as *Lemmas 5.1 to 5.3* and *Theorem 5.2*. These characteristics can then be used in devising solution to the identified problem.
5.7 Composite Keys, BCNF and Dependency Preservation

Though work on normal forms of relational schemas seemed to reach an end, work in design theory continued to some extent as found in [Paredaens et al. 1989, Mannila and Raiha 1992, Thalheim 1994, Makowsky and Ravve 1998]. It was observed that the relations which are in 3NF and not in BCNF exhibit interesting properties. These properties can be found in well known text books [Ullman 1990, Elmasri and Navathe 2004] and research [Date and Fagin 1992, Vincent and Srinivasan 1993]. In [Date and Fagin 1992], it was proved that if a relation is in 3NF but not in BCNF then the relation must contain at least one composite key. In another paper, Vincent derived a stronger result that if a relation is in 3NF but not in BCNF then the relation must have at least two candidate keys which overlap [Vincent and Srinivasan 1993]. However, their formal proof did not explicitly cover that the two candidate keys are composite. Therefore, we give a formal proof of the above mentioned statement along with the proofs of certain other properties that we observe for the 3NF-but-not-BCNF case.

In the literature, it was also observed that decomposing a given schema to another one which satisfies higher normal forms is not a sufficient condition for a good design. The decomposition should be further reinforced by additional properties like lossless join and dependency preservation [Elmasri and Navathe 2004]. This requirement becomes debatable when we observe relations in 3NF converted to BCNF losing dependency preservation. Though Makowsky has recently discussed a splitting attribute technique
[Makowsky and Ravve 1998] which preserves dependencies, splitting attribute is not always possible. We, in this chapter, clearly state and prove that whenever a 3NF-but-not-BCNF relation is converted to BCNF, some dependency will be lost.

### 5.7.1 At Least Two Composite Keys

Vincent proved in [Vincent and Srinivasan 1993] that a relation R which is in 3NF but not in BCNF has at least two overlapping candidate keys. We now prove a stronger result in Lemma 5.1 that both of these candidate keys have to be composite.

**Lemma 5.1:** If R is a relation scheme which is in 3NF but not in BCNF then the two overlapping candidate keys are composite.

**Proof:** Let K and K’ be the two candidate keys of relation R which overlap. This implies:

\[ K \neq K’ \]  
(5.1)

and

\[ K \cap K’ = Z \neq \emptyset \]  
(5.2)

which, in turn, implies:

\[ Z \subseteq K \]  
(5.3)

and

\[ Z \subseteq K’ \]  
(5.4)

For sake of argument, say, K’ is not composite. Then, from (5.2) and (5.4):

\[ Z = K’ \]  
(5.5)

From (5.3) and (5.5),
5.7.2 Functional Dependency Violating BCNF

**Lemma 5.2:** If R is a relation scheme which is in 3NF but not in BCNF due to a non-trivial functional dependency $X \rightarrow Y$ then:

a) $Y$ is not a candidate key of the relation R, and

b) $X \cap Y = \emptyset$

**Proof:** For a), by negation, let:

Y be the candidate key of R. \hspace{1cm} (5.7)

Now, R satisfying 3NF and not BCNF implies that:

i) $X$ is not a superkey, and

ii) $Y$ is a prime attribute.

Let K be the candidate key of R for which:

$Y \subseteq K$ \hspace{1cm} (5.8)

From (5.7) & (5.8), $Y = K$.

Then $X \rightarrow K$ because $X \rightarrow Y$. It implies $X$ is a superkey which contradicts our stated implication. Hence the proof.

For b), again by negation, let:

$X \cap Y = Z \neq \emptyset$ \hspace{1cm} (5.9)
which, in turn, implies:

\[ Z \subseteq X \quad (5.10) \]

and

\[ Z \subseteq Y \quad (5.11) \]

Since \( Y \) is a singleton set (being R.H.S. of the FD), \( Z = Y \) from (5.9) and (5.11). Using (5.10) then:

\[ Y \subseteq X \quad (5.12) \]

which makes the FD: \( X \rightarrow Y \) trivial contradicting the given in the Lemma.

### 5.7.3 Candidate Key Contains Determinant of the Violating Functional Dependency

**Lemma 5.3:** If \( R \) is a relation scheme which is in 3NF but not in BCNF due to a non-trivial functional dependency \( X \rightarrow Y \) then there exist a candidate key containing \( X \) for the relation \( R \).

**Proof:** Let \( K \) be the candidate key for which:

\[ Y \subseteq K \quad (5.13) \]

Using Lemma 5.2a, \( Y \neq K \). So,

\[ Y \subset K \quad (5.14) \]

Let:

\[ K-Y = M \quad (5.15) \]
Since $X \rightarrow Y$, $(X \cup M) \rightarrow K$ which implies $(X \cup M)$ is a superkey. In order to prove that $(X \cup M)$ is minimal, let us negate it by saying that there exists an attribute $A$ which can be reduced from $X$ such that $(X-A) \cup M$ is still a superkey.

Since $K-Y$ can’t be a key if $K$ is the key and by (5.15), $M \cap Y = \emptyset$, we imply:

$$M \rightarrow Y \quad (5.16)$$

Also,

$$(X-A) \rightarrow Y \quad (5.17)$$

because $X \rightarrow Y$ is a non-trivial FD. Hence:

$$(X-A) \cup M \rightarrow Y \quad (5.18)$$

or $(X-A) \cup M$ cannot be a superkey. Hence the proof.

### 5.7.4 Dependency Preservation and BCNF

**Theorem 5.2:** If $R$ is a relation scheme which is in 3NF but not in BCNF then there does not exist a dependency preservation decomposition of $R$ in which every relation satisfies BCNF.

**Proof:** Let by Lemma 5.1, $K$ be a composite candidate key of the relation $R$. Also, let $X \rightarrow Y$ (FDJ) be the non-trivial FD due to which $R$ is not in BCNF. Then by Theorem 3 in [1]:

$$X \text{ cannot be a superkey} \quad (5.19)$$

and by Lemma 5.2,
For any $A \subseteq R$, let $K \rightarrow A$ \textit{(FD2)} be another non-trivial FD in $F_{\text{min}}$, that is, the minimal cover of the set of functional dependencies related to the problem domain. A decomposition $D$ of $R$ which has every relation in BCNF can be written as:

\[ D = (R1, R2) \]

where

\[ R1 = R - Y, \text{ and} \]

\[ R2 = X \cup Y \]

We now prove that $FD1$ is lost in $D$, that is, it exists neither in $R1$ nor in $R2$. For this purpose, it is sufficient to prove that $K$ of $FD1$ is present neither in $R1$ nor in $R2$.

For $R1$, let $y \in Y$, then using (1), $y \in K$.

But, $y \notin (R-Y)$. Hence:

\[ K \not\subseteq (R-Y) \] \hspace{1cm} (5.21)

For $R2$, let us negate by saying:

\[ K \subseteq (X \cup Y) \] \hspace{1cm} (5.22)

It implies:

\[ (K-Y) \subseteq X \Rightarrow X \rightarrow (K-Y) \Rightarrow X \rightarrow K \] \hspace{1cm} (5.23)

which means $X$ is a superkey contradicting (5.19).

Hence the proof.
5.8 Summary

In this chapter, we have elaborated that a given ERD translated into a relational schema may not satisfy certain normal forms. We have analyzed reasons which cause violations of normal forms and have found that such violations occur due to improper or insufficient representation of real world constraints in the corresponding ER model.

In particular, we have shown that:

a. violation of 1NF is not possible if composite attributes (if any) are represented,

b. violation of 2NF and 3NF occurs when some functional dependencies are ignored or improperly represented in the ERD. A set of rules has been proposed which states how these dependencies should be represented by adding weak entity types and regular entity types,

c. applying the proposed rules always generates relation schemas that do not violate the respective normal forms,

d. the proposed rules are quite similar to the schema transformation rules – Rule 4.1 and Rule 4.3 proposed in chapter 4 for semantic improvement. Therefore, these rules get validated using a reverse-engineering approach presented in this chapter.

Using formal methods, we have also analyzed the case when primary key is composite. We observed and proved the following properties for 3NF-but-not-BCNF case:

a) there have to be at least two candidate keys which overlap and which are composite

b) for a functional dependency due to which the relation is not in BCNF,
i) the right-hand side cannot be a candidate key, and

ii) no attribute is common between the attributes of the two sides

iii) there must exist a candidate key containing the determinant (that is, left-hand side) of the functional dependency

c) when the relation is decomposed into a set of relations such that every relation satisfies BCNF, at least one dependency is lost. In other words, there cannot exist a dependency preservation decomposition of a relation which is in 3NF but not in BCNF.

These findings provide us the insight that violation of BCNF can occur when key is composite. However, any solution that converts it to BCNF satisfying schema loses some functional dependencies. Hence, for this particular case, we need not present a solution as it cannot be better than what we otherwise achieve through normalization.

An important implication of the work presented in this chapter is that a relational schema need not go through the normalization process at least till BCNF if a real world problem and its constraints are properly represented in the ER model. This can save database designer’s time significantly meeting the objective of our research laid out in section 2.5.
Chapter 6

Improving Semantic Representation

6.1 Introduction

We have defined, in section 1.3, a three-dimension strategy for our research regarding improving conceptual models – Improve ER schema, Improve semantic representation and Measure quality. We have addressed the first dimension in chapters 4 and 5 proposing schema transformation rules that can improve an ER schema while studying the impact of suggested rules on the normalization of resulting relational schema. In this chapter, we are addressing the second dimension, namely, improve semantic representation. Along this dimension, instead of improving semantics through some kind of schema transformations, we intend to identify the missing constructs in the existing ER model. We, in this chapter, elaborate that relationship attributes do not have a clear semantic representation in the ER model which creates modeling ambiguity as well as un-normalized relations. Therefore, we formally propose concepts of Single-Valued Relationship Attributes (SVRA) and Multi-Valued Relationship Attributes (MVRA). We also differentiate amongst real-world situations which require a single MVRA, multiple MVRAs, or SVRA and MVRA together while modeling. We also propose a diagrammatic notation that can be used in the ER model corresponding to these concepts. Finally, we propose an algorithm which provides an ER-to-relational mapping and
establishes that the resulting relations satisfy those normal forms which are otherwise violated in the absence of our proposed constructs.

### 6.2 Problem with Relationship Attributes

In order to describe the semantic ambiguity in the existing ER model, we consider *Scenario 6.1*, as given below, for a small part of a sales system. In this system, CUSTOMER and PRODUCT are identified as entity types having appropriate attributes and PURCHASES is identified as a relationship type between these two entity types.

*Scenario 6.1:* Consider a situation where a customer may purchase a number of products and a product can be purchased by a number of customers. We are interested in keeping track of the date when a customer purchases a product.

This situation is represented in Fig. 6.1 using ER model notation. In order to keep track of the date when a customer purchases a product, ‘Date’ is marked as an attribute on the relationship type ‘Purchases’. This model can be viewed in terms of a semantic net model as shown in Fig. 6.2, where for example the relationship instance r1 relating customer (id) 29 to the product (code) P-101 has a value 14-Jan.-2003 for its ‘Date’ attribute.

![Fig. 6.1: A Relationship Attribute](image-url)
When this part of the ER model is mapped to a relational database schema following the
*Algorithm 5.1*, the following relations are created:

CUSTOMER (ID, Name)

PRODUCT (Code, Desc)

PURCHASES (CustID\textsubscript{f.k}, ProdCode\textsubscript{f.k}, Date)

The underlined attribute represents a primary key of the relation whereas a foreign key is represented by an attribute with subscript \textsubscript{f.k}. For relations CUSTOMER and PRODUCT, ID and Code are primary keys respectively, whereas, for the relation PURCHASES created corresponding to M:N relationship type ‘Purchases’, primary keys of relations corresponding to participating entity types namely CUSTOMER and PRODUCT become foreign keys and together form the primary key of relation PURCHASES.

This model and its mapping works fine as long as a customer purchases a product only on a single date, that is, only a single date is defined for each relationship instance of Fig. 6.2. In terms of functional dependency, this constraint can be written as:

\[ \text{CustID, ProdCode} \rightarrow \text{Date} \]

But this is an unrealistic constraint for most of the real world situations where a customer is not bound to purchase a product only once. This leads us to define a concept for a relationship attribute, as given in the following section, which can have more than one value.
6.3 Multi-Valued Relationship Attributes (MVRA)

Let us define Scenario 6.2 where we present a relationship attribute having more than one value.

**Scenario 6.2:** A customer purchases the same product on different dates and we want to keep track of all such purchases

This scenario is represented in Fig. 6.3 where the relationship instance r1 has two values for the attribute ‘Date’: 14-Jan.-2003 and 12-May-2003 against two purchases of the same product by the same customer. This requires a relationship attribute which can have more than one value (defined as multi-valued relationship attribute in Definition 6.1) to be differentiated from the one which may have at most one value for a relationship instance (defined as single-valued relationship attribute in Definition 6.2).
Definition 6.1: A Multi-Valued Relationship Attribute (MVRA) is a relationship attribute which may have more than one value for a relationship instance of the relationship set.

Definition 6.2: A Single-Valued Relationship Attribute (SVRA) is a relationship attribute which cannot have more than one value for a relationship instance of the relationship set.

A relationship attribute (single-valued or multi-valued) can now be defined mathematically as:

Definition 6.3: An attribute $A$ of relationship type $R$ whose value set is $V$ is a function from $R$ to the power set $P(V)$ of $V$:

$$A : R \rightarrow P(V)$$

This definition covers single-valued and multi-valued relationship attributes, as well as nulls. A null value is represented by the empty set. For single-valued relationship attributes, $A(r)$ is always a singleton for each relationship instance $r$ of the set $R$; whereas there is no such restriction for a MVRA. Here $A(r)$ refers to the value of attribute $A$ for relationship instance $r$. 
In order to represent MVRA in an ER diagram, a new notation is proposed. This notation is writing the name of the attribute in the set notation i.e. braces within the oval (symbol for attribute) corresponding to the idea that this attribute may have a set of values. Now ‘Date’ attribute in Fig. 6.1 changes to ‘\{Date\}’ in Fig. 6.4 for the revised situation.

Consequently Rule (3c) in the ER-to-relational mapping algorithm, Algorithm 5.1, should also be modified to take care of this situation because otherwise there will be multiple tuples in the relation PURCHASES with identical values of ‘CustID’ and ‘ProdCode’ violating the primary key constraint for this relation. This modification (for mapping M:N relationship type) is presented in Algorithm 6.1 below:

Algorithm 6.1: Mapping M:N relationship type to a relational schema

For every binary M:N relationship type $R$ between entity types $E_1$ and $E_2$, having a set of multi-valued relationship attributes $\{MVRA\}$,

1. create a new relation $S$ to represent $R$ such that:

$$Attr(S) = \{PK(E_1)\} \cup \{PK(E_2)\} \cup Attr(R) \cup \{MVRA\}$$

where $PK(E_i)$ is primary key of the relation created for entity type $E_i$, and
\( \text{Attr}(R) \) is the set of simple attributes (or simple components of composite attributes) of \( R \)

\[ \{\text{PK}(S)\} = \{\text{PK}(E_1)\} \cup \{\text{PK}(E_2)\} \cup \{\text{MVRA}\} \]

It should be noted that \( \text{PK}(E_1) \) and \( \text{PK}(E_2) \) are foreign keys in relation \( S \). Step 2 of this algorithm suggests that \( \text{MVRA} \) should also be marked as part of the primary key of relation \( S \) along with \( \text{PK}(E_1) \) and \( \text{PK}(E_2) \). Applying this algorithm, we get the following relation in our schema:

**PURCHASES** (\( \text{CustID}_{f,k} \), \( \text{ProdCode}_{f,k} \), \( \text{Date} \))

In this context, it is important to note the following observations:

- The concept of a multi-valued relationship attribute is primarily different from:
  - nested attributes (tuple constructors or set constructors) defined in [Thalheim 2000] because nested attributes combine attributes together and do not describe multiple values of a single attribute, as described in section 2.3.6.2,
  - composite attributes (like Thalheim’s nested attributes) defined in [Elmasri et al. 1985], because composite attributes also combine attributes together and do not describe multiple values of a single attribute, as described in section 2.3.5,
  - multi-valued attributes defined in [Elmasri et al. 1985] because multi-valued attributes are defined for entity types and not for relationship types, as given in section 2.3.5.
One can think of modeling scenario 6.2 by introducing a weak entity type PURCHASE and relating it to CUSTOMER and PRODUCT. It raises the question which entity type is then owner of PURCHASE – CUSTOMER or PRODUCT? None of these two entity types can be the owner because an instance of PURCHASE cannot be uniquely identified with any of the two entity types. Hence, this solution is not correct. In section 6.3.2, we shall also analyze the possibility of suggesting another alternative in which PURCHASE can be defined a weak entity type with respect to both of these entity types.

The new concept of MVRA helps resolve elegantly the conceptual differences in many of the problems as illustrated in the following sections.

### 6.3.1 Multiple MVRAs

Now, we present another scenario which requires more than one MVRA.

**Scenario 6.3**: Apart from the date, we want to keep track of the quantity of a product, as well, purchased in every instance.

In this scenario, the relationship type PURCHASES will have two attributes ‘Date’ and ‘Quantity’. The question then arises: Is each of these two attributes a MVRA? According to the definition of MVRA given above, the answer is, of course, yes; because each of these attributes may have multiple values for a single relationship instance present in the relationship set. This implies that, as per *Algorithm 6.1*, the relation
PURCHASES will have attributes ‘Date’ and ‘Quantity’ both as a part of the primary key.

\[
\text{PURCHASES (CustID}_{f.k.}, \text{ProdCode}_{f.k.}, \text{Date}, \text{Quantity})
\]

### 6.3.2 MVRA and SVRA

A scenario which requires both SVRA and MVRA can also be discussed.

**Scenario 6.4:** We have an additional constraint that a customer purchases a particular product always in the same quantity.

The attribute ‘Quantity’, in this case, is no more a MVRA but it is a SVRA; because it always has only a single value for each relationship instance. The ER model for this situation is given in Fig. 6.5

![Fig. 6.5: A SVRA and MVRA Together](image)

Applying *Algorithm 6.1* on Fig. 6.5 for its transformation to a relational schema, a relation PURCHASES is created having attributes ‘CustID’ and ‘ProdCode’ as foreign keys, and the attributes ‘Date’ and ‘Quantity’. Since ‘Date’ is a MVRA, the primary key for this relation comprises of \text{CustID\_ProdCode\_and\_Date}. This
solution, however, violates second normal form (2NF), in this case, due to the existence of the following functional dependency:

\[ \text{CustID, ProdCode} \rightarrow \text{Quantity} \]

This requires further refinement of the mapping algorithm (Algorithm 6.1) which is then given below as Algorithm 6.2:

**Algorithm 6.2: Mapping M:N relationship type generating a normalized relational database schema**

For every binary M:N relationship type \( R \) between entity types \( E_1 \) and \( E_2 \),

1. create a new relation \( S \) to represent \( R \) such that:
   \[
   \text{Attr}(S) = \{\text{PK}(E_1)\} \cup \{\text{PK}(E_2)\} \cup \{\text{SVRA}\}
   \]
   where \( \text{PK}(E_i) \) is primary key of the relation created for entity type \( E_i \), and
   \( \{\text{SVRA}\} \) is the set of all single-valued relationship attributes of \( R \)

2. \( \{\text{PK}(S)\} = \{\text{PK}(E_1)\} \cup \{\text{PK}(E_2)\} \)

3. if there exists a MVRA of \( R \), then:
   a. create a new relation \( T \) such that:
      \[
      \text{Attr}(T) = \{\text{PK}(E_1)\} \cup \{\text{PK}(E_2)\} \cup \{\text{MVRA}\}
      \]
      \[
      \{\text{PK}(T)\} = \{\text{PK}(E_1)\} \cup \{\text{PK}(E_2)\} \cup \{\text{MVRA}\}
      \]
      where \( \{\text{MVRA}\} \) is the set of all multi-valued relationship attributes of \( R \).
Applying this algorithm for relationship type PURCHASES, we get the following relations:

PURCHASES1 (CustID, ProdCode, Quantity)

and

PURCHASES2 (CustID, ProdCode, Date)

Each of these relations now satisfies 2NF and, if no other functional dependency violation occurs, 3NF and BCNF are also satisfied.

Before we conclude this chapter, it is interesting to analyze one possible alternative that may be suggested to model scenario 6.4. This alternative is presented in Fig. 6.6 and does not use the concept of SVRA or MVRA proposed in this chapter.

Fig. 6.6: A Possible Alternative for SVRA and MVRA
Comparing this ERD with the one using our proposed constructs shown in Fig. 6.5, we observe that it is overly complex. [Thalheim 2000] has a detailed discussion on misuse and complexity created due to such weak entity types thereby making the conceptual model less explicable. Further, this solution requires careful analysis. Obviously, the solution proposed in Fig. 6.6 does not use the SVRA and the MVRA concept proposed in this chapter. This implies that it carries the same semantics of relationship attributes which are prevailing so far. The existing semantics permit the relationship attribute to take on any number of values against a relationship instance. In other words, Fig. 6.6 does not enforce the FD: ID, Code → Qty; whereas the functional dependency is a valid FD under the given scenario. Hence the solution is not correct.

6.4 Summary

In this chapter, we have highlighted a deficiency of the ER model in representing correct semantics for relationship types which have their own attributes. We have proved that mapping such relationships to a relational database schema generates relations having primary keys which cannot guarantee unique tuples for real world data; thus violating the definition of a primary key. In addition, these relations may not satisfy second normal form.

We have thus proposed the concept of multi-valued relationship attribute and have demonstrated with various examples that the new concept of MVRA nicely resolved the semantic and normalization problems. For this new concept, an ER diagram notation and a mapping algorithm for its transformation to relational schema have also been proposed.
It has been demonstrated that the relations created using this algorithm satisfied the normal forms which were otherwise violated in the absence of our proposed construct.

With this, we have concluded the study of first two dimensions of our proposed strategy – improving ER schema and improving semantic representation – and part II of this thesis. In the next part, part III, we shall focus our attention on the quality aspects of a conceptual model and the effectiveness of all those rules which have been proposed in part II on the quality of the ER model.
Part III

Improving Conceptual Model - A Quality Perspective
Chapter 7

Defining Quality Metrics

7.1 Introduction

In Part II of the thesis, we have studied in detail a methodology to improve a conceptual model. This methodology has proposed primarily a set of schema transformation rules and new constructs to semantically improve the ER model. We have also studied the impact of our proposed methodology on the “quality” of relational schema, defined from the corresponding ER model, in the context of normalization theory. However, we are still unable to compare two ER models for the same problem to evaluate which of the two models is of better quality. This comparison cannot be made unless we understand the quality concept of an ER model in quantitative terms. In chapter 3, we have studied various quality parameters but these are either not quantitatively defined or suffer from applicability in practice. Therefore, the need of defining quantitative and objective measures to assess the quality of an ER model cannot be over-emphasized.

In this part (Part III), we are interested in studying the quality aspect of a conceptual model in quantitative terms. The motivation is that we should be able to compare two models, for the same real-world problem considered semantically equivalent, in terms of
the quantitative measures and evaluate which of the two models is of better quality. This
will also enable us to further improve a model, if need be.

In chapter 3, we studied completeness as one of the most important parameters to measure
the quality of a conceptual model. Completeness was defined as a characteristic of a
conceptual model or an ER model to contain all true statements and aspects of the
However, no quantitative measure for completeness of an ER model is found in the
literature. We have viewed, in previous chapters, functional dependencies as important
constraints of a conceptual model. This encourages us to introduce functional
dependencies as relevant requirements or statements which should be “completely”
present in a conceptual model. This view supports definitions of completeness by various
researchers given in chapter 3.

In this chapter, we define a set of easy-to-apply quantitative metrics to measure the
quality of an ER model in terms of completeness, in terms of the “quality” of resulting
relational schema and in terms of the overall quality. Therefore, we propose three quality
metrics, namely, Completeness Index, Normalization Index and Overall Quality Index.
The application of these metrics is demonstrated with the help of a scenario. Based on
these quality metrics, we also validate the improvement in ER schemas using our
proposed schema transformation rules. Hence, the objective of this chapter is to elaborate:

1. computation of our proposed quality metrics for a given ER model,
2. effectiveness of our proposed schema transformations in improving the quality of a given ER model, and
3. usefulness of our proposed metrics to compare the quality of alternative ER models.

### 7.2 Quality Metrics

In this section, we suggest and define the following metrics to measure the quality of a conceptual model:

1. Completeness Index
2. Normalization Index
3. Overall Quality Index

#### 7.2.1 Completeness Index (CI)

As stated earlier, one criterion to measure completeness can be defined in terms of number of functional dependencies represented in an ER model. So, a quantitative measure for the semantic quality is proposed as a completeness index. Completeness Index can be defined as a ratio between number of functional dependencies represented in a given ERD to the total number of functional dependencies defined for that problem domain.

**Definition 7.1 – Completeness Index:** Completeness Index is defined as:
CI = \frac{n(f)}{n(\mathcal{I})} \quad (7.1)

where,

\mathcal{I} is a set of functional dependencies identified from a problem domain,

\xi is a given ERD,

f is the projection set of \mathcal{I} on \xi i.e. f = \pi_\mathcal{I}(\xi),

n(\mathcal{I}) = total number of functional dependencies in \mathcal{I}, and

n(f) = total number of functional dependencies in f.

This definition corresponds to Moody’s Metric 2 under completeness which was described in section 3.4.1. It can be noted that 0 \leq CI \leq 1.0 and that the higher the value of CI, the greater is the quality in terms of completeness.

### 7.2.2 Normalization Index (NI)

Another important aspect of the quality of a conceptual model can be defined in terms of the ease with which it can be translated to a logical model or in terms of the quality of the logical model it is translated to. Since relational model is dominating even today, we assume that an ERD will be transformed to a relational schema. In this relational schema, we know that every relation may satisfy a certain normal form [Codd 1972, Fagin 1977, Ullman 1990, Elmasri and Navathe 2004] which determines the quality of the relational schema. Thus, we can define a metric which measures the extent or degree to which each relation of the transformed schema is normalized. We call this metric as Normalization
Definition 7.2 – Normalization Index: Normalization Index is defined as:

\[
NI = \frac{\sum_{i=1}^{n} NV(R_i)}{n},
\]

(7.2)

where,

\[ R = \{R_i \mid \text{Every } R_i \text{ is a relation obtained by transforming a given ERD,} \]

and \( i = 1, 2, 3, ..., n \}, \)

\[ w_j = \text{weight arbitrarily assigned to } j^{th} \text{ normal form,} \]

where \( w_k > w_l \forall k > l, \text{ that is,} \)

\[ w_{BCNF} > w_{2NF} > w_{2NF} > w_{1NF} \]

and

\[ NV(R_i) = \text{Normalization Value of Relation } R_i = w_j \]

if \( R_i \) is in \( j^{th} \) normal form.

It can be noted that \( 0 \leq NI \leq 1.0 \) if weights are assigned in such a manner that:

\[ w_{BCNF} = 1, \]

\[ w_{1NF} > 0, \text{ and} \]

\[ NV(R_i) = 0 \text{ if } R_i \text{ does not satisfy 1NF.} \]
Further, the higher the value of NI, the greater is the quality in terms of normalization. Based upon a number of quality metrics defined, comparing two models may not be easy. Therefore, we define an overall quality index which can be used for this purpose.

### 7.2.3 Overall Quality Index (QI)

It can be defined as a weighted average of all the quality metrics used for a conceptual model.

**Definition 7.3 – Overall Quality Index:** Overall Quality Index is defined as:

\[
QI = \frac{\sum_{i=1}^{n} (w_i M_i)}{\sum_{i=1}^{n} w_i} ,
\]

(7.3)

where,

- \(M_i\) represent value of \(i^{th}\) metric, and
- \(w_i\) represent the respective weight assigned to \(M_i\)

Considering it for CI and NI only, it reduces to:

\[
QI = \frac{w_{CI} (CI) + w_{NI} (NI)}{(w_{CI} + w_{NI})}
\]

(7.4)

where \(w_{CI}\) and \(w_{NI}\) represent weights assigned to Completeness Index and Normalization Index, respectively, in order to determine the overall quality. It can be observed that \(0 \leq QI \leq 1.0\) if \(0 \leq NI \leq 1.0\).
Now, in the next section, we present a small case study for which the conceptual model is gradually improved using transformation rules. In addition, the above mentioned quality indexes are evaluated and compared for each such transformation.

### 7.3 Evaluating Quality: A Case Study

Consider the following scenario described by a user:

“In our company, we want to keep track of the employees’ information regarding which project he is working in and which department he belongs to. An employee is uniquely identified by his ID and name. The department number he belongs to, the respective department name and department location are also stored. Also recorded for an employee are some other attributes like project number for the project he is presently working on, its location, cost and supervisor. A supervisor is assigned the responsibility for each location a project is carried out. Our company may assign same project numbers to two different projects but then those should be located at two different locations. An employee belongs to one department only and is not allowed to work on more than one project.”
Now, for this description, we may get an ERD given in Fig. 7.1. In order to compute $CI$ for Fig. 7.1, let us first identify $\mathcal{F}$ as consisting of the following set of FDs:

- $FD_1$: $P\#$, LOC $\rightarrow$ Cost
- $FD_2$: LOC $\rightarrow$ Super
- $FD_3$: ID $\rightarrow$ Name
- $FD_4$: ID $\rightarrow$ Dept#
- $FD_5$: Dept# $\rightarrow$ DName
- $FD_6$: Dept# $\rightarrow$ DLoc
- $FD_7$: ID $\rightarrow$ P#
- $FD_8$: ID $\rightarrow$ LOC
- $FD_9$: ID $\rightarrow$ Cost
- $FD_{10}$: ID $\rightarrow$ Super

It can be observed that $\mathcal{F}$ consists of only $FD_3$, $FD_4$ and $FD_7$ through $FD_{10}$. So, $CI = \frac{6}{10} = 0.60$

To calculate $NI$, we assume the following values:

- $w_{BCNF} = 1.0$
- $w_{3NF} = 0.9$
- $w_{2NF} = 0.6$
\[ w_{1NF} = 0.3 \]

It should be noted that there seems to be a subjectivity involved in assigning values to these weights but it is immaterial. Any weights can be assigned within the given definition (definition 7.2) because it does not affect the nature of comparative results for NI.

Applying the Algorithm 5.1 (for ER-to-Relational schema mapping), the ERD in Fig. 7.1 is transformed yielding the following relation:

\[ \text{EMP (ID, Name, Dept#, DName, DLoc, P#, Loc, Cost, Super)} \]

This relation does not satisfy 3NF due to violation of \( FD_2 \), at least. Hence, \( NI = 0.6 \). It is just a coincidence that for this particular ERD, \( CI = NI \); otherwise, no such relationship exists.

Likewise, for computing QI, any appropriate weights can be assigned, we assume \( w_{CI} = w_{NI} = 0.50 \). Based upon these weights, \( QI = 0.5(0.6) + 0.5(0.6) = 0.6 \) for this ERD.

Let us now consider schema transformations of ERD in Fig. 7.1 by applying the given rules. Starting with the Rule 4.2 first, considering \( FD_1 \) and \( FD_2 \), the ERD in Fig. 7.1 is transformed to a conceptual schema represented in Fig. 7.2. For this figure, we compute:
CI = 7/10 = 0.70 (f now consists of $FD_1$, $FD_3$, $FD_4$ and $FD_7 – FD_{10}$), which shows 17% improvement on completeness.

\[ \text{Fig. 7.2: Improved Conceptual Schema after First Transformation} \]

The relational schema obtained for the ERD in Fig. 7.2 is:

\[ \text{EMP (ID, Name, Dept#, DName, DLoc, P#, Loc)} \]

satisfying 2NF, and

\[ \text{PROJECT (LOC, P#, Cost, Super)} \]

satisfying 1NF.

Hence NI = (0.6 + 0.3)/2 = 0.45, which is less than the previous value (0.6) indicating that more transformations are required. QI = 0.5 (0.7) + 0.5(0.45) = 0.58 which is 3% deterioration on original QI due to decrease in the NI component.

Another transformation can take place by applying Rule 4.4 on the basis of $FD_5$ and $FD_6$ generating a conceptual schema represented in Fig. 7.3. For this figure, f now consists of
all functional dependencies except $FD_2$. Hence, $CI = 9/10 = 0.90$, which shows 50% improvement on completeness (based upon the original value of CI).

The relational schema obtained for the ERD in Fig. 7.3 is:

**EMP** ($ID$, Name, Dept#, P#, Loc)
satisfying BCNF,

**DEPT** (Dept#, DeptName, DLoc)
satisfying BCNF,

**PROJECT** (LOC, P#, Cost, Super)
satisfying 1NF.

![ERD Diagram]

**Fig. 7.3: Improved Conceptual Schema after Second Transformation**
Hence NI = (1.0 + 1.0 + 0.3) / 3 = 0.77, which shows 28% improvement on Normalization Index. QI = 0.5 (0.9) + 0.5(0.77) = 0.84 which is 40% improvement on original QI.

A final transformation can now take place by applying Rule 4.1 on the basis of FD2 generating a conceptual schema represented in Fig. 7.4. For this figure, we compute: CI = 10/10 = 1.0, which is 67% improvement on completeness.

![Fig. 7.4: Improved Conceptual Schema after Third Transformation](image)

The relational schema obtained for the ERD in Fig. 7.4 is:

\[
\text{EMP (ID, Name, Dept#, P#, Loc)}
\]
satisfying BCNF,

\[
\text{DEPT (Dept#, DeptName, DLoc)}
\]
satisfying BCNF,
PROJECT (LOC, P#, Cost) satisfying BCNF,

PROJECT_LOCATION (LOC, Super) satisfying BCNF.

Hence NI = 4 (1.0) / 4 = 1.0, which shows 67% improvement on Normalization Index.
QI = 0.5 (1.0) + 0.5(1.0) = 1.00 which is 67% improvement on original QI.

Computations of all these quality metrics are summarized in Table 7.1 for the three transformations for comparative purpose.

<table>
<thead>
<tr>
<th>Table 7.1 Summary of Quality Metrics for the Given Case Study</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Original</td>
</tr>
<tr>
<td>1st Transformation</td>
</tr>
<tr>
<td>2nd Transformation</td>
</tr>
<tr>
<td>3rd Transformation</td>
</tr>
</tbody>
</table>

*Increase is determined based upon the original value.*

We observe that quality in terms of CI and QI for the problem under discussion has gradually improved with schema transformations. We also observe a slight decrease in QI after 1st transformation which is due to deteriorated value of NI. This shows that the value
of NI (or change in NI due to a schema transformation) does not depend upon the value of CI (or change in CI due to a schema transformation). Since every schema transformation increases \( f = \pi_\mathcal{G}(\xi) \), CI is bound to improve with every schema transformation but this is not the case for NI. The effect of schema transformations on CI is discussed in detail in section 8.5.

7.4 Summary

In this chapter we have attempted to define quality of a conceptual model in quantitative terms. We have proposed three quality metrics, Completeness Index, Normalization Index and Overall Quality Index to measure the quality of a given conceptual model quantitatively. Application of these metrics to compare the quality of two conceptual models, which are otherwise considered equivalent, has also been illustrated. It has been demonstrated that computations for proposed metrics are fairly simple and therefore can be applied easily to any practical problem. Applying these metrics to a hypothetical case study presented in this chapter for illustration, it is also observed that the overall quality index improves up to 67% for this case when schema transformations are gradually applied in order to improve the semantic quality of the conceptual model. It has also been shown that completeness index gets improved with every schema transformation but it is not necessary for NI. This shows that CI and NI are independent of each other.
Chapter 8

Refining Completeness Index Using Fuzzy Approach

8.1 Introduction

In the previous chapter, completeness was described as an important measure for semantic quality of an ER model. We defined a metric completeness index (CI) based on functional dependencies identified in a problem domain as important features, statements or requirements of the problem which should be ‘completely’ present in a conceptual model. Previously, Completeness Index was defined as the ratio between number of functional dependencies represented in an ERD and total number of FDs identified in the problem domain, that is, CI = n(ℑ) / n(ℱ). The definition used a projection set f of ℱ on ξ defined as f = πℱ (ξ) representing the set of functional dependencies, where ℱ is the set of all functional dependencies identified from a problem domain, and ξ denotes the ERD.

The definition also uses a count operator ‘n’ such that n(ℱ) represents total number of functional dependencies in ℱ and n(f) represents total number of functional dependencies in f. Formally, n(f) = ∑ count (FD_i) where count(FD_i) ∈ {0,1}; 0 if i\textsuperscript{th} functional dependency in ℱ is not represented in the ERD; otherwise 1. The use of CI has been shown to be effective in knowing the fraction of the total functional dependencies that is
represented in an ERD. In this chapter, we propose a more refined approach to measure CI. This refinement is done on the basis of the extent a functional dependency is represented or not represented in an ERD instead of considering just binary values for $\text{count}(\text{FD}_i)$. That is, $0 \leq \text{count}(\text{FD}_i) \leq 1$. This suggests the need for introducing a fuzzy measure of completeness that indicates the degree to what extent a functional dependency is represented in an ERD. This approach is especially helpful in comparing the quality of two given conceptual models in terms of completeness if both have the same CI which is less than 1, as elaborated in the following example:

Consider the two ER diagrams given in Fig. 8.1 and Fig. 8.2, respectively, developed for the same problem domain.

![Fig. 8.1: A Sample ERD for Computing CI](image)

Let $\mathcal{I}$ be defined as:

$$\{A \rightarrow B, B \rightarrow C, W \rightarrow Y, X \rightarrow Z, X \rightarrow W\}$$

for this particular problem.
Now, for Fig. 8.1,

\[ f = \{ A \rightarrow B, X \rightarrow Z, X \rightarrow W \} \]

so, CI = 3/5 = 0.6.

Similarly, for Fig. 8.2,

\[ f = \{ A \rightarrow B, X \rightarrow Z, W \rightarrow Y \} \]

and hence CI = 3/5 = 0.6.

Therefore, we conclude that based upon CI, Fig. 8.1 and Fig. 8.2 are equally good; although they are not, intuitively, at least. Which of these figures is actually better depends upon a number of factors like the real-world semantics of the problem or relative positioning of the attributes in an ERD (i.e. on the same entity type or different entity types, as simple attribute or as a composite attribute). A factor like real-world semantics
requires different types of measures, for example, normalization index (section 7.2.2) and semantic correctness (section 9.2) whereas relative positioning is now being accommodated in the measure proposed in this chapter. It should also be noted that getting same CI values for two ER models of the same problem may imply any of the following possibilities:

1. The difference in the models is due to the difference in some quality parameter other than the completeness, or
2. irrespective of the values of other quality parameters for the two models, the difference is in the degree of completeness which is not indicated by CI.

The second possibility corresponds to the relative positioning of the attributes of the functional dependencies which are not represented in a given model.

Hence, we propose an improved metric, Fuzzy Completeness Index (FCI), and a set of rules to fuzzify [Ross 1995] the measure of completeness index. We also elaborate, with the help of an example:

- the computation of FCI, and
- that, the quality in terms of FCI of a conceptual model gets improved following the schema transformation rules already proposed in Part II of the thesis.

This chapter is organized in the following way: In section 8.2, a set of rules for transforming an ERD to a graph is presented. Our approach requires, for ease of
understanding and ease of implementation, that a given ERD be first transformed to a graph representation having nodes and edges. For this purpose, we propose a representation called Tauqueer-Awais-Shamail (TAS) Graph and a set of transformation rules which transforms a given ERD to its TAS Graph. In this section, we also propose an algorithm to map functional dependencies related to the problem on the specified graph. We also discuss how this graph can be searched to identify and derive database related information. In section 8.3, we formally define Fuzzy Completeness Index and introduce the methodology for calculating the FCI. Section 8.4 discusses the application of FCI to three different ERDs which are actually obtained through schema transformations of a single ERD. The final section concludes the findings of this research.

### 8.2 Transforming an ERD to a TAS Graph

The first step in computing the proposed metric namely FCI is to transform an ERD to a graph. In this section, we propose a colored multi-graph representation called Tauqueer-Awais-Shamail (TAS) Graph and a set of rules which can be applied for the transformation. The transformation is required to understand the computation of FCI as well as to automate its computation.

#### 8.2.1 Transformation Rules

Following are the rules for the transformation of a given ERD:

1. For every entity type $E_i$, draw a colored node $e_i$ on the graph. Such a node is called an E-node.
2. For every non-recursive relationship type \( R_k \) between entity types \( E_i \) and \( E_j \), draw a node \( r_k \) on the graph. Such a node is called an \( R \)-node. Also draw edges connecting \( r_k \) with \( E_i \) and \( E_j \).

3. For every attribute \( A_j \), which is not composite or part of a composite attribute, draw a node \( a_j \). Such a node is called an \( A \)-node.
   a. If \( A_j \) exists on an entity type \( E_i \), then draw an edge connecting \( a_j \) and \( e_i \).
   b. If \( A_j \) exists on a relationship type \( R_i \), draw an edge connecting \( a_j \) and \( r_i \).

4. For every composite attribute \( CA \) on the entity type \( E_i \) which is broken into attributes \( B_1, B_2, \ldots, B_k \), draw nodes \( b_1, b_2, \ldots, b_k \) respectively. Such nodes are called \( SA \)-nodes. Also draw edges connecting each \( SA \)-node with \( e_i \).

5. For every recursive relationship type \( R_j \) with entity type \( E_i \), draw a node \( r_j \) and a bi-directional edge between \( r_j \) and \( E_i \).

For demonstration, the above-mentioned rules are applied to an ERD given in Fig. 8.3 and a TAS graph is generated as shown in Fig. 8.4.

Fig. 8.3: A Sample ERD for Transformation to TAS Graph
8.2.2 Mapping Functional Dependencies on a TAS Graph

When an ERD has been transformed to its TAS graph, the next step is to mark FDs on the graph. We propose Algorithm 8.1 for this purpose.

**Algorithm 8.1: Labeling a TAS Graph with FDs**

Let $\mathcal{F}$ be the set of functional dependencies identified from the problem domain.

For every FD $X \rightarrow Y$ in $\mathcal{F}$:

If $X$ is a singleton set,

- search $A$-node $X$ on the TAS graph,
- search $A$-node $Y$ on the TAS graph,
- draw a directed edge from $X$ to $Y$
Else

together := null

For every $A_i \in X$

search $A$-node $A_i$ on the TAS graph,

draw a directed edge from $A_i$ to $Y$,

together := together + edge ($A_i$, $Y$)

and_edges (together)

For instance, consider the following set of functional dependencies:

$\{x \rightarrow b$

$x \rightarrow n$

$x, y \rightarrow h$

$l, j \rightarrow c$

$l \rightarrow a\}$

Applying Algorithm 8.1, Fig. 8.4 is converted to Fig. 8.5.
8.2.3 Searching TAS Graph

Searching the TAS graph for any E-node, A-node or R-node is quite straightforward, so functions like `findEnode()`, `is_Enode(Enode)`, `findAnode(Enode)`, `is_Anode(Anode)`, `findRnode()` and `is_Rnode(Rnode)` can be written. However, identifying key(s) for every entity type involves application of functional dependencies marked on the TAS graph. For this purpose, an algorithm for a function `findKey(Enode)` can be developed which searches an A-node from where uni-directed edges can span all A-nodes connected with the respective E-node.

Now, we can present our approach to refine the completeness index.
8.3 Fuzzy Completeness Index (FCI)

In this section, we introduce a fuzzy logic [Ross 1995] based approach to measure completeness of a conceptual model and propose a metric called Fuzzy Completeness Index (FCI). For ease of understanding, we first present the basics of fuzzy theory and then introduce our proposed metrics. The fuzzy concept of membership values and hedges has been incorporated, in this research, to define the FCI of an Entity-Relationship Diagram (ERD).

8.3.1 Fuzzy Theory

Fuzzy systems can be regarded as a formal mathematical complement to the traditional mathematical notion of set theory and classical logic. Although, the origins can be traced back to some 2500 years, the formal mathematical exposition of fuzzy logic is quite recent, appearing only about thirty years ago, but it is already finding utility at the leading edge of technological development [Jamshidi and Ross 1993]. Moreover due to its relative novelty, much ground both theoretical and applied remains to be covered and hence gives lot of area for research.

Since its beginnings in the sixties, with the seminal paper on fuzzy sets by Lotfi Zadeh [Zadeh 1965], literature on fuzzy logic has proliferated enormously and is widely accessible. A brief perspective on the relation of fuzzy logic to complex systems is presented herein (in order to provide the background for further discussion).
As described lucidly by Ross [Ross 1995], the real world is complex; complexity in the world generally arises from the uncertainty in the form of ambiguity. Problems featuring complexity and ambiguity have been addressed subconsciously by humans since they could think; these ubiquitous features pervade most social, technical and economic problems faced by humans. The question then arises that as to why information processing machines designed by humans, i.e. computers, are incapable of addressing complex issues? How can humans reason about real systems, when the complete description of a real system often requires more detailed data than a human could ever hope to recognize simultaneously and assimilate with understanding? The answer is that humans have the capacity to reason approximately, a capability that computers currently do not have. In reasoning about a complex system, humans reason approximately about its behavior, thereby maintaining only a generic understanding about the problem.

It is appropriate now, to quote Lotfi Zadeh, the founder of fuzzy logic on the relationship between complexity and ambiguity (imprecision). Zadeh [Zadeh 1973] says "As the complexity of a system increases, our ability to make precise and yet significant statements about its behavior diminishes until a threshold is reached beyond which precision and significance (or relevance) become mutually exclusive characteristics". And "The closer one looks at a real world problem, the fuzzier becomes its solution".

It is therefore for complex systems that fuzzy reasoning provides a way to understand system behavior by allowing to interpolate approximately between observed input output conditions.
8.3.2 Basic Fuzzy Logic Terminology

A description of basic fuzzy terminology is presented in this section.

Fuzzy Sets
The seminal paper by Lotfi Zadeh [Zadeh 1965] on fuzzy sets can be regarded as its beginnings. Fuzzy sets can be looked upon as an extension of crisp or Boolean sets. While crisp sets allow only either a full membership or no membership at all, fuzzy sets allow partial memberships. In a crisp set, the membership or non-membership of an element $x$ in a set $A$ is described by a characteristic function $\mu_A(x)$, where,

$$\mu_A(x) = 1 \text{ if } x \in A$$
$$\mu_A(x) = 0 \text{ if } x \notin A$$

Fuzzy set theory extends this concept by defining partial memberships which can take values ranging from 0 to 1,

$$\mu_A : X \rightarrow [0,1]$$

where $X$ refers to the universal set defined in a specific problem and $\mu_A$ is the membership function which returns a value of the degree of membership for an element $x$. The membership function can also be considered as a curve that defines how each point in a given domain of interest maps to a degree or extent of membership between the interval 0 and 1.
If the universal set is countable and finite, then a fuzzy set $A$ in this universe can be defined by listing each member and its degree of membership in the set $A$,

$$A = \mu_A(x_1) / x_1 + \cdots + \mu_A(x_n) / x_n,$$

or

$$A = \sum_{i=1}^{n} \frac{\mu_A(x_i)}{x_i}, \quad (8.1)$$

Similarly, if $X$ is continuous, then a fuzzy set can be defined by,

$$A = \int \frac{\mu_A(x)}{x}, \quad (8.2)$$

A fuzzy set can therefore be represented as a set of ordered pairs of the generic element $x$ and its degree of membership $\mu_A$.

It should be noted that in these definitions, "/" does not refer to a division and is used as a notation to separate the membership of an element from the element itself. The symbol "+" denotes the set theory union operator rather than arithmetic sum.

As mentioned above the membership function can be considered as a curve. This curve can take various shapes such as triangular, trapezoidal, Gaussian, bell, pi etc., or any arbitrary shape. Usually, the first two types of curve i.e. triangular and the trapezoidal which is essentially a truncated triangular curve, are adequate for most applications and do not impose a high computational overhead.

In general a fuzzy set may have linguistic variables attached to it. Suppose we have a set containing temperature values of a room represented by $A$, then the linguistic variables
that may categorize A could be subset comprising of high temperature, medium temperature and / or low temperature. Here high, medium and low is the linguistic description of temperature. In addition "hedges" can be introduced to modify fuzzy membership values of the linguistic variables such as: "very" high temperature can be represented as $[\mu_A(x)]^2$, or “very very” high temperature can be represented as $[\mu_A(x)]^3$.

**Membership Value Assignment**

There are possibly many ways of assigning the membership values or functions to fuzzy variables. This assignment process can be intuitive or it can be based on some algorithmic or logical equations. In this chapter, we have devised a new way of assigning membership values to an ERD from a family of ERDs pertaining to the same problem domain. The membership thus determined indicates the extent of completeness of an ERD with reference to other related ERDs. This extent of completeness is referred to as Fuzzy Completeness Index (FCI) in this chapter. The proposed FCI is explained in the next subsection.

It is worth mentioning here that application of fuzzy logic in the field of conceptual modeling and databases has been studied in the research. [Zvieli and Chen 1986, Chaudhry et al. 1994, Chen and Kerre 1998, Vert et al. 2000, Ma et al. 2001] have suggested various extensions to the ER model in order to manage fuzzy data [Galindo et al. 2001]. [Galindo et al. 2001] have introduced fuzzy quantifiers to fuzzify structural constraints of the EER model. [Urrutia et al. 2002] have suggested Fuzzy Attributes. [Chaudhry et al. 1994] have proposed a method for designing Fuzzy Relational Databases
based on the extension to the ER model suggested by [Zvieli and Chen 1986]. In the context of structural complexity of an ERD, [Piattini et al. 2001] have developed a fuzzy rule system to predict the *understandability* (section 3.5) of the ERD based upon given values for metrics proposed in their research. However, to the best of our knowledge, a fuzzy measure (e.g. FCI) for assessing the quality of a conceptual model has not been introduced in the literature so far.

### 8.3.3 Proposed Fuzzy Completeness Index (FCI)

The following expression represents the Fuzzy Completeness Index (FCI):

\[
FCI_{overall} = \frac{\sum_{i=1}^{n} FCI_i}{n(3)},
\]

where \( FCI_i \) is the FCI value of \( i^{th} \) functional dependency.

For a given functional dependency \( FD_i: X \rightarrow Y \), the FCI value can be calculated using the formula given below.

\[
FCI_i = 1 - \frac{TAS(FD_i)}{A_{Total}},
\]

where \( TAS \) is a fuzzy operator applied on a functional dependency for a given TAS graph, and \( A_{Total} \) represents the total number of attributes present in the ERD under consideration. The TAS operator works on the basis of following rules:
**Rule F1:** The value for $TAS(FD) \rightarrow ZERO$, if $\{X, Y\} \subseteq Attr(E_i)$ \mid X$ is a Key connected to some $i^{th}$ E-Node (found using findKey($E_i$)) in a TAS Graph. This suggests that $FCI$, in this case, approaches 1 using Eq. 8.4.

**Rule F2:** The value for $TAS(FD) \rightarrow ONE$ and consequently FCI approaches 0, if $\{X, Y\} \not\subset \cup (Attr(E_i))$, that is, either $X$ or $Y$ (A-Node or SA-Node) or both do not belong to any E-Node in a TAS graph.

**Rule F3:** The value of $TAS(FD) \rightarrow \gamma$, if

(a) $\{X\} = Key(E_i)$ and $Y \in Attr(E_j)$ for some E-Nodes $E_i$ and $E_i \neq E_j$

(b) $\{X\} = Key(E_i)$ and $\{Y\} \subseteq Attr(CA_j)$; for some $E_i$ and $CA_j$ is the $j^{th}$ A-Node on $E_i$.

(c) $Y \in Attr(E_i)$ and $\{X\} \subseteq Attr(CA_j)$ and $\{X\} = Key(E_i)$; where $E_i$ is some E-Node in the TAS graph and $CA_j$ is the $j^{th}$ A-Node with SA nodes as its children on E-Node $E_i$.

(d) $\{X\} \neq Key(E_i)$

For **Rule F3 (a)**, the value of $\gamma$ represents the effort required to bring X and Y on the same E-Node. This effort is calculated by applying the counting operator on the number of R-Nodes that separate $E_i$ and $E_j$. 
For Rule F3(b) and Rule F3(c), the concept of fuzzy hedge is used to find out the FCI. In this case the formula for calculating FCI is modified as below:

\[ FCI_i = \left[ 1 - \frac{TAS(FD_i)}{A_{total}} \right]^m. \]  \hspace{1cm} (8.5)

Here \( \gamma \) is found by applying the counting operator on the number of composite attributes nodes present between X and Y and the number of R-Nodes that separate \( E_i \) and \( E_j \) when \( E_i \neq E_j \), and \( m = \gamma + 1 \).

For Rule F3(d), The TAS operator approaches \( \gamma \) that is obtained by applying Rule F3(a), Rule F3(b) or Rule F3(c) plus the effort to mark X as the key attribute on \( E_i \). This effort can be envisaged through hypothetically applying the schema transformation rules suggested in chapter 4, according to which a new E-Node \( E_p \) is introduced which connects to \( E_i \) through a new R-Node. The calculation of the TAS operator can be illustrated with the help of an example.

Consider a TAS graph as given in Fig. 8.6(a). After the schema transformation rules (chapter 4) are applied to Fig. 8.6(a), a modified TAS graph shown in Fig. 8.6(b) is obtained. The count for the TAS operator for Fig. 8.6(a) will be 1 (i.e. marking X as a key) and the FCI value turns out to be 0.75. On the other hand, the TAS operator assumes a value of zero in Fig. 8.6(b) and consequently the value of FCI becomes 1; demonstrating the completeness of TAS graph 8.6(b).
Now, we propose an algorithm for finding the FCI using the TAS graph process. This algorithm is presented as Algorithm 8.2.

**Algorithm 8.2: Computing FCI of a TAS Graph**

*Convert an ERD into a graph*

*For a given Functional Dependency X → Y*

*Find attribute nodes X and Y in the graph*

*IF X or Y not found THEN*

\[
FCI_i = 0
\]

*ELSE*

Counter = 0, \(\gamma = 0\)

*IF X is not a key attribute THEN*

Mark X as a key attribute

Increment Counter by 1
IF $Y$ belongs to the set of all attributes on which directed edges emerging from $X$ incident THEN

$$TAS(FD_i) = 0, \text{ and } FCI_i = 1 \text{ from Eq. 8.4}$$

ELSE

Count the number of R-Nodes (stored in CountR) and A-Nodes (stored in CountA) that separate $X$ and $Y$

Also count the nodes separating $X$ from its nearest E-Node (stored in CountX) and the nodes separating $Y$ from its nearest E-Node (stored in CountY)

Counter = Counter + CountR + CountA + CountX + CountY

$\gamma = \text{Max} (\text{CountX, CountY})$

$$TAS(FD_i) = \text{Counter}$$

$m = \gamma + 1$

Calculate $FCI_i$ from Eq. 8.5

REPEAT above steps for all FDs

Calculate $FCI_{overall}$ from Eq. 8.3

In the next section, we elaborate our methodology of computing FCI with the help of a real-world example (the same scenario as we discussed in chapter 7).
8.4 Evaluating Quality: An Example

Consider the scenario:

“In our company, we want to keep track of the employees’ information regarding which project he is working in and which department he belongs to. An employee is uniquely identified by an ID and his name, department# he belongs to, the respective department name and department location are stored. Also recorded for an employee are some other attributes like project# for the project he is presently working on, its location, cost and supervisor. A supervisor is assigned the responsibility for each location a project is carried out. Our company may assign same project #s to two different projects but then those should be located at two different locations. An employee belongs to one department only and is not allowed to work on more than one project. Queries can refer to any of the data specified here.”

Now, for this description, we may get an ERD given in Fig. 8.7. Applying the rules and Algorithm 8.1 on this ERD, the corresponding TAS graph is shown in Fig. 8.8. For this example has already been given in chapter 7 but for sake of ease of reference is reproduced below:
Fig. 8.7: An ERD for a Company Database

Fig. 8.8: TAS Graph for ERD in Fig. 8.7

\[ FD1: P\#, LOC \rightarrow Cost \]
\[ FD2: LOC \rightarrow Super \]
\[ FD3: ID \rightarrow Name \]
\[ FD4: ID \rightarrow Dept\# \]
\[ FD5: Dept\# \rightarrow DName \]
\[ FD6: Dept\# \rightarrow DLoc \]
\[ FD7: ID \rightarrow P\# \]
\[ FD8: ID \rightarrow LOC \]
\[ FD9: ID \rightarrow Cost \]
\[ FD10: ID \rightarrow Super \]
As we noted in chapter 7, \( f \) consists of only \( FD3, FD4, FD7 \) through \( FD10 \). So, \( CI = \frac{6}{10} = 0.6 \).

The values of FCI\( s \) for each FD as per the rules described in the previous section are:

- FCI\(_1\): 0.308  
- FCI\(_2\): 0.444  
- FCI\(_3\): 1.000  
- FCI\(_4\): 1.000  
- FCI\(_5\): 0.875  
- FCI\(_6\): 0.875  
- FCI\(_7\): 0.790  
- FCI\(_8\): 0.790  
- FCI\(_9\): 0.790  
- FCI\(_{10}\): 0.790

Overall FCI can be computed as:

\[
FCI_{\text{overall}} = \frac{\sum_{i=1}^{n} FCI_i}{n(f)} = \frac{8.132}{10} = 0.766. 
\]  

(8.6)

Another ERD for the same problem domain is represented in Fig. 8.9, which is an improved version of Fig. 8.7.
The corresponding TAS graph for Fig. 8.9 is shown in Fig. 8.10. The CI value for the new TAS graph is 0.7, whereas the overall FCI value comes out to be 0.972.
Fig. 8.11: Improved Conceptual Schema after Second Transformation

Fig. 8.12: TAS Graph for ERD in Fig. 8.11
Fig. 8.12 represents yet another TAS graph for the conceptual schema shown in Fig 8.11. For TAS graph in Fig. 8.12, the CI value is 1 and the overall FCI is 1.000.

Computations in this approach may seem very complex and time-consuming making this approach hard to apply in practice. However, we have the following two observations:

1. The computation of FCI is not required in every case. Table 8.1 summarizes all possibilities and presents cases when FCI computations are recommended.

2. For the cases when FCI need to be computed, algorithms are proposed which can be automated in any appropriate case tool. As stated in chapter 1 (section 1.1), the objective is to reduce the database designers’ subjectivity in the conceptual design stage, which otherwise has always been a problem [Noah and Lloyd-W 1995] in evaluating the quality of a conceptual model.

<table>
<thead>
<tr>
<th>Comparable Models</th>
<th>Sets of Functional Dependencies</th>
<th>Completeness Index</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \xi_1 = \xi_2 )</td>
<td>Same ( (f_1 = f_2) )</td>
<td>( \text{CI}_1 = \text{CI}_2 )</td>
<td>Trivial Case. FCI not required.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{CI}_1 \neq \text{CI}_2 )</td>
<td>Not Possible</td>
</tr>
<tr>
<td></td>
<td>Different ( (f_1 \neq f_2) )</td>
<td>( \text{CI}_1 = \text{CI}_2 )</td>
<td>Not Possible</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{CI}_1 \neq \text{CI}_2 )</td>
<td>Not Possible</td>
</tr>
<tr>
<td>( \xi_1 \neq \xi_2 )</td>
<td>Same</td>
<td>( \text{CI}_1 = \text{CI}_2 )</td>
<td>FCI recommended</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{CI}_1 \neq \text{CI}_2 )</td>
<td>Not Possible</td>
</tr>
<tr>
<td></td>
<td>Different</td>
<td>( \text{CI}_1 = \text{CI}_2 )</td>
<td>FCI recommended</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \text{CI}_1 \neq \text{CI}_2 )</td>
<td>FCI not required</td>
</tr>
</tbody>
</table>

Table 8.1: Decision Rules for applying FCI

Note: \( \xi_1 \) and \( \xi_2 \) are the two comparable conceptual models for the same problem, \( \mathcal{I} \) be the set of functional dependencies related to the problem domain, and \( f_1, f_2 \subset \mathcal{I} \).
Hence, it can be concluded that we need to compute FCI only when we have two different ER models giving same CI values.

Before we conclude this chapter, for sake of completion of discussion on completeness index, we find it important to give some observations, in the next section, on the behavior or trend of CI and FCI with respect to the proposed schema transformations.

8.5 Effect of Schema Transformations on Completeness Indices

In order to study the effect of schema transformations on CI and FCI, the data in [TR CS001-2006] were analyzed. The data relates to six scenarios presented in [TR CS001-2006] for 3 different problem domains – one the Company database referred throughout in this thesis, two a County-Lots case given in [Elmasri and Navathe 2004] and three a Library database to store information about books and authors. Schema transformations were then applied to each case (a scenario) and it was observed that in some cases, only a single transformation was required whereas in other cases there were two transformations to get n(f) = n(Ω). In the next two sub-sections, we discuss the effect of schema transformations on CI and FCI one by one.
8.5.1 Effect on CI

For each case study [TR CS001-2006], CI is computed for original ERD and the transformed ERDs, and the results are summarized in Table 8.2. It is observed that CI varies in discrete linear fashion (similar to a non uniform staircase function) with respect to proposed transformations. Every transformation does increase n(f) but the increase may not always be the same. Hence the increase in the value of CI after the application of schema transformation rules is also non-uniform. Theoretically, the step between transformations should be the same thus approximating the behavior as a continuous staircase function. In practical terms, the change in the CI values for a given conceptual model approximates as a discontinuous staircase function or a ceiling function. The data reveal that:

\[ R_{step} \propto n(f) \]

which means that the rise in the step \( R_{step} \) is non-uniform.

<table>
<thead>
<tr>
<th>Case Study</th>
<th>CI for Different Transformations</th>
<th>Step Rise</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0.25</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td>0.67</td>
<td>1.00</td>
</tr>
<tr>
<td>3</td>
<td>0.83</td>
<td>1.00</td>
</tr>
<tr>
<td>4</td>
<td>0.20</td>
<td>0.70</td>
</tr>
<tr>
<td>5</td>
<td>0.82</td>
<td>0.91</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0.75</td>
</tr>
</tbody>
</table>
Since we are not restricting our transformations to increment \( n(f) \) by exactly one, the increase in \( n(f) \) can be by one or more in the next transformation. Therefore, the \( R_{\text{step}} \) will not be a constant. Nevertheless, with every transformation, a monotonic increase in CI value is observed.

### 8.5.2 Effect on FCI

The main reason to introduce FCI is to convert the discontinuities in CI into a continuous behavior in order to capture the information loss which is observed otherwise. Table 8.3 summarizes the computations of FCI for the case studies. It reveals that the variation in the FCI is much smoother than the CI. For the case studies gone through two transformations (that is, case study 4, 5 and 6), it is also observed that FCI values show a logarithmic trend. Ideally a linear continuous behavior may be expected but practically the variation approximates a logarithmic trend of the form:

\[
\alpha \ln [n(f)] + \beta,
\]

<table>
<thead>
<tr>
<th>Case Study</th>
<th>FCI for Different Transformations</th>
<th>Step Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0.771</td>
<td>1.00</td>
</tr>
<tr>
<td>2</td>
<td>0.944</td>
<td>1.00</td>
</tr>
<tr>
<td>3</td>
<td>0.857</td>
<td>1.00</td>
</tr>
<tr>
<td>4</td>
<td>0.769</td>
<td>0.957</td>
</tr>
<tr>
<td>5</td>
<td>0.957</td>
<td>0.985</td>
</tr>
<tr>
<td>6</td>
<td>0.725</td>
<td>0.958</td>
</tr>
</tbody>
</table>
where \(a\) and \(b\) are constants for a particular case study under consideration. Nevertheless, with every transformation, a monotonic increase in FCI values is observed.

### 8.6 Summary

In this chapter, we demonstrated with the help of an example that two different ER diagrams for the same problem may have the same values for CI. This anomaly emphasizes the need of improving the concept of CI. We therefore, proposed a fuzzy based approach to incorporate in the completeness index the notion of partial existence of a functional dependency in an ER model. A quality metric, Fuzzy Completeness Index was proposed for this purpose. The proposed quality measure provides a more realistic method of quantifying the completeness index of the ERD. The measure indicates the effort required to make the model more complete, simply because the metric varies between 0 and 1 and has been fuzzified. The research used fuzzy membership values and fuzzy hedges to calculate the proposed FCI. The proposed measure was applied to a scenario for real life situation and it was demonstrated how quality was improved in terms of completeness for various schema transformations using rules already proposed in Part II of the thesis.

The method proposed for computing FCI was based on three steps – 1) Transforming a given ERD to our proposed TAS graph for which a graph notation and a set of transformation rules were proposed, 2) Mapping a given set of functional dependencies to the TAS graph for which we proposed an algorithm, and 3) Computing FCI from the mapped TAS graph for which another algorithm was proposed.
In the next chapter, we are analyzing another aspect of quality of an ER model – the structural complexity. We intend to discuss how it can also be measured in quantitative terms to meet the objectives of our research.
Chapter 9

Measuring Structural Complexity of A Conceptual Model

9.1 Introduction

It has been discussed in chapter 3 that the ISO/IEC 9126 standard [ISO99] defines software quality as composed of six external characteristics – functionality, reliability, usability, efficiency, maintainability and portability. For an Entity-Relationship Diagram, maintainability can be assessed by its structural complexity and the structural complexity itself can be determined by the two sub-characteristics called understandability and modifiability. As discussed in section 3.5, GENERO ET AL. [Genero et al. 2002] identified some measures which have a high correlation with the understandability time and the modifiability time. However, their research was restricted to specifying the correlation and it did not specify any metric. In chapter 3, we have concluded that there is a need to define these characteristics quantitatively as a metric which can be easily applied to practical problems.

Therefore, in this chapter, we intend to propose metrics for understandability and modifiability that are easy to use in practical applications and that can contribute towards determining the quality of a conceptual model. In the following section, section 9.2, we
formally define our proposed metric for understandability and describe how to compute its value. Section 9.3 defines a metric for modifiability of an ER diagram and describes the methodology for its computation. In section 9.4, we demonstrate the application of these metrics to a small real-world situation. In the last section of this chapter, we summarize our work.

9.2 Understandability and Correctness

For understandability, defining a precise metric is not an easy task. We therefore introduce correctness as an appropriate measure to predict understandability. Correctness can be further categorized into syntactic correctness and semantic correctness. We define syntactic correctness as usage of correct symbols of an ERD for corresponding concepts in the problem domain and we define semantic correctness as the representation of all real world requirements in the ERD correctly. Based upon these definitions, it can be understood that syntactic correctness can be easily dealt with, for example, training the designer to use ERD constructs correctly. Hence, we focus on semantic correctness only.

**Definition 9.1 – (Semantic Correctness):** Semantic correctness can be measured as the fraction of requirements correctly represented in the ERD to total number of requirements explicitly or implicitly stated by the user in his problem statement. Mathematically, let:

\[ N_r = \text{Total number of requirements stated in the problem domain}, \]

\[ N_r' = \text{Number of requirements represented in the ERD out of } N_r, \text{ and} \]
\[ C = \text{Correctness of a given ERD with respect to the given problem domain, then} \]

\[ C = \frac{N_r^*}{N_r} \quad (9.1) \]

It can be noted that \( C \) is always bounded as \( 0 \leq C \leq 1.0 \).

We have observed that user requirements are represented on an ERD into the following forms:

1. An attribute marked on an entity type
2. An attribute marked on a relationship type
3. An attribute represented as of a proper type, for example, a single-valued attribute, a multi-valued attribute or a composite attribute.
4. An entity type represented as of a proper type, for example, a strong entity type or a weak entity type.
5. Key constraints, for example, key or partial key attributes
6. A relationship type between a number of entity types.
7. Structural constraints on entity types, that is, cardinality and participation constraints

This implies that, for a given problem, requirements of the above forms can be listed and then can be checked against the corresponding ERD. Applying Definition 9.1 will then yield understandability (in terms of correctness) of the ERD.

### 9.3 Modifiability

When a change is requested in an ERD, this change can result in any of the following:
1. Addition / deletion of an entity type
2. Addition / deletion of an attribute
3. Change in attribute type
4. Addition / deletion of a relationship type
5. Change in cardinality of a relationship type
6. Change in participation constraints of a relationship type
7. Change in degree of a relationship type

A change requested by a user can be decomposed into a number of changes which can further be classified into a number of categories according to some appropriate classification criteria similar to the one given above. With each category a weight can be associated which describes a relative effort required to make the desired change. Now, modifiability can be defined in terms of the total effort required to make the required change. Let us formally define this metric now.

**Definition 9.2 – Change Decomposition:** If a change $C$ can be decomposed into $n_1$ changes of category $C_1$, $n_2$ changes of category $C_2$, and so on up to $n_m$ changes of category $C_m$, then a decomposition $D$ of $C$ can be expressed as:

$$C = D (n_1C_1, n_2C_2, ..., n_mC_m)$$  \hspace{1cm} (9.2)

**Definition 9.3 – Effort-to-Change:** Let $D$ be a decomposition of $C$ and $w_i$ be the relative effort required for change in the $i^{th}$ category, then a measure Effort-to-Change (ETC) can be defined as follows:
\[ ETC = \sum_{i=1}^{m} (n_i \times w_i) \]  \hspace{1cm} (9.3)

**Definition 9.4 – Modifiability:** Let \( D \) be a decomposition of \( C \) and \( ETC \) is the effort-to-change, then modifiability can be expressed as:

\[ M = \frac{1}{ETC} \]  \hspace{1cm} (9.4)

It can be noted that if \( \text{Min}(w_1, w_2, \ldots, w_m) = 1 \), then \( M \) is always bounded as \( 0 < M \leq 1.0 \).

In order to elaborate the computational process for \( ETC \), let us classify changes into the following categories:

1. Adding (or deleting) a Regular Entity Type
2. Adding (or deleting) a Weak Entity Type
3. Adding (or deleting) a Relationship Type
4. Changing an Attribute Type
5. Changing Structural Constraints (Cardinality or Participation)
6. Changing Degree of a Relationship Type

In addition, we propose the weights as shown in Table 9.1. Though any appropriate weights can be selected, weights in Table 9.1 are proposed based upon our experience, representing the relative effort required to introduce the respective change. For example, changing an attribute type, say from a single-valued attribute to a multi-valued attribute,
is quite trivial whereas adding a weak entity type requires identification of the owner
entity type, identifying relationship type, partial key attributes and so on.

<table>
<thead>
<tr>
<th>Category</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding a Regular Entity Type</td>
<td>1.0</td>
</tr>
<tr>
<td>Adding a Weak Entity Type</td>
<td>1.5</td>
</tr>
<tr>
<td>Adding a Relationship Type</td>
<td>0.5</td>
</tr>
<tr>
<td>Changing an Attribute Type</td>
<td>0.5</td>
</tr>
<tr>
<td>Changing Structural Constraints</td>
<td>0.5</td>
</tr>
<tr>
<td>(Cardinality or Participation)</td>
<td></td>
</tr>
<tr>
<td>Changing Degree of a Relationship Type</td>
<td>2.0</td>
</tr>
</tbody>
</table>

### 9.4 Measuring Structural Complexity: A Hypothetical Case

Consider the following scenario described by a user:

“In our company, we want to keep track of the employees’ information regarding
which project he is working on and which department he belongs to. An employee
is uniquely identified by his ID. Along with an employee’s name, the department
number of the department he belongs to, the respective department name and the
department location are also stored. Also recorded for an employee are some other attributes like project number for the project he is presently working on, its location, cost and supervisor. A project number is uniquely assigned at a location.”

For this description, different designers may develop different ERDs; however, in order to elaborate our approach, we discuss only two possible solutions given in Fig. 9.1 and Fig. 9.2.

![Fig. 9.1: An ERD for a Company Database](image)

These figures of course are not exactly equivalent; however, depending upon the designer’s experience and his perception of the problem statement, all these figures can possibly be developed from the same problem statement.

We further add to the scenario that after some time a new requirement is to be introduced. This requirement is stated below:

“A supervisor is assigned the responsibility for each location a project is carried out and every project is being controlled by a department.”
9.4.1 Computing Correctness

Following set of requirements can be identified from the given problem statement:

R1: A number of employees
R2: A number of departments
R3: A number of projects
R4: An employee is uniquely identified by ID
R5: A department is uniquely identified by Dept#
R6: A project is uniquely identified by P#,LOC
R7: An employee is related to a project
R7-1: An employee may be related to many projects
R7-2: A project may be related to many employees
R8: An employee is related to a department
R8-1: An employee is related to a single department only
R8-2: A department may be related to several employees
R8-3: Every employee is related to a department

Fig. 9.2: Another ERD for the Same Application
Table 9.2  Computing Understandability

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Fig. 9.1</th>
<th>Fig. 9.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R2</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R3</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R4</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R5</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>R6</td>
<td>X</td>
<td>√</td>
</tr>
<tr>
<td>R7</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R7-1</td>
<td>X</td>
<td>√</td>
</tr>
<tr>
<td>R7-2</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R8</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R8-1</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R8-2</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R8-3</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Correctness</td>
<td>9/13</td>
<td>11/13</td>
</tr>
</tbody>
</table>

For Fig. 9.1 and Fig. 9.2, understandability can be computed on the basis of information presented in Table 9.2: Now, let us evaluate the understandability of the modification:

The change requested results in two additional requirements:
R9: A supervisor related to a project location

R10: A project related to a department

These changes might lead to modifications of ERDs given in Fig. 9.1 and Fig. 9.2 resulting in the development of Fig. 9.3 or Fig. 9.4. For these modified ERDs, Table 9.2 is reconstructed as Table 9.3.

Fig. 9.3: Modifications to Fig. 9.1
9.4.2 Computing Modifiability

Introducing the new requirement may possibly lead to another design, given in Fig. 9.5, which is evaluated for modifiability. It can be observed that Fig. 9.2 is more easily transformed to Fig. 9.5 than Fig. 9.1. Let us now quantitatively determine the ease based upon our proposed approach. For Fig. 1, changes and associated weights are summarized in Table 9.4. Therefore Modifiability is $1/8.5 = 0.118$.

Table 9.5 shows computation of ETC for Fig. 9.2. Therefore, Modifiability is $1/6.0 = 0.167$. 

Fig. 9.4: Modifications to Fig. 9.2
<table>
<thead>
<tr>
<th>Requirement</th>
<th>Fig. 9.3</th>
<th>Fig. 9.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R2</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R3</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R4</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R5</td>
<td>X</td>
<td>√</td>
</tr>
<tr>
<td>R6</td>
<td>X</td>
<td>√</td>
</tr>
<tr>
<td>R7</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R7-1</td>
<td>X</td>
<td>√</td>
</tr>
<tr>
<td>R7-2</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R8</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R8-1</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R8-2</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R8-3</td>
<td>X</td>
<td>√</td>
</tr>
<tr>
<td>R9</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>R10</td>
<td>X</td>
<td>√</td>
</tr>
<tr>
<td><strong>Correctness</strong></td>
<td><strong>10/15</strong></td>
<td><strong>1</strong></td>
</tr>
</tbody>
</table>
Table 9.4  Effort Required For Fig. 9.1

<table>
<thead>
<tr>
<th>Ci</th>
<th>n_i</th>
<th>w_i</th>
<th>n_iw_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding Regular Entity Types</td>
<td>2</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Adding Weak Entity Type</td>
<td>1</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>Adding Relationship Types</td>
<td>3</td>
<td>0.5</td>
<td>1.5</td>
</tr>
<tr>
<td>Attributes Moved to Regular ET</td>
<td>5</td>
<td>0.5</td>
<td>2.5</td>
</tr>
<tr>
<td>Attributes Moved to Weak ET</td>
<td>2</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td><strong>ETC</strong></td>
<td></td>
<td></td>
<td><strong>8.5</strong></td>
</tr>
</tbody>
</table>

Fig. 9.5: Modified ERD after Incorporating the Change
### Table 9.5 Effort Required For Fig. 9.2

<table>
<thead>
<tr>
<th>$C_i$</th>
<th>$n_i$</th>
<th>$w_i$</th>
<th>$n_i w_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adding Regular Entity Types</td>
<td>1</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Adding Weak Entity Type</td>
<td>1</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>Adding Relationship Types</td>
<td>2</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td>Attributes Moved to Regular ET</td>
<td>3</td>
<td>0.5</td>
<td>1.5</td>
</tr>
<tr>
<td>Attributes Moved to Weak ET</td>
<td>2</td>
<td>0.5</td>
<td>1.0</td>
</tr>
<tr>
<td><strong>ETC</strong></td>
<td></td>
<td></td>
<td><strong>6.0</strong></td>
</tr>
</tbody>
</table>

### 9.5 Summary

It has been discussed in this chapter that structural complexity is an important quality aspect of an ER model. Structural complexity can be measured in terms of understandability and modifiability. We have suggested that understandability of an ERD should be measured in terms of its correctness which refers to the fraction of requirements correctly represented in the conceptual model. We have also proposed that modifiability can be determined in terms of effort required to make a desired change. We have therefore proposed definitions and formulae for these metrics accordingly. In order to elaborate the use of correctness and modifiability metrics, we have presented a scenario and have demonstrated the computations for proposed metrics. The
methodology verifies that these computations are fairly simple and therefore can be applied easily to any practical problem. Like quality metrics proposed in chapter 7 and chapter 8, these metrics can play an important role to compare and to evaluate the quality of two ER diagrams objectively, for the same problem.
Part IV

Conclusions and Further Work
Chapter 10

Conclusions and Further Work

10.1 Conclusions

We have explored, in this thesis, the existing database design methodology and its evolution in the context of relational database management systems. Starting from the era when databases were introduced, the design methodology comprised of two stages: defining a relational schema, and its normalization to remove data anomalies. Later, it was realized that conceptual modeling had an important impact on the database design process, so it was introduced as the first stage in the database design methodology, that is, before that of defining the relational schema, as shown in Fig. 10.1. We have focused our research on the ER model which is widely used to define a conceptual model. We have observed that since 1976, when the original ER model was proposed, researchers have been suggesting various improvements in the ER modeling technique in order to reduce the semantic gap between the real-world problem and the corresponding ER model. In this thesis, we have successfully identified some of the semantic gaps and have proposed solutions in terms of multi-valued relationship attributes and single-valued relationship attributes. With examples from real-world, we have demonstrated the usefulness and effectiveness of our proposed concepts in resolving the semantic ambiguities arising otherwise.
We have also identified the problem of unavailability of structured and well-defined rules which can be used to improve a conceptual model (that is an ER model) developed by a database designer for a particular problem domain. In the absence of such rules, one has to rely on skills and expertise of the database designer. As an important part of this thesis, we have suggested a set of rules which can transform a conceptual schema or an ER schema to an improved ER schema. Basis of these rules is incorporating real-world constraints in the ER model identified in the form of functional dependencies. These schema transformations take place within the conceptual design stage in a systematic manner. While identifying certain types of composite attributes and non-key attributes in an ER schema, our rules suggest that appropriate entity types and relationship types
should be added in the model. These rules also define what attributes to be marked for newly added entity types.

Hence, we have proposed that improving a conceptual model systematically should be considered as a new stage to be introduced after the initial conceptual modeling stage. This proposed improvement is shown in Fig. 10.2. Having defined proper rules for it and demonstrated their effectiveness with practical examples, we believe that the new stage will create a significant impact on the database design methodology. As a matter of fact, we have studied this impact in two dimensions. Firstly, we have observed that the resulting ER model after schema transformations is quite closer to the real world scenario and is more flexible towards future enhancements. Secondly, there is a significant impact on the quality of relational schema obtained after mapping the respective ER schema. The quality of a relational schema is usually described in terms of the highest normal form satisfied by its relations. We have observed that following our proposed schema transformation rules, a relational schema satisfies Boyce-Codd Normal Form (BCNF) and hence the normalization stage for this purpose need not be considered.

We have also validated our schema transformation rules using a “reverse-engineering” approach. We started with a relation schema and analyzed the reasons due to which it does not satisfy certain normal forms. We have observed the following in this regard:

- violation of 1NF is not possible if composite attributes are properly represented in an ER model,
violation of 2NF and 3NF can occur when some functional dependencies are ignored in an ER model. A set of rules has been proposed which states how these dependencies should be represented by adding weak entity types and regular entity types,

- applying the proposed rules always generates relation schemas that do not violate the respective normal forms,

- the proposed rules are equivalent to the schema transformation rules already proposed for semantic improvement. Therefore, these rules get validated through our approach.
for BCNF, violation cannot occur provided that a relation satisfies up to 3NF and its primary key consists of only one attribute.

We have also observed that the existing literature does not have quantitative metrics to compute the quality of a model. Consequently, it is not possible quantitatively to compare two models for the same problem in order to decide which one is of better quality. Without such metrics, any improvements suggested in the modeling technique can only be considered subjective. This deficiency has motivated us to define appropriate and easy to apply metrics which can determine the quality of an ER schema.

We have initially defined three quality metrics – a Completeness Index (CI), a Normalization Index (NI) and an overall Quality Index (QI). The first two metrics are defined in line with our approach of the thesis discussed above in which improving a conceptual model is studied along two dimensions – the semantic quality and the normalization quality of the relational schema. CI deals with the semantic quality and so is based upon number of functional dependencies represented in an ER schema related to a real world problem. NI deals with a quantitative measure corresponding to the highest normal form each relation of the database schema is in. The overall quality index quantifies the overall quality of an ER schema based upon CI and NI (and other metrics, if available). This helps in comparing two models in terms of their overall quality. We have demonstrated the ease of application of these metrics to examples related to practical scenarios. In addition, we are encouraged to study the quality of proposed schema transformations using these metrics. The results have verified that schema
transformations improve the quality in the quantitative terms, too. On our examples, it has shown 67% improvement which is quite significant.

However, it has also been identified that the quality metrics we proposed are though effective but need further refinement. We have shown with an example that two different ER diagrams for the same problem may have the same values for CI. This anomaly has motivated us to improve the concept of CI. We have analyzed that the anomaly arises due to the fact that our approach either completely counts a functional dependency or does not count it at all (depending upon whether it exits in the model or not, respectively). We have therefore suggested a fuzzy based approach to incorporate in the completeness index the notion of partial existence of a functional dependency. A quality metric, Fuzzy Completeness Index (FCI) has been proposed accordingly. The approach introduces a graph notation named as a TAS Graph to which the ERD and the set of functional dependencies are mapped. On the TAS Graph, based on relative positioning of the attributes involved in the functional dependencies, FCI is computed. Algorithms for mapping and computing FCI have thus been proposed. The proposed quality measure provides a more realistic method of quantifying the completeness index of the ER schema. The proposed metric is applied to the same scenario for which CI has been computed initially and it has been demonstrated successfully how two different models using the new metric can be compared more precisely for their semantic quality.

Finally, we have addressed another important aspect of quality of an ER model. This aspect is the structural complexity which has already been defined in the literature. It has
been suggested that structural complexity can be measured in terms of understandability and modifiability. However, we have identified the need of defining metrics for these quality attributes quantitatively. After studying the problem, we have suggested that understandability of an ERD should be measured in terms of its correctness which refers to the fraction of requirements correctly represented in the conceptual model. We have also proposed that modifiability can be determined in terms of effort required to make a desired change. We have therefore proposed definitions for these metrics accordingly. We have also shown that the proposed metrics for understandability and modifiability can play an important role to compare and to evaluate the quality of two ER diagrams objectively.

The results of our proposed approach are summarized in Table 10.1. We have given pointers to further research in the next section.
<table>
<thead>
<tr>
<th><strong>EXISTING APPROACH</strong></th>
<th><strong>PROPOSED APPROACH</strong></th>
<th><strong>IMPACT / RESULTS</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>3-stage DB design</td>
<td>4-stage DB design</td>
<td>Conceptual model</td>
</tr>
<tr>
<td></td>
<td></td>
<td>improved</td>
</tr>
<tr>
<td>Semantic gap</td>
<td>Schema transformations</td>
<td>Semantic gap reduced in the ER model</td>
</tr>
<tr>
<td>Anomalies in relationship attributes</td>
<td>SVRA and MVRA proposed</td>
<td>Semantic anomalies resolved</td>
</tr>
<tr>
<td>After mapping, relations may not be in BCNF</td>
<td>Schema transformations</td>
<td>Relations satisfy BCNF</td>
</tr>
<tr>
<td>Hard to compute the quality of an ER model</td>
<td>- Completeness Index</td>
<td>“Equivalent” ER models can be compared quantitatively</td>
</tr>
<tr>
<td></td>
<td>- Normalization Index</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Correctness Metric</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Modifiability Metric</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Overall Quality Index</td>
<td></td>
</tr>
<tr>
<td>Fuzzy based quality Metrics</td>
<td>Fine-grained quality comparisons yielding improved ER models</td>
<td></td>
</tr>
</tbody>
</table>
10.2 Further Work

In our thesis, we have primarily considered functional dependencies for improvement in the conceptual model. Other dependencies, for instance, multi-valued dependencies should also be analyzed expecting that conceptual model can be improved in such a way that resulting logical schema is normalized up to 5NF.

Higher degree relationship types, for example, ternary relationships and the associated attributes may lead to some semantic problems. Our work can be extended to study such problems and propose solutions accordingly.

Proposed schema transformation rules can further be extended to incorporate other constructs of the extended entity-relationship (EER) model, for instance, hierarchies and categories.

We have concentrated on ER models in this research. Problems of conceptual modeling in other methodologies, for instance, UML or Aspect Oriented Development can also be studied in similar fashion.

Definitions of the quality metrics proposed in this thesis, for example, Normalization Index and Correctness, can further be improved using fuzzy based approach as we have demonstrated in the case of Completeness Index (CI).
The concept of CI and its definition can further be improved by incorporating other requirements of the conceptual model and the multi-valued dependencies related to the problem. We can define CI very comprehensively this way.

Identifying an approach to optimize the overall software quality should be investigated. For instance, it has been shown in the thesis that CI and NI are independent of each other. Therefore, QI can be optimized for a given set of weights for CI and NI, that is, research can be done on finding the values of CI and NI (given the weights) where QI gives the optimal value. This can help determine the point where schema transformations may not be applied any further.

With project management perspective, research can be done on estimating cost, effort, and implementation size of the product based upon the proposed quality metrics. The estimates will then be available at a very early stage of SDLC.
References

APCCM: Asian-Pacific Conferences on Conceptual Modeling. 
http://apccm.massey.ac.nz/apccm07/


Codd, E. (1972) Further normalization of the data base relational model. In Rustin


Codd, E. (1979) Extending the database relational model to capture more meaning. ACM Transactions on Database Systems, 4(4), pp. 397-434


Nicolas, J. (1978) Mutual Dependencies and Some Results on Undecomposable Relations, *VLDB*


TR CS001-2006: Technical Report (2006), Measuring Quality of a Conceptual Model through Fuzzy Completeness Index and Fuzzy Normalization Index, Department of Computer Science, LUMS, Lahore, Pakistan


Vincent, W. and Srinivasan, B. (1993) A Note on Relation Schemes which are in 3NF but not in BCNF. Information Processing Letters 48(6), December 1993, pp 281-283


Zadeh, L. (1965), Information and Control, pp. 335-353.
