

Semantic Extensions and a Novel Approach to Conceptual Modelling

by

Somluck La-Ongsri, *B.Sc. (Applied Mathematics), M.Sc. (Information Technology)*

School of Computer Science, Engineering and Mathematics,
Faculty of Science and Engineering

February 25, 2009

A thesis presented to the
Flinders University
in total fulfilment of the requirements for the degree of
Doctor of Philosophy

Adelaide, South Australia, 2009
© (Somluck La-Ongsri, 2009)

Contents

Contents	ii
List of Figures	viii
List of Tables	xii
Abstract	xiii
Certification	xiv
Acknowledgements	xv
1 Introduction	1
1.1 Background	2
1.1.1 The Design of Conceptual Models	2
1.1.2 A Historical Background and Perspectives of Conceptual Modelling	5
1.2 Motivation for Research	9
1.3 Objectives	12
1.4 Scope	13
1.5 Approach and Structure	13
1.5.1 Approach of the Thesis	13
1.5.2 Structure of the Thesis	17
2 Revisiting the Fundamentals of Conceptual Modelling Concepts	19
2.1 Introduction	19
2.2 Data Models	21
2.3 Conceptual Schema	24

2.4	Semantics in Conceptual Modelling	25
2.5	Conceptual Design as Part of Database Design	28
2.6	Key Considerations of a Conceptual Data Model	31
2.7	Conceptual Modelling Approaches	33
2.7.1	The Entity-Relationship (ER) Model	34
2.7.2	Object Role Modelling (ORM)	38
2.7.3	Unified Modelling Language (UML) Class Diagrams	43
2.7.4	A Comparison of ER, ORM and the UML Class Diagram	48
2.8	Evolution of Data Models	51
2.9	Summary	58
3	ER Modelling Extensions: A Survey and Comparative Review	59
3.1	Introduction	60
3.2	Limitations of the ER Model	63
3.3	Exploring ER Modelling Extensions	66
3.3.1	Structural Aspect	66
3.3.2	Data Abstraction Aspect	70
3.3.3	Temporal Aspect	72
3.3.4	Spatio-Temporal Aspect	75
3.3.5	Data Warehousing Aspect	79
3.3.6	Domain-Specific Application Aspect	81
3.3.7	Knowledge Base Aspect	83
3.3.8	Fuzzy Data Aspect	85
3.3.9	XML Data Aspect	88
3.4	A Comparative Study of ER Modelling Extensions	90
3.5	Summary	94
4	Mesodata in Conceptual Modelling	96
4.1	Introduction	97
4.2	Mesodata Approach	101
4.2.1	What is Mesodata?	102
4.2.2	Key Components of Mesodata	103
4.2.3	Example Use of Mesodata	105

4.3	The Mesodata Entity-Relationship (MDER) Model	106
4.3.1	MDER Data Structures	107
4.3.2	MDER Integrity Constraints	111
4.3.3	MDER Languages	112
4.4	The Mesodata Object Role Modelling (MDORM) Model	112
4.4.1	MDORM Data Structures	114
4.4.2	MDORM Integrity Constraints	114
4.4.3	MDORM Languages	115
4.5	Example MDER and MDORM Schemata	116
4.6	Summary	116
5	Ontology in Conceptual Modelling	119
5.1	Introduction	120
5.2	Review of Ontologies	123
5.2.1	What is Ontology?	124
5.2.2	Types of Ontology	127
5.2.3	Terminological Clarifications of Ontologies	130
5.2.4	Ontologies versus Conceptual Data Models	134
5.3	Ontology-Based Conceptual Data Modelling	136
5.3.1	Using Ontology for High-Level Conceptual Data Models	136
5.3.2	Ontology's Class Hierarchy Modelling	137
5.4	The Ontological Entity-Relationship (OntoER) Model	141
5.4.1	OntoER Data Structures	142
5.4.2	OntoER Constraints	144
5.4.3	OntoER Languages	145
5.5	The Ontological Object Role Modelling (OntoORM) Model	145
5.5.1	OntoORM Data Structures	146
5.5.2	OntoORM Integrity Constraints	147
5.5.3	OntoORM Languages	148
5.6	The Ontological Unified Modelling Language (OntoUML) Class Diagram Model	148
5.6.1	OntoUML Data Structures	150
5.6.2	OntoUML Integrity Constraints	150

5.6.3	OntoUML Languages	152
5.7	Example of OntoER, OntoORM and OntoUML Class Diagram Schemata	152
5.8	Summary	157
6	Polymorphic Relationships in Entity-Relationship Modelling	158
6.1	Introduction	159
6.2	Review of Relationship Types in Conceptual Modelling	161
6.2.1	Types of Relationship	161
6.2.2	Constraints on Relationship Types	163
6.3	Motivation	166
6.4	Links as Overloaded Polymorphic Relationships	169
6.4.1	Link Type Structure: Formal Definition	170
6.4.2	Link Type Representation	171
6.5	Cardinality, Default Value and Precedence for Link Types	173
6.5.1	Cardinality Constraints on Link Types	173
6.5.2	Default Values for Link Types	174
6.5.3	Precedence	176
6.6	Summary	177
7	Data Modelling in Rapidly Changing Complex Environments	179
7.1	Introduction	180
7.2	Significant Issues in Driving Rapid Conceptual Modelling Techniques	182
7.2.1	Systems Issues	182
7.2.2	Data Issues	184
7.3	Late Binding the Conceptual Modelling	185
7.3.1	Deferred Schema Deployment	185
7.3.2	Common Conceptual Schemata	186
7.4	LtER Modelling	188
7.4.1	The LtER Schema	188
7.4.2	The LtER Architecture	190
7.4.3	The LtER Characteristics	193
7.5	Summary	194

8 From Conceptual Design to Logical Design for the Relational Data Model	196
8.1 Introduction	197
8.2 The Relational Data Model	198
8.3 Transformation from the MDER Schema to a Relational Schema .	201
8.4 Transformation from the MDORM Schema to a Relational Schema	204
8.5 Transformation from the OntoER Schema to a Relational Schema	206
8.6 Transformation from the OntoORM schema to a Relational Schema	210
8.7 Transformation from the OntoUML Schema to a Relational Schema	213
8.8 Summary	216
9 Conclusions and Future Research	219
9.1 Research Contributions	219
9.1.1 A Survey on ER Modelling Extensions	219
9.1.2 Accommodating Mesodata into Conceptual Modelling Methodologies	220
9.1.3 Incorporating Ontology-based Semantics in Conceptual Modelling	220
9.1.4 Polymorphic Relationships in ER Modelling	221
9.1.5 Rapid Conceptual Modelling	222
9.1.6 Data Model Mapping	222
9.2 Future Research Directions	223
9.2.1 Extending This Work	223
9.2.2 New Areas to Explore: Advances in Conceptual Modelling	225
9.3 Conclusions	226
Appendices	228
A Publications Resulting from This Thesis	228
B Sample Proposals from the CERME Framework	229
B.1 The Enhanced Entity-Relationship (EER) Model	229
B.2 The Time Extended EER (TIMEER) Model	232
B.3 The Spatio-Temporal ER (STER) Model	236

B.4	The Modelling of Application Data with Spatio-temporal features (MADS)	240
B.5	The DIstributed design of SpaTIO-temporaL data (DISTIL)	243
B.6	The starER Model	244
B.7	The MultiDimER Model	245
B.8	The FuzzyEER Model	248
B.9	The ER extended for XML (EReX) Model	253
B.10	The XSEM-ER Model	255
C	The Full Schema Mapping for MDER and OntoER models	258
C.1	The Result of the Full Mapping of the INVENTORY MDER Schema	258
C.2	The Result of the Full Mapping of the MEDICAL OntoER Schema	258
C.3	Correcting the Schema: Normalisation	259
D	The Full SQL Data Definition Language (MDDL and DDL)	263
D.1	SQL Data Definition Statements for Defining the INVENTORY Schema	265
D.2	SQL Data Definition Statements for Defining the MEDICAL Schema	266
	Bibliography	269

List of Figures

1.1	Conceptual modelling issues during four eras.	6
1.2	Representation of common domain semantics	10
1.3	Overview of the thesis structure	16
2.1	Conceptual schema as the heart of an integrated system	25
2.2	Three main phases of database design.	29
2.3	An example of an ER diagram for a university database.	35
2.4	Cardinality ratio and participation constraint in the <code>WORKS_IN</code> and <code>MANAGES</code> relationship types.	36
2.5	Cardinality ratios in the ER model.	36
2.6	Two design options of Country	38
2.7	An example of an ORM diagram.	39
2.8	An example of a binary association with the expression of multiplicity constraints in a UML class diagram.	44
2.9	An example of a UML class diagram.	46
2.10	UML aggregation.	47
4.1	Examples of reference files	99
4.2	Mesodata layer between metadata and data	102
4.3	Symbols of major components in the <code>MDER</code> model.	107
4.4	A <code>COUNTRIES</code> mesodata entity type.	108
4.5	An example of a Suburb connectivity.	109
4.6	An example of a Suburb attribute referencing a weighted graph.	110
4.7	An <code>MDER</code> Example.	111
4.8	ORM Symbols Extension.	113
4.9	The <code>WGRAPH</code> value type and the mesodata mandatory role constraint.	114
4.10	<code>MDER</code> diagram for an example <code>INVENTORY</code> database.	115

4.11 MDORM schema for an example INVENTORY database.	117
5.1 A portion of a medical database	122
5.2 Different types of ontology.	127
5.3 Ontological categories	128
5.4 Different types of ontological specifications.	131
5.5 Ontology spectrum	132
5.6 Ontologies and the underlying conceptual schemata diagram.	137
5.7 An example of a representation of terms and relationships.	139
5.8 Symbols of major components in the OntoER model.	142
5.9 A DISEASES ontological entity type.	143
5.10 An OntoER Example.	144
5.11 An extension of the ORM Symbols	146
5.12 An ontological label type	147
5.13 A simple OntoORM diagram.	147
5.14 A representation of ontological class types.	150
5.15 A simple OntoUML class diagram.	151
5.16 OntoER schema for an example MEDICAL database.	153
5.17 A portion of the medical school location ontology.	154
5.18 OntoORM schema for an example MEDICAL database.	155
5.19 OntoUML class diagram for an example MEDICAL database.	156
6.1 Cardinality and participation constraints.	164
6.2 Another representation of cardinality and participation constraints.	165
6.3 Constraints of the ternary SUPPLY relationship.	166
6.4 Two design diagrams.	167
6.5 Duplication of Form and Dose attributes.	168
6.6 Three options for the modelling of meetings.	169
6.7 Examples of links representation.	172
6.8 A link type.	173
7.1 An example schema for a parts example.	185
7.2 LtER schema.	188
7.3 LtER architecture.	191

7.4	Two design choices.	193
8.1	An example of a relation <code>PROPERTY_FOR_RENT</code>	199
8.2	A referential integrity constraint.	200
8.3	MDER schema for an example <code>INVENTORY</code> database.	202
8.4	Mapping the mesodata entity types.	203
8.5	MDORM schema for an example <code>INVENTORY</code> database.	205
8.6	OntoER schema for an example <code>MEDICAL</code> database.	207
8.7	Corresponding source relational schemata for the ontological entity types.	208
8.8	OntoORM schema for an example <code>MEDICAL</code> database.	211
8.9	Mapping an <code>OMR</code> constraint in <code>OntoORM</code>	213
8.10	OntoUML schema for an example <code>MEDICAL</code> database.	214
B.1	Subclasses and specialisation in an EER diagram.	230
B.2	The temporal entity type <code>employee</code>	233
B.3	The temporal relationship type <code>WORKS_FOR</code>	233
B.4	The temporal attribute <code>Salary</code>	234
B.5	Representation of snapshot participation constraints in <code>TIMEER</code>	234
B.6	Representation of lifespan participation constraints in <code>TIMEER</code>	235
B.7	The temporal relationship type <code>WORKS_FOR</code> with participation constraints	235
B.8	Entity type capturing existence time.	237
B.9	Spatial entity types.	237
B.10	A Spatio-temporal entity type with valid time support.	238
B.11	An attribute with valid time support.	238
B.12	A spatial attribute.	238
B.13	A spatio-temporal attribute in <code>STER</code>	239
B.14	A spatio-temporal relationship type.	239
B.15	An example of a multi-association relationship type.	241
B.16	An example of a starER diagram.	244
B.17	Examples of specialisation, aggregation and membership relationship types.	245
B.18	Notations for a multidimensional model.	246

B.19	A conceptual multidimensional schema.	247
B.20	Entity employee with fuzzy attributes.	249
B.21	Possibility distributions for the fuzzy attribute.	250
B.22	Fuzzy entity with a membership degree.	250
B.23	An example of a fuzzy relationship.	251
B.24	An example of a fuzzy participation constraint.	251
B.25	An example of a fuzzy cardinality constraint.	252
B.26	An example of fuzzy (min,max) notation.	252
B.27	An example of an EReX schema.	253
B.28	XSEM-ER data node types and cluster types.	256
C.1	Result of mapping the INVENTORY MDER schema.	259
C.2	Result of mapping the MEDICAL OntoER schema.	260
C.3	Functional dependencies on schemata.	261
C.4	Normalisation into 3NF.	261
C.5	A normalised schema of the MEDICAL OntoER schema.	262

List of Tables

2.1	Examples of Multiplicities.	45
2.2	Equivalent data structures in ER, ORM and the UML class diagram.	49
2.3	Equivalent integrity constraints in ER, ORM and the UML class diagram.	50
2.4	Equivalent languages in ER, ORM and the UML class diagram.	50
2.5	The data models in ten historical eras.	51
3.1	Overview presentation of the CERME survey framework.	62
3.2	A comparison of the main CERME proposals.	92
3.3	Resources available for each CERME aspect.	93
4.1	Examples of mesodata types and their operators.	103
4.2	Examples of mesodata types and the structure of source relations.	104
4.3	A relation zipcoderel.	105
5.1	Corresponding use of terms between semantic domain relationships and common domain structures	140
B.1	Assigning temporal aspects to ER constructs.	236

Abstract

Conceptual modelling presented as a framework for database design is a discipline of great importance in many areas in computer science that seeks to represent real-world phenomenon using semantic primitives. To date, traditional (static) conceptual models have been successfully used and extended to deal with the semantics of relatively stable real world applications. However, the capturing of semantics is a seemingly endless task as it involves various dimensions and categories.

It is argued in this thesis that the incorporation of complex domain structures in conceptual modelling to represent the semantic domains of an attribute and the relationships within a concept in ontologies would provide more expressive and richer semantics. Additionally, it is argued that basic relationships in the entity-relationship model may need to be modified or extended to handle a broad spectrum of situations that arise from differing perspectives of the real world. Furthermore, it is argued that a conceptual model should allow rapid and simultaneous storage of data and data modelling as unexpected and sudden events require data to be modelled rapidly. This thesis begins with an extensive review of the field of conceptual modelling and an exploration of the concepts of mesodata, ontologies and semantic relationships in conceptual modelling as well as various aspects of extensions to the entity-relationship model.

Using these foundations, a classification of ER modelling extensions (CERME) framework is introduced that forms the basis of common aspects and comparative criteria which can be used to categorise and compare various proposals. In addition, the Mesodata Entity-Relationship (MDER) model, Mesodata Object Role Model (MDORM), Ontological Entity-Relationship (OntoER) model, Ontological Object Role Modelling (OntoORM) and Ontological Unified Modeling Language (OntoUML) class diagrams are presented that allow advanced semantics to be associated with the domains of an attribute. It is also demonstrated that these proposed models can be mapped into the commonly accepted standard relational model. Furthermore, for some of the modelling issues that are not easily accommodated into the ER model, this thesis introduces a new relationship construct, **polymorphic relationships**, to handle this situation. To this end, a novel approach to conceptual modelling, the **LitER** model, is presented that incorporates the previously proposed concepts of mesodata, ontologies and polymorphic relationships into the model which allows data to be modelled rapidly.

Certification

I certify that this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text.

Signed

Dated

Somluck La-Ongsri

Acknowledgements

My PhD journey and this thesis would not have been accomplished without the support of other colleagues and friends. I would like to express my gratitude to those who have helped me along my journey:

— my supervisor, Professor John F. Roddick, for his guidance on the ideas and concepts that provided me with inspiration for my thesis and for his support and encouragement throughout my candidature;

— my teacher, Associate Professor Suphamit Chittayasothorn, for his teaching, guidance and support that provided me with inspiration for my academic career and PhD research and for his constructive feedback on my work;

— my sponsor, Sukhothai Thammathirat Open University, Thailand, for providing the scholarship that gave me the opportunity to pursue a PhD abroad;

— Flinders fellows: Carl Mooney for his assistance with \LaTeX and proofreading, Denise de Vries and Paul Gardner-Stephen for their helpful suggestions and proofreading, Murk Bottema for his assistance, helpful suggestions and reading parts of the drafts, Paul Calder who always made himself available to students and for reading parts of the drafts, Rino Calaycay who is friendly and always ready to help, Neville Williams for his warmth and friendliness, Martin Luerssen for providing his \LaTeX sources and lastly, Edi Winarko, May Zhao, Kenny Ma, Huan-Min Shen, Richard Leibbrandt, Anna Shillabeer and Ping Liang for sharing their experiences;

— Flinders loop bus driver, David Petch, for his caring and generosity; Security officers for assisting me to get home safely after long nights of study; The Document Services Unit for delivering research materials; The Staff Development and Training Unit for their useful training courses and programs;

— my best friend, Steppy, for his continual support in reviewing my manuscript and in bringing me happiness in my last tough year; my close friends (PTong, PPai, Synraha's family and relatives, Dao and PJuk) for their wonderful friendship and support; and finally

— my family for providing the support that only a family can and for their understanding, encouragement and confidence in me; my heartfelt thanks goes to Mum and Dad who will be proud to see me finally achieve my PhD and my supportive sister and brother, PPen and PPat.

Chapter 1

Introduction

Research on conceptual modelling suggests that more meaningful information about application data should be captured in the data model (Badia, 2004; Codd, 1979; Hull and King, 1987; Tu and Wang, 1993). However, this capture of the semantics is a seemingly endless task because it involves various dimensions and categories (Badia, 2004; Tu and Wang, 1993). Moreover, the semantics of complex attribute domains are not explicitly represented in conceptual models.

Prior research into data modelling has considered concepts such as the complex domains of an attribute (de Vries and Roddick, 2004) and the use of conceptual models in supporting ontologies (Evermann and Wand, 2005; Jarrar, Demey and Meersman, 2003; Purao and Storey, 2005; Spaccapietra, Parent, Vangenot and Cullot, 2004; Sugumaran and Storey, 2002, 2006; Storey, 2005; Wand, Storey and Weber, 1999). However, these studies have not fully examined how the semantics of the complex domains of an attribute can be integrated into a theoretical framework of conceptual models as a single integrated schema.

As has been suggested in interviews with Peter Chen (Winslett, 2004), the future directions and applications of ER databases are more likely to be focused on providing answers to the high-level matching of concepts and the identification of complex relationships. The boundary of this focus will not be as easily definable as in the past, as the questions being asked will cover broader and more intangible questions such as those related to national security and other open-ended questions relating to risk identification and mitigation. Consequently, exploring enhancements and extensions to conceptual models still continues to be a legitimate and important research area (Badia, 2004). These are the driving forces for exploring the ideas in this thesis.

This chapter provides the foundations of the presented thesis, beginning with

the background of the thesis (Section 1.1) and the motivation behind the relevance of the work (Section 1.2). Next, the objectives of the research (Section 1.3) and its scope (Section 1.4) is discussed. The chapter concludes by presenting the approach that will be followed to accomplish these objectives together with an overview of the thesis structure (Section 1.5).

1.1 Background

Conceptual models attempt to represent the clear semantics of an application environment of the real world so as to make the database schema generally understandable, useful and adaptable. Thus, instead of having modelling constructs based on the concept of the logical schemata (as represented in the relational, hierarchical and network models) (King and McLeod, 1985), conceptual models attempt to represent data structures in a natural way by using graphics to represent the entities (or objects) in the real world. For example, the Entity-Relationship (ER) model (Chen, 1976) represents the data structures as *entity types*, which are a group of entities, *relationship types*, which are a set of relationships as associations among entity types, and *attributes* which characterise the properties of entities and relationships.

The following topics are now briefly introduced to provide the foundation and motivation for the contributions of this thesis.

1.1.1 The Design of Conceptual Models

“The design of conceptual models is the most difficult stage in data model development to learn (and to teach). There is no mechanical transformation from requirements to candidate solutions.”

Graeme Simsion and Graham Witt (2005)

Every database can be defined by a data model. A data model is an abstract description of reality according to certain conceptualisations, which cannot be produced by a mechanical transformation. Conceptual data modelling is the central activity in data modelling, providing a high level of abstraction in describing the requirements of real world applications, aimed at achieving independence from implementation issues (Simsion and Witt, 2005). Conceptual modelling is widely recognised to be a necessary foundation for building a database that satisfies the

user requirements. It relies on a graphical notation that facilitates understanding and management of conceptual schemata by both designers and users (Rizzi, Abelló, Lechtenböcker and Trujillo, 2006).

Database design covers three main phases: conceptual design, logical design and physical design. Often, there is some confusion between the terms *conceptual* and *logical* when describing different data models. Similarly, conceptual schemata are also ambiguously referred to as logical schema. The two notions of conceptual and logical references are different and distinguishable as described below.

Conceptual design. A conceptual design aims to derive an implementation-independent and expressive conceptual schema, starting from the user requirements. In this phase, the database requirements are analysed and modelled using conceptual data models. A main constituent of the context in which user/application requirements specification are modelled is related to the *universe of discourse (UoD)* or *subject domain* (Simsion and Witt, 2005). For example, a medical database specification of requirements refers to concepts in a UoD comprising entities such as patients, physicians, treatments and so on. Conceptual models typically represent the UoD as a collection of objects/entities (entity types). Entities are typically associated with each other via relationships and are classified according to types (classes) and subtypes (subclasses).

Conceptual modelling is the central activity in data modelling and serves as a framework for database design and information system development. It represents real-world phenomena using semantic primitives (Chen, Song and Zhu, 2007). The ER model (Chen, 1976) is the fundamental principle for conceptual modelling.

Logical design. A logical design takes the conceptual schema and creates a corresponding logical schema for the chosen platform by considering a certain set of constraints. In other words, the next step after a creation of a conceptual schema is to translate it into a logical data model suitable for implementation using the target database management system (DBMS). The most well-known and extensively used logical model is the relational model. Logical design, in particular referring to relational models, can be characterised by two main methodologies as follows.

- 1. Transformations (or Mappings).** A mapping is a process of transforming results (including requests) between schemata (Elmasri and

Navathe, 2007). Conceptual schemata are only a *description* of data and do not include implementation details. This logical design step focuses on the actual implementation of the database using a commercial DBMS. Most current commercial DBMSs use the relational data model, so the conceptual schema is transformed from the high-level data model into the relational model. This step is called *data model mapping* or *logical design*. Its results are the relational schemata that serve as the framework for the implementation on the chosen platform. As the relational model is the implementation model of choice, a transformation of conceptual schemata into relational schemata is also discussed in this thesis. This includes procedures to create a relational schema from the proposed modelling constructs, e.g. the mesodata entity types and the total domain participation constraints in the Mesodata Entity-Relationship (MDER) schema. This thesis relates the modelling constructs presented in Chapters 4 and 5 to the constructs of the relational model as presented in Chapter 8.

- 2. Normalisation.** Functional dependencies (FDs) are derived from the relationships between attributes. The concept of FDs are not taken into consideration in conceptual modelling techniques such as the ER model. There are two main types of FDs: (1) those that represent the dependencies among non-key attributes; and (2) those that represent dependencies of attributes which depend on only partial keys of tables (A primary key of the table in this case is a composite key that uses several attributes). The relational tables associated with all FDs are normalised, i.e. tables are decomposed or split into smaller tables using a standard methodology called *normalisation*, which is performed in the logical design phase.

Normalisation aims to restructure database schemata through decomposition in such a way that functional and join dependencies between attributes are extracted from the real world semantics. An example of this normalisation technique for the example MEDICAL database is discussed in Appendix C.

Note that database tool vendors sometimes use the term *logical model* to refer to the conceptual data model, and use the term *physical model* to refer to the DBMS-specific implementation model (Teorey, Lightstone and Nadeau, 2006). Note also that conceptual data models can not only be built from scratch, but

can also be generated through a process of *reverse engineering* from an existing DBMS-specific schema (Silberschatz, Korth and Sudarshan, 2005).

Generally, when people first develop an application, they use ER, ORM or UML diagrams for data modelling, followed by an alternate technique, such as Data Flow Diagrams (DFD) for functional modelling¹. ER models are conceptually oriented and are only a design methodology, as there are no commercial ER database management systems (ER DBMSs) currently available. The logical design step is necessary in order to allow the mapping of the ER diagrams to the relational model which is more implementation oriented. For this reason, this thesis also covers this aspect of logical design as it relates to ER modelling.

Designing a conceptual data model involves conceptualisation, abstraction and other skills that are difficult to use on a day-to-day basis without considerable practice (Simsion and Witt, 2005). Common experience tends to indicate that even database design experts who have had many years experience in dealing with the theories of conceptual modelling still struggle when faced with the scenario of designing data models for challenging real world applications.

1.1.2 A Historical Background and Perspectives of Conceptual Modelling

Conceptual modelling has always been one of the cornerstones for information systems as it describes the general knowledge of the system in the so-called conceptual schema (Krogstie, Opdahl and Brinkkemper, 2007). The evolution of research and practice in the area of conceptual modelling during the past four decades as discussed by Bubenko jr (2007) is depicted in Figure 1.1, and discussed below:

- 1. Modelling research issues in the seventies.** This era was characterised by the introduction of new models as well as the refinement and extensions of a number of existing data modelling methodologies. The most enthusiastic data modelling researchers at this time came from the database community. Some notable activities during the seventies were as follows:

- In 1975, the Standards Planning and Requirements Committee (SPARC) of the American National Standards Institute Information Processing

¹Functional modelling is a different modelling approach focussing on describing the processes rather than entities and relationships between entities. A well-known example of this approach is Data Flow Diagrams.

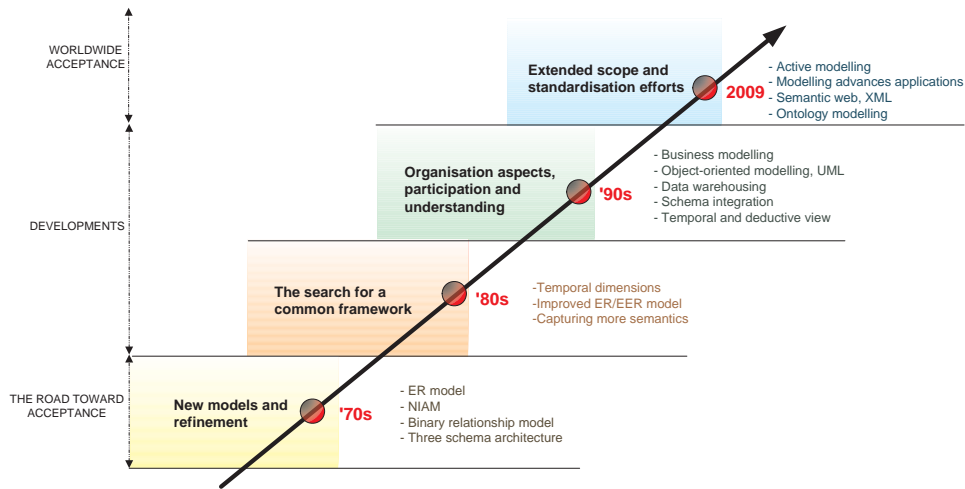


Figure 1.1: Conceptual modelling issues during four eras.

Systems (ANSI/X3) Committee proposed the *three-schema architecture* (the external schema, conceptual schema and internal schema) for describing data in a database, in addition to the concept of data independence:

- the external schema describes the views of different user groups.
 - the conceptual schema describes the structures, which is a high-level description of the whole database focussing on entities, data types, relationships, user operations and constraints, but ignores any physical storage structures.
 - the internal schema describes the physical storage structures of the database.
- Substantial research was carried out to provide high-level semantics for modelling information systems, such as the binary relationship model (Abrial, 1974; Senko, 1975), the ER model (Chen, 1976) and Nijssen’s Information Analysis Methodology (NIAM) (Falkenberg, 1976; Nijssen, 1976, 1977).

In summary, the essential basic concepts of modelling were invented and presented during the seventies. Not all of the presented approaches became practicable, but they formed a solid platform for further developments during the eighties and nineties. This era is commonly referred to as “the road toward acceptance” (Chen, Wong, Delcambre, Akoka, Sølvsberg and Liuzzi, 2008).

2. Modelling research issues in the eighties. Modelling approaches in this era focused on the search for a common modelling and methodology framework. As a large number of more or less similar modelling methodologies and concepts were published during the seventies, it is apparent that this era was marked by a common desire to compare the different models and to try to find a common acceptable framework. Some important developments during the eighties included:

- improving the expressive power of semantic data models, and adding the temporal dimension;
- capturing more real-world semantics in an improved ER/EER model; and
- increased understanding of existing methods and tools that consequently led to their improvement.

Several other semantically rich and expressive modelling approaches were introduced during the eighties, e.g. the Requirement Modeling Language (RML) (Greenspan, Borgida and Mylopoulos, 1986) that focused on requirement engineering as a significant phase in the system development life cycle. An interest was also expressed in the topic of historical databases, which gave rise to data models that treated several aspects of time, e.g. checking valid time and the transaction time of an event. This database concept was called temporal databases. A substantial number of temporal extensions are reported in the literature (Ferg, 1985; Klopprogge, 1981; Klopprogge and Lockeman, 1983; Snodgrass, 1987).

3. Modelling research issues in the nineties. Modelling approaches in this era focused on organisational aspects, stakeholder participation and mechanisms to improve understanding of the various models. This illustrated the movement for extending the scope of modelling to business or enterprise modelling which included, for example, work practice impacts, managing database relationship changes and user needs. The widening scope during the nineties included:

- increased understanding and support of work activities at all levels in an organisation;
- interoperable systems and semantic heterogeneity; and
- enterprise modelling to support user and stakeholder participation in enterprise analysis and requirements.

These have led to a marked increase in the development and use of advanced conceptual modelling methodologies and techniques throughout the database industry. This has not only impacted the development of new information system architectures and DBMS, but has brought many benefits to a wide range of organisations in different application domains such as medicine, defence and transportation.

Approaches based on temporal, spatio-temporal and deductive views of application domains, as well as object-oriented modelling are attributable to the nineties. Substantial research of this era focused on temporally enhanced ER models (Elmasri, El-Assal and Kouramajian, 1990; Elmasri, Wu and Kouramajian, 1993; Lai, Kuiboer and Guynes, 1994; Tauzovich, 1991; Theodoulidis, Loucopoulos and Wangler, 1991*a,b*; Zimanyi, Parent, Spaccapietra and Pirotte, 1997) and spatio-temporal ER models (Tryfona and Jensen, 1999; Parent, Spaccapietra and Zimányi, 1999). The Unified Modeling Language (UML) (Booch, Rumbaugh and Jacobson, 1999; Muller, 1999) was one of the most well known object-oriented modelling approaches that was developed in the nineties.

Just as in the eighties, research in the nineties was focused on seeking to refine and rationalise the theory behind conceptual modelling (Chen et al., 2008).

4. Modelling research issues in the new millennium. Modelling approaches in this era have focused on extended scope and standardisation efforts. To satisfy the need for better understanding and a shared conceptual view of different domains, the concept of *ontologies* became increasingly popular and gained widespread use in various disciplines. In addition, increasing changes in the real world has demanded a shift in conceptualisation and a new way of viewing reality using evolving knowledge (Chen et al., 2008). Some important issues of this era include:

- a focus on the web and ontological perspectives of conceptual modelling;
- development of a single dynamic conceptual model through multi-perspective knowledge and technology integration within a new framework of active paradigm; and
- development of database systems and technologies that can benefit from active conceptual modelling.

The present status of modelling is that its theory and practical applications have gained sufficient maturity for its worldwide acceptance in the IT community (Chen et al., 2008).

1.2 Motivation for Research

As discussed by Berild (2004), conceptual models are traditionally used to capture the meaning and structure of the information to be managed in database applications. In this respect, the conceptual model also acts as input for the generation of a specific database schema conformant to some chosen implementation technology.

A number of recent research efforts on conceptual modelling have investigated the use of natural relationships within a body of information using conceptual models in supporting ontologies. Examples include a conceptual markup language within the ontology-engineering framework based on an ORM schema (Jarrar et al., 2003), a methodology for creating or evaluating ER models using domain ontologies (Sugumaran and Storey, 2006, 2002), an ontology for classifying the verb phases of relationships (Storey, 2005; Puroo and Storey, 2005), a formal analysis of the meaning of the relationship construct (Wand et al., 1999), ontologically based semantics for object-oriented constructs using UML (Evermann and Wand, 2005) and an analysis of some arguments concerning whether conceptual data models can adequately support the design and use of ontologies (Spaccapietra et al., 2004).

As suggested by Guizzardi (2005), the world view that is represented by conceptual modelling can not be considered as an adequate conceptualisation of reality. As a consequence, it falls short of offering its users suitable sets of modelling concepts for constructing an explicit representation of their knowledge of the domain. Within the database design community, there has been a change in the application of databases away from the data and information specific level to a domain specific level that focuses on semantic interaction. This thesis argues that semantics can be completely represented by modelling structures, and the focus of conceptual modelling should be on the adequacy and expressiveness of the representation structures.

Having a precise representation of a given conceptualisation becomes more important where advanced semantics are integrated into the data model. Consider the role of ontologies in modelling the complex domains of an attribute.

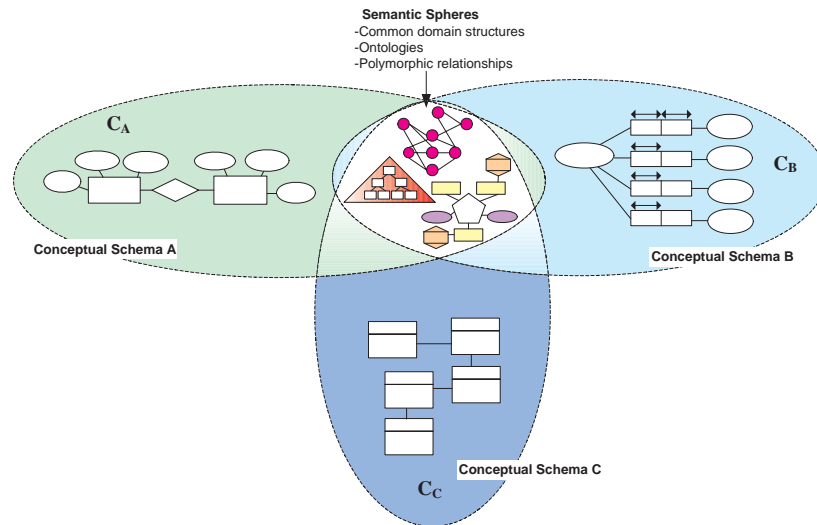


Figure 1.2: A representation of common domain semantics between three different conceptual modelling techniques.

Within this context, ontologies are referred to as a semantic representation of the relationships between concepts/terms within the domain. In order for these semantics to be represented precisely, the modelling of domain semantics for the real world attributes must be explicitly represented in the conceptual models.

Consider the situation depicted in Figure 1.2, where C_A , C_B and C_C represent the conceptual schema of the same application created by different models. These models can be equivalent. C_A , C_B and C_C are modelled based on, for example, the ER, ORM and UML modelling techniques which reference the semantic sphere that includes common domain structures providing a shared set of semantics. C_A , C_B and C_C could share the semantic spheres that provide their ontologies and other semantics. For this situation, common domain structures describing real world phenomena have not been fully investigated, and limited research has been made into developing and promoting unified common domain structures for the data modelling community. Extensions to the ER model can facilitate such a solution as it is a powerful conceptual design tool. It can be used to consider different conceptual models at the same time for different user groups, and then map these models to each other (Chen, Thalheim and Wong, 1999).

Since conceptual modelling is meant to be used by the wider technology community, its semantic expressiveness and comprehensibility plays a fundamental role. Thus, modelling methodologies should be sufficiently expressive to suitably characterise the conceptualisation of its domain. The development of common

domain structures involves the expansion of the semantics of each technique that will improve its expressiveness and comprehensibility thus allowing for designers to easily recognise what each modeling construct means in terms of domain concepts or any other concepts involved. This thesis is motivated by these requirements in trying to *incorporate more data semantics and explicitly represent them in the data models*. This thesis argues that this can be achieved through the incorporation of mesodata, ontologies and links as polymorphic relationships into conceptual modelling approaches that leads to the development of the rapid data modelling approach that is presented in Chapter 7. The practicability of the suggested models is also presented in Chapter 8.

As applications and user needs of database systems have grown in complexity over the decades, new demands have arisen that require event traceability in order to deal with the changing world state. This new insight gained from the evaluation of the relationships between events that cause these changes may provide a significant impact on the understanding of the current world state (Chen, 2006). Recent dramatic incidents (e.g. the devastating cyclone Nargis in Burma, the earthquake disaster in China, the tsunami in Southeast Asia, the current global financial market crash and the September 11 attack on the World Trade Centers) require changes in conceptual modelling from a static conceptual model to an *active conceptual model*² (Chen et al., 2008). This is a reflective approach where the facts, activities and trends that were precursors to these past incidents are examined with the aim of creating reference points that can be used to assess current events, circumstances, and behaviours. This threat and risk assessment technique can be used to try and predict similar events in the future. This serves to encourage the establishment of socio-economic and emergency response programs to cope with such situations. To deal with these issues on a timely and adaptive basis requires active and rapid conceptual modelling techniques to analyse these situations to assess appropriate responses. This is another motivation for this thesis, namely to *represent a different approach to conceptual modelling, allowing data to be modelled rapidly*.

In summary, it is argued in this thesis that the expressiveness of conceptual modelling methodologies can be enhanced through (a) the semantics of complex domain structures for attribute domains, (b) the semantic relationships between concepts in ontologies, and (c) the flexibility of dealing with existing relationships.

²Chen (2006) has defined active conceptual modelling as a continual process of describing all aspects of the open world, its activities, and its changes under different perspective, based on our knowledge and understanding.

These aspects should be explicitly incorporated into the conceptualisations of their underlying subject domains. In addition, since unexpected and sudden events requires data to be modelled rapidly, a conceptual model should easily accommodate new data to reflect the rapidly changing complex environment. Therefore, the major research questions are:

- What extensions to the ER model have been previously described in the body of research literature?
- How can mesodata be modelled in conceptual modelling methodologies?
- How can ontologies be modelled in conceptual modelling methodologies?
- How can the relatively restricted and static modelling of relationships be modified to handle a broad spectrum of situations, or is there a more feasible approach to model overloaded relationships?
- How can these presented modelling constructs be related to a (re)design of conceptual modelling approaches to facilitate the rapid exploration of data?
- How can the presented modelling constructs be mapped into the relational database schema?

The answers to these questions are addressed through the 4 objectives that are detailed in the next section.

1.3 Objectives

The purpose of this thesis is to promote better practices in conceptual modelling. Specifically, the thesis introduces the semantic extensions of **mesodata**, **ontologies** and **polymorphic relationships** to traditional conceptual models. In addition, this thesis presents a unique and innovative modelling approach that accommodates changes in situations where data needs to be modelled rapidly. Furthermore, data model mapping steps are discussed to show how to design a relational database schema based on a conceptual schema design. This thesis also discusses a survey of various extensions to the ER model including the fundamentals of conceptual modelling concepts. In summary, the objectives of this thesis are:

1. to review the field of conceptual modelling, ontologies and the emerging discipline of mesodata to address existing conceptual modelling problems and to survey extensions to the ER model;
2. to extend conceptual data models to enhance expressiveness;
3. to establish a conceptual modelling approach that can support the modelling of data in rapidly changing environments
4. to illustrate the mapping procedures by creating a relational schema from a schema of the proposed conceptual models.

1.4 Scope

The central theme of this thesis is on conceptual modelling methodologies aimed at supporting database designers and users in explicitly modelling the semantics. The overall goal is to represent more semantics of the real world in the database schema. The work presented here is expressed using the concepts provided by the high-level conceptual data model and concentrates on entities, data types, relationships, user operations and constraints.

As the focus is on the conceptual level, physical-level and external-level aspects all fall outside the scope of this thesis. These aspects that are excluded from consideration include implementation issues, such as the development of a proof-of-concept system that will demonstrate these techniques or promote further development of the ideas for possible commercialisation, and the external issues such as the screen forms used for data entry.

1.5 Approach and Structure

The approach and structure of this thesis reflect the successive elaboration of the objectives specified in Section 1.3.

1.5.1 Approach of the Thesis

The approach followed in this thesis is to accomplish each of the four objectives is detailed as follows:

Objective 1: *To review the field of conceptual modelling, ontologies and the emerging discipline of mesodata to address existing conceptual modelling problems and to survey extensions to the ER model.*

This objective is primarily accomplished in Chapters 2, 3 and parts of Chapters 4, 5 and 6 of this thesis. It begins with a review of conceptual modelling concepts and methodologies. In Chapter 3, a comprehensive survey and consolidated overview of extensions to the ER model is presented. In this survey, various aspects of the ER extensions are analysed, salient points of the extended ER models are described, and a comparison of the models is discussed. Lastly, the concepts of mesodata, ontologies and relationships in conceptual modelling are discussed in Chapters 4, 5 and 6, respectively.

Objective 2: *To extend conceptual data models to enhance expressiveness.*

The accomplishment of this objective constitutes the core of this thesis as detailed below:

1. In Chapter 4, the concept of mesodata is incorporated into conceptual modelling methodologies based on the ER and ORM model. In that chapter it is argued that conceptual modelling methodologies would be semantically richer if they were able to express the semantics of complex data types for attribute domains. It starts with a systematic examination of how the concept of mesodata provides advanced semantics that can be associated with the domain of an attribute. The main research problem which is addressed in Chapter 4 is: *how can mesodata be modelled in conceptual modelling methodologies?* The chapter thus presents the Mesodata Entity-Relationship (MDER) model and Mesodata Object Role Modelling (MDORM) to solve this research problem.
2. In Chapter 5, the thesis demonstrates how ontologies can be incorporated into conceptual modelling methodologies. Besides extensions to the ER model and the ORM, this chapter includes a discussion of UML class diagrams. In that chapter it is argued that the conceptual model should support and express relevant aspects of the underlying domain associated with the world view. The Ontological Entity-Relationship (OntoER), Ontological Object Role Modelling (OntoORM) and Ontological Unified Modelling Language (OntoUML) class diagram models are thus presented to illustrate how the two concepts of mesodata and ontologies can be merged in order to

support a richer level of semantics. This shows how conceptual models can be enhanced with the purpose of improving domain semantics, through the use of common domain structures that facilitate more advanced semantic relationships between concepts/terms.

3. In Chapter 6, a new relationship construct, termed **polymorphic relationships**, is presented in the ER model. It is argued that existing relationship constructs of traditional conceptual models may need to be modified or extended to handle a broader spectrum of situations due to differing world views. It is shown how some of the modelling issues that are not easily accommodated in conceptual modelling (i.e. the situation that requires a treatment of default values in relationships where the values provided hold *unless more specific information is available*) can be handled in an intuitive manner. This research investigates the *links* between entities defined for polymorphic (overloaded) relationships. This allows the conceptual models to reflect situation changes in the real world and to continue providing a sound basis for database design.

Extensions proposed in this thesis attempt to embed the semantics of an application environment into the practice of conceptual modelling. This will improve the semantic expressiveness of conceptual data models and database schemata, and will assist in the evolution of semantically enriched conceptual schemata.

Objective 3: *To establish a conceptual modelling approach that can support the modelling of data in rapidly changing environments.*

In Chapter 7, a new conceptual modelling approach, the **LtER** model (Roddick, Ceglar, de Vries and La-Ongsri, 2008), is presented which allows rapid data modelling. This methodology allows data to be stored immediately and a more refined conceptual schema to be developed later. It is argued that a common conceptual schema should exist for the initial storage of data in the absence of a more specialised model and that it should be flexible enough to capture all data, and simple enough to create a full conceptual model. The adoption of this novel modelling approach will also allow for the integration of ontologies and mesodata combining emerging technologies such as data mining and knowledge based technologies including hypotheses, probabilistic reasoning and temporal auditing.

The contribution towards this objective provides further support for the practical utility of the work presented in the preceding chapters of this thesis (i.e.

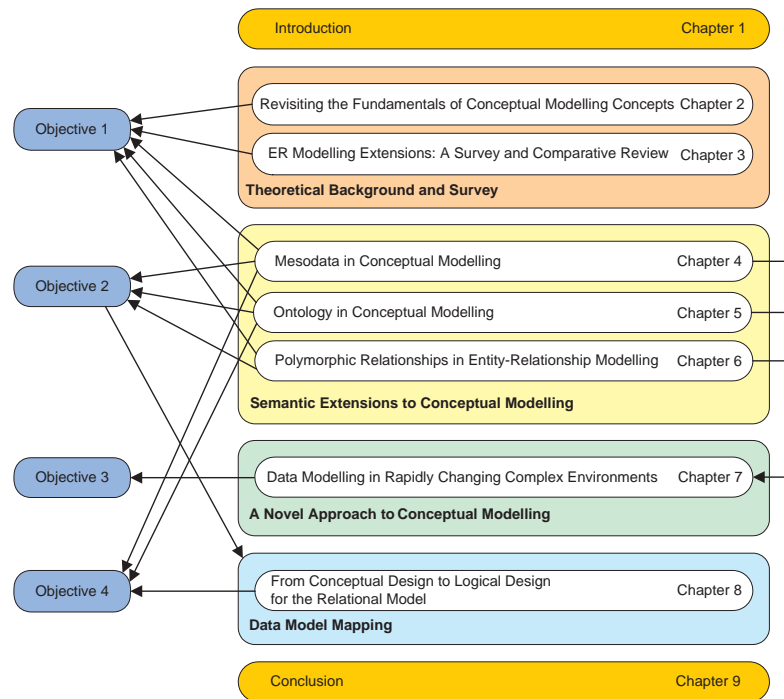


Figure 1.3: Overview of the thesis structure relating the objectives of the thesis with the chapters in which they are accomplished.

Chapters 4, 5 and 6).

Objective 4: *To illustrate the mapping procedures by creating a relational schema from a schema of the proposed conceptual models.*

In Chapter 8, the results of Objective 2 are used to show how the proposed constructs from the MDER, MDORM, OntoER, OntoORM or OntoUML class diagram schemata can be mapped to a relational database schema. This chapter argues that the ability to transform a conceptual schema to a relational schema that can be manipulated by the system is essential. It then describes how to design a relational database schema based on a conceptual schema design. These instructions include additional steps that are added to the existing mapping algorithm, and this transformation is illustrated with examples from the INVENTORY and MEDICAL databases.

These ideas further illustrate the assertion that these modelling techniques are useful and pragmatic.

1.5.2 Structure of the Thesis

Between this introduction and the conclusion, this thesis contains seven chapters which contain important research contributions. An overview of the structure of this thesis is presented in Figure 1.3, and is summarised below:

Chapter 1 provides the foundations of the presented thesis.

Chapter 2 comprises an overview of fundamentals of conceptual modelling concepts.

Chapter 3 presents a comprehensive survey and consolidated overview of ER model extensions.

Chapter 4 extends the ER model and ORM to show how mesodata can be accommodated into these modelling approaches, culminating in the Mesodata Entity-Relationship (MDER) model and Mesodata Object Role Model (MDORM). A review and further discussion of mesodata is included.

Chapter 5 extends three conceptual modelling methodologies, the ER model, ORM and UML class diagrams, to show how the concept of common domain structures can be used to accommodate ontologies, resulting in the Ontological Entity-Relationship (OntoER) model, Ontological Object Role Modelling (OntoORM) and Ontological Unified Modelling Language (OntoUML) class diagrams. This chapter also includes a comprehensive review of the concept of ontologies.

Chapter 6 introduces the concept of *polymorphic (i.e. overloaded) relationships* in ER modelling and shows how the static modelling of relationships can be modified to accommodate the natural semantics of applications. This allows for a conceptualisation of applications where a specific attribute value is not necessary for the relationship being described to be modelled in a conceptual model. This chapter also reviews the concept of (static) relationships in conceptual modelling.

Chapter 7 presents a new conceptual modelling approach that allows data to be modelled in rapidly changing environments. This research incorporates the concepts of mesodata, ontologies and links as polymorphic relationships

developed in Chapters 4, 5 and 6 into the the Low Instance-to-Entity Ratio (LtER) modeling approach. Their structures are presented as parts of the components of the LtER schema and architecture.

Chapter 8 presents a mapping algorithm that converts the proposed constructs of the MDER, MDORM, OntoER, OntoORM and OntoUML schemata into logical schemata that can be optimised and implemented in relational database systems as illustrated through the two major database examples of this thesis.

Chapter 9 summarises the contributions of this thesis and outlines areas for future research.

Appendix A provides references to relevant publications.

Appendix B provides the sample proposals from the classification of ER modelling extensions (CERME) framework that have been described according to key considerations of a conceptual data model.

Appendices C and D provide the schema mappings and data definition languages of the two major examples from the INVENTORY and MEDICAL databases in this thesis, respectively.

Chapter 2

Revisiting the Fundamentals of Conceptual Modelling Concepts

In the early days of conceptual modelling when data models were first proposed and the relational model had just conquered the database technologies, numerous data models had emerged with the aim of providing more expressive power for capturing the semantics of applications. These models, called *conceptual models*, have been used to capture the meaning and structure of the information to be managed in database applications for more than three decades. This chapter revisits the theoretical background of conceptual modelling concepts and presents a summary of the various data models proposed in each of the different eras.

The structure of this chapter is as follows. The foundations of conceptual modelling concepts is given in Sections 2.1 to 2.7. These includes an overall introduction (Section 2.1), a review of data model terminologies (Section 2.2), conceptual schema (Section 2.3), semantics (Section 2.4), conceptual design (Section 2.5), key considerations of a conceptual data model (Section 2.6), and three well-known conceptual data models (the ER, ORM and UML class diagram models) including their comparison (Section 2.7). Next, a summary of the evolution of data models is presented in Section 2.8. Finally, this chapter is summarised in Section 2.9.

2.1 Introduction

Database management is “all about mapping the *informal* real world into some *formal* machine representation” (Date and Darwen, 1992). The traditional database

community's dissatisfaction with its early systems was due to unwanted dependencies embedded in applications (Date and Darwen, 1992). Applications often relied on detailed information about the physical layout of data records and this made it impossible to evolve the database, or even to move it to a new architecture without rewriting all affected applications. (Raymond, Tompa and Wood, 1996)

The solution developed by the database community was to generalise the semantics of data by developing data models that described the logical properties of data, independently of how it was stored (Raymond et al., 1996). The relational model, proposed by Codd (1970), makes the evolution of a database much easier than using a data model based on a specific data representation such as a hierarchical or network data model.

Most current database applications are implemented in either the relational, object or object-relational data models. The relational model is well suited to transaction processing, but has shortcomings in terms of semantic expressiveness that has led to the development of semantic models such as the ER model. A number of database research efforts have concentrated on expanding the expressiveness of the database modelling mechanism in order to increase the understanding and usability of conceptual schemata. These so-called *semantic* or *conceptual* data models, use primitives such as entities, attributes, relationships, aggregation, generalisation, and constraint mechanisms, and are based on set theory which typically provides a much more expressive modelling mechanism.

For the database community, the semantics of applications is of great concern. Focussing on data models is a natural way to ensure the retention of semantics, since data models deal with semantics directly, without taking into consideration the machine, operating system or data representation (Raymond et al., 1996). Conceptual modelling has traditionally been used to capture the meaning and structure (schema) of the information to be managed in database applications. In this respect, the conceptual model also acts as input for the generation of a specific database schema conformant to some chosen implementation technology (Berild, 2004). As discussed in the preface of the ER conference 2006 (Embley, Olivé and Ram, 2006), conceptual modelling has now become fundamental to any domain in which organisations have to cope with complex, real-world systems. It has become a key mechanism for understanding and representing computing systems and environments of all kinds, including the new e-applications and the information systems that support them.

In addition, as discussed by Parent, Spaccapietra and Zimányi (2006a), today's research on the semantic web again emphasises the need for conceptual modelling approaches to facilitate information exchange over the Internet and within heterogeneous, distributed or federated databases. In such contexts, a conceptual model provides the best vehicle for delivering a common understanding between application partners with different technical and application backgrounds.

2.2 Data Models

The various data models proposed in the literature fall into two types: conceptual models, used in database design; and logical models, supported by the database management systems (DBMSs) that create, modify and maintain databases (Batini, Ceri and Navathe, 1992). This section presents definitions of data models.

The term *data model* has been used in the database community with various meanings and in diverse contexts. Various definitions of data models (listed in ascending order by published year) are as follows:

— *A data model is a common language for describing constraints on data and the effect of operations on that data* (Kerschberg, Klug and Tsichritzis, 1976).

— *A data model is a mathematical framework for representing knowledge* (McGee, 1976).

— *A data model is a combination of three components: a collection of data structure types, a collection of operators or inference rules, and a collection of general integrity rules* (Codd, 1980).

— *A data model is a collection of concepts that can be used to describe a set of data and operations to manipulate the data* (Batini et al., 1992)¹.

— *A data model is a set of concepts that can be used to describe the structure of and operations on a database* (Navathe, 1992).

— *A data model is a collection of conceptual tools for describing the real-world entities to be modelled in the database and the relationships among these entities* (Silberschatz, Korth and Sudarshan, 1996).

— *A data model is a collection of concepts that can be used to describe the*

¹Based on this definition, when a data model describes a set of concepts from a given reality, it is called a **conceptual data model**.

*structure of a database providing the necessary means to achieve data abstraction*² (Elmasri and Navathe, 2007).

The term *data model* as used in the literature denotes different levels of abstraction and this accounts for a degree of confusion when trying to provide a consensus view of what a data model actually represents. To remove any ambiguity, this thesis relies on the most concrete sense of the term *data model* as defined by Codd (1980).

Codd (1980) further pointed out that many authors appear to understand that a data model is only a collection of data structure types and often ignore the operators and integrity rules. These operators and integrity rules are essential to any understanding of how the structures behave, and as a consequence, when they are omitted, such models should be regarded as incomplete.

Data models first appeared in the early 1970s with the aim of providing some level of data abstractions so that different users may perceive data at their preferred level of detail. For example, the *hierarchical data model* was defined by a process of abstraction as part of the DBMS called Information Management System (IMS) created by IBM. The *network data model* was defined by a process of abstraction introduced by the Conference on Data Systems and Languages (CODASYL) Database Task Group. The *relational data model* (Codd, 1970), which is based on mathematical concepts and formal definitions such as relations and normal forms, organised data according to three levels of data description (external, conceptual and internal).

These three data models are examples of *logical* models. A **Logical model** is a data model that represents data descriptions in a form that can be processed by computer (Batini et al., 1992). However, such data models retain less of the original meaning of the data that is necessary for database designers and users in being able to interpret the contents of a database. As a response to this perceived need, better modelling tools are required to capture (in a more or less formal way) more of the meaning of the data so that database design can become semantically richer and the database system itself can behave more intelligently (Codd, 1979).

In the mid 1970's, the creation of **conceptual** or **semantic** data models helped to overcome these deficiencies of the logical data models by providing a stronger semantic foundation that retain more of the original meaning of the data. The

²Data abstractions are relevant to three levels of data description (external, internal and conceptual) through which the user can visualise the schema levels in a database system. This three levels or three-schema architecture is known as the ANSI/SPARC proposal (1975).

Entity-Relationship (ER) model (Chen, 1976) established in 1976 is the most popular model for conceptual modelling.³

A data model is qualified as conceptual if it enables a direct mapping between the perceived real world and its representation within the concepts of the model (Parent et al., 2006a). The conceptual data model helps designers capture the real world data requirements as it allows them to focus on semantic details of the concepts and their relationships, more than that which would be provided by the relational data model. The semantics represented in the ER model, for example, allow for direct transformations of entity types and relationship types to at least first normal form (1NF) tables. They also provide clear guidelines for integrity constraints (Teorey et al., 2006).

In addition, as discussed by Batini et al. (1992) and Navathe (1992), any semantic data model used for the purpose of conceptual design should be a suitable tool for representing reality. These data models should possess the following qualities:

- *Expressiveness.* The model must be sufficiently expressive to bring out the distinctions between different types of data, relationships and constraints, allowing analysts to capture relevant information not previously available. Additional features are introduced through the modelling of any new constructs that contribute to the semantics of the model.
- *Simplicity.* The model must be simple enough for any end user to use and understand. Hence, it must always be accompanied by easy diagrammatic notation.
- *Minimality.* The model must consist of a small number of basic concepts that are distinct and orthogonal in their meaning.
- *Formality.* The concepts of the model should be formally defined. It should be possible to state the criteria for the validity of a schema in the model.
- *Unique Interpretation.* Ideally, there should be a single semantic interpretation of a given schema. In turn, this implies that complete and unambiguous semantics can be defined for each modelling construct.

³Conceptual modelling was used either as a synonym to semantic data modelling or in the technical sense of the ANSI/X3/SPARC report (1975) where it referred to a model that allows for the definition of schemata lying between external views, as defined for different groups of users, and internal schemata defining one or several databases (Mylopoulos, 1992).

This thesis also suggests the inclusion of orthogonality as below:

- *Orthogonality.* A way that purportedly makes the modelling dimensions independent of each other, so that the modelling of new constructs in one dimension are independent of the existing constructs in another dimension. Orthogonality is the best way to provide maximum expressive power while retaining maximum simplicity in the constructs of the model (Spaccapietra, Parent and Zimányi, 2008).

At the highest and most abstract level, a conceptual data model describes how relevant information is structured in the natural world for purposes of communicating a common understanding. It should be independent of any database management system or other implementation considerations and is usually expressed in verbal or graphical form based on formal notations that allows for the capture of the semantics of the application.

In the design of databases, conceptual models are first used to produce a high-level description of reality then the conceptual schema is translated into a logical schema. The goal of conceptual data modelling is to capture real-world data requirements in a simple and meaningful way that is understandable by both the database designer and the end user. The conceptual data model has been most successful as a tool for communication between the designer and the end user and helps them understand and describe the contents of database in an intuitive way.

2.3 Conceptual Schema

A conceptual schema is a global description of the database that hides the details of physical storage structures and concentrates on describing entities, data types, relationships and constraints (Thalheim, 2000). As discussed by Sowa (2005), the need for standardised ways of encoding knowledge has been recognised since the 1970s. The American National Standards Institute (ANSI) proposed that all pertinent knowledge about an application domain should be collected in a single *conceptual schema* (Tsichritzis and Klug, 1978).

As described by Sowa (2005), Figure 2.1 illustrates an integrated system with a unified conceptual schema at the centre. Each circle is specialised for its own purposes, but they all draw on the common application knowledge represented in the conceptual schema. The user interface calls the database for query and editing facilities, which, in turn, calls the application programs to perform actions and

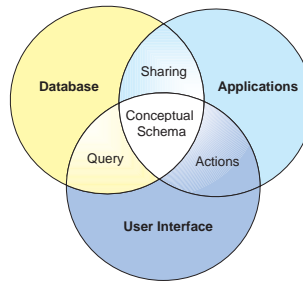


Figure 2.1: Conceptual schema as the heart of an integrated system (from Sowa, 2005)

provide services. Thus, the database supports both the application programs with facilities for data sharing and persistent storage. The conceptual schema binds all three circles together by providing the common definitions of the application entities and the relationships between them.

Sowa (2005) stated that for more than twenty years, the conceptual schema has been important for integrated application design, development and use. The most recent attempt to integrate all the world’s knowledge is the semantic web and so far, its major contribution has been to propose XML as the common syntax for sharing common semantics between applications.

It is important to distinguish between a conceptual data model and a conceptual schema. The former refers to the technique (including its notation) that is used to model any database. The ER (Chen, 1976), NIAM/ORM (Falkenberg, 1976; Halpin, 2001; Nijssen, 1976, 1977; Nijssen and Halpin, 1989; Verheijen and van Bekkum, 1982) and UML (Booch, Jacobson and Rumbaugh, 2005; Muller, 1999) are all examples of conceptual data models. On the other hand, conceptual schemata refer to the result of the modelling, namely a set of diagrams described by a specific data model in the form of diagrammatic conventions or a language syntax to express the specific data structures for an application that is going to be developed (Fonseca, Davis and Câmara, 2003; Fonseca and Martin, 2007).

2.4 Semantics in Conceptual Modelling

“...Data by itself is not enough — what we really need is information, the meaning or semantics behind the data. Since computers lack common sense, we need to pay special attention to semantics when we use computers to model some aspect of reality.”

Terry Halpin (2001)

Next generation database systems will not work without semantics (Mylopoulos, 2004). Several perspectives on semantics appeared in the literature, in particular, in a panel discussion of the Data Semantics (DS-6) conference compiled by Sheth, Meersman and Navathe (1995). Various definitions of semantics are given below:

— *Semantics is the meaning and the use of data* (Woods, 1988).

This is the classic definition of semantics.

— *Semantics can be viewed as a mapping between an object modelled, represented and/or stored in an information system (e.g. an ‘object’ in a database) and the real-world object(s) it represents* (Sheth et al., 1995).

This mapping represents the semantics of the modelled object by describing or identifying the meaning and the use perspectives.

— *Semantics is all pervasive and covers many things such as the interpretation and the use of data, or the interaction of people to convert data into information (that is, semantics is everywhere and has broad interpretation).*

This definition is given by Navathe, one of the four panellists of DS-6 conference, who defined semantics in a broader term (Sheth et al., 1995).

— *Semantics is dependent on humans, thus it is difficult to address it in the context of machines (no wonder the systems oriented database researchers often find it a ‘soft science’).*

This is another definition by Navathe, given at the DS-6 conference (Sheth et al., 1995) that shows another interesting philosophical point of this term.

— *Semantics, in a formal way, is the precise description of the link between a representation and concepts in the real world; in a casual way, semantics is a feature of a language or representation that supposedly can be measured or compared in intuitive ways* (Meersman, 1996).

This latter definition appears to be the most frequently used in the database and information system literature. Examples of this form are approaches where semantics are conceptually linked to relationships, constraints, objects or application functionality.

— *Semantics can be identified by relationships between objects* (Wiederhold, 1995) in Sheth et al. (1995).

This term is identified by Gio Wiederhold, one of the three keynote speakers at the DS-6 conference (Sheth et al., 1995). In addition, Sheth, Thacker and

Patel (2003) stated that relationships between entities (terms or concepts) are the basis of capturing, representing and supporting semantics. This definition is indeed the perspective taken by many in the knowledge representation and conceptual modelling fields.

— *Semantics is defined as the meaning of a term or a mapping from a construct to the real world* (Purao and Storey, 2005).

According to this definition, understanding a relationship requires that one understands the semantics of the accompanying verb phrase.

— *Semantics is defined as the meaning, or essential message, of the terms used in the conceptual model, that is, of words and phrases representing entities and verbs* (Storey, 2005).

According to this definition, mechanisms for capturing some of the semantics of the real world are needed to compare the entities and verb phrase relationships.

For the purpose of this thesis, the concept of semantics that is used throughout this thesis encompasses all of these understandings and perspectives that reflect upon the common understanding of this term. Understanding the semantics of the real world and representing them in a conceptual schema are becoming common in conceptual modelling. Ontologies capture useful semantics about the domains that they model, relationships between them and the characteristics of the domain (Sheth et al., 2003). A variety of the semantics of relationships can be seen in context through conceptual schemata and ontologies. This understanding that relationships between objects are the key to semantics is also reflected in the definitions suggested by both Wiederhold (Sheth et al., 1995) and Sheth et al. (2003).

Entities in the real world are related to each other in various ways (Sheth et al., 2003). These relationships can be simple such as **is-a** and **is-part-of**, which are basic hierarchical structures or can be much more complex relationships which require complex structures. For example, the semantic relationships such as **CLOSETO** and **NEXTTO** presented in this thesis capture the degrees of proximity of address/location (refer to Chapter 4). Similarly, the domain knowledge that uses ontologies to describe terms or concepts provides additional meanings to a location attribute through the relationships between its concepts (refer to Chapter 5).

Recent research efforts have been devoted to the further understanding of the semantics of relationships. Bergholtz and Johannesson (2001) proposed an ontology for classifying relationships based on data abstractions. Purao and Storey

(2005) proposed a layered ontology for classifying the semantics of relationships. Other similar efforts provide a way to create database designs that capture and represent the semantics of an application. For example, Storey (2005) proposed an ontology for classifying relationship verbs based upon the domain and context of the application within which the relationship appears. Sugumaran and Storey (2006) showed how domain knowledge stored in an ontology can be used to assist in the generation of complete and consistent database designs. These are based on the concept that understanding the semantics of relationships requires an explicit characterisation of the meaning underlying the descriptive verb that is part of each relationship.

It is believed that semantics is a grand challenge for the current generation of computer technology (Embley, 2004) and ontologies will help solve the major problem of incorporating semantics into data modelling. This thesis will show one potential use of ontologies in enhancing richer semantics in conceptual modelling (refer to Chapter 5).

2.5 Conceptual Design as Part of Database Design

“Conceptual modeling is a very important phase in designing a successful database application.”

Ramez Elmasri and Shamkant B. Navathe (2007)

As stated by Rolland and Cauvet (1992), conceptual models have proved to be extremely useful throughout the information system life cycle. The growing demand for information systems of ever-increasing complexity and size, calls for high level concepts and formal techniques to model systems at different levels of abstraction. The need for powerful conceptual tools and better forms of abstraction, particularly in the earlier phase of systems development, has been recognised throughout industry, business and administration.

As discussed by Halpin (2001), when a database is designed for a particular application, a model of the application area is created. Technically, the application area being modelled is called the Universe of Discourse (UoD), since it is the world (or universe) that we are most interested in. The UoD is also called the ‘the application domain’ and typically represents ‘part’ of the ‘real world’. The main challenge is to describe the UoD clearly and precisely.

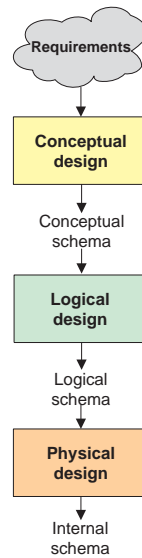


Figure 2.2: Three main phases of database design (from Batini et al. (1992)).

Designing databases is still considered as an art rather than a science. When supported by a good conceptual data model which has enough expressive power to support the modelling of all situations of interest, this design process can deliver high quality database applications. However, there is no obvious answer to the question of how much expressive power a data model should possess, since being conceptual, data models may have more or less expressive power depending on the number of concepts, constructs and constraints they support (Parent et al., 2006a).

Database applications are modelled using a three-step design paradigm. Conceptual design is the first stage in the process of top-down database design which is decomposed into *conceptual*, *logical* and *physical* design as shown in Figure 2.2. The objective of conceptual design is to describe the information used by an organisation in a way which is not governed by implementation-level issues and details, and is initiated from requirement specifications that describe the reality. In this first phase, the model should be expressed at the conceptual level to make it easy for people to see the overall picture, thus enabling any non-technical stakeholders to be able to view, understand and to contribute to any discussions on the database design.

The three-step design process as discussed by Batini et al. (1992) and Shekhar, Vatsavai, Chawla and Burk (1999) is summarised as follows:

Conceptual Design. A conceptual design starts from the specification of requirements and results in the conceptual schema of the database. A *con-*

ceptual schema is a high-level description of the structure of the database, *independent* of the particular DBMS that will be used to implement the database. A common method of analysis involves identifying the essential data that needs to be stored:

- entities that the organisation has to deal with;
- attributes and items of information that characterise and describe these entities; and
- relationships between entities that exist and must be taken into account when processing information.

At the conceptual design level, the focus is on the entity types of the application, their relationships and constraints, and therefore the actual implementation details are omitted from this step. To assist the design process, graphical notation or visual tools are commonly used to express the data and their relationships.

Of all the available conceptual design methodologies, the ER model is the one that is used most prevalently.

Logical Design. A logical design starts from the conceptual schema and results in the logical schema. A *logical schema* is a description of the structure of the database that can be processed and supported by a commercial database management system (DBMS). The logical data model is often directly used as part of the computer implementation in some DBMSs but omits the physical details related to the implementation. It contains modelling constructs that are easy for users to follow, and is also referred to as an implementation model.

The most widely used logical data model in current commercial databases is the *relational* data model. At this point of the process, the logical schema now represents familiar relations, tuples, attributes, primary keys and foreign keys of relational databases and allows queries to be expressed without direct reference to specific indexes. Normalisation⁴ is applied at this step to minimise redundancy in the data and to leverage maximum utility of relational concepts. *Object-oriented*, *object-relational*, *network* and *hierarchical* data models are also examples of the actual implementation of the conceptual data model.

⁴Normalisation is a methodology to analyse an association between attributes to extract functional and join dependencies from the real-world semantics.

Note that the entire activity, starting from requirements and producing the definition of the final implementable schema in a DBMS, can also be called **schema design** (Navathe, 1992).

Physical Design. A physical design starts from the logical schema and results in the physical schema. A *physical schema* is a description of the implementation of the database and is strongly coupled to the DBMS. It is used to describe the storage structures and access methods used in order to effectively access data. Issues relating to the disk representation of data related to the storage, clustering, partitioning, indexing, and space and memory management are handled at this level. There may be an iterative process loop between the physical and logical design phases as the decisions taken during physical design for improving performance might affect the structure of the logical schema.

The most demanding phase in database design is **conceptual design**. The conceptual schema is the first formalised description of the database application which serves as the basis for discussions with stakeholders on database design or on further design steps to achieve fuller functionality of an existing database system (Engels, Gogolla, Hohenstein, Hülsmann, Löhr-Richter, Saake and Ehrlich, 1992). Conceptual data modelling is thus imperative for successful information systems development (ter Hofstede, Lippe and van der Weide, 1995).

2.6 Key Considerations of a Conceptual Data Model

From a database point of view, conceptual data models that constitute a data(base) model should at least possess: data structures that define the template of a database; constraints (rules) that define the set of accurate and consistent database states; and languages and operations on the database that allow retrievals and updates.

According to these criteria, conceptual models are thus formally considered as comprising three parts; data structures, integrity constraints and languages. These criteria correspond to a combination of three components (a set of data structure types, a set of operators or inferencing rules, and a set of general integrity rules) used to describe a data model as proposed by Codd (1980). These key considerations of a conceptual data model are described below.

Data Structures. The structure of a data model refers to the modelling constructs with which the structure of a database can be described. For example, the data structures of the ER model are *entity types*, *relationship types* and *attributes*.

Integrity Constraints. Constraints are an additional restriction on the occurrences of data within a database that must hold at all times (Navathe, 1992). They are used to ensure accuracy and consistency of data in a database. As discussed by Navathe (1992), data model constraints serve two primary goals:

- *Integrity.* Integrity constraints are the rules that constrain the valid states of a database. They arise either as properties of data or as user-defined rules that reflect the meaning of data. For example, if a user tries to insert data that does not meet the rules, the DBMS will not allow it. Also, requirements (i.e. rules) imposed by a relationship in the ER model must be met when updating the database.
- *Security and Protection.* This applies to restrictions and authorisation limitations that are applied to a database so as to protect it from misuse and unauthorised usage.

Constraints as described by Borysowich (2007), Elmasri and Navathe (2007) and Navathe (1992) can be visualised at different levels and are summarised as:

- *Inherent-based constraints.* An inherent-based constraint is a constraint that is built into the rules of the data model itself. For example, in the ER model, a relationship must have a least two participating entity types.
- *Schema-based constraints.* A schema-based constraint is a constraint that can be directly expressed in a schema of a data model, typically by specifying this constraint in the data definition language. These constraints are expected to be automatically enforced. For example, a total participation constraint in the ER model, states that a specific entity must participate in a particular relationship and therefore it requires a minimum of one occurrence for any particular entity that participates in the relationship. Consider the relationship ‘Teacher teaches Course’. A Course cannot exist unless a Teacher has been assigned to teach the Course.

- *Application-based constraints.* An application-based constraint is a constraint that can not be directly expressed in a schema of a data model and hence must be expressed and enforced by the application program. This constraint is more general and relates to the meaning as well as behaviour of attributes. Since this is difficult to express and enforce within the data model, it is usually checked within the application program.

Languages. A data model should provide languages and operations for the database that allow retrievals and updates including insertions, deletions and modifications. A data model in the database parlance is associated with a variety of languages such as data definition language (DDL) and data manipulation language (DML). The DDL allows the database designer to define the database schema. The DBMS has a compiler to process the schema definition in DDL and to convert it into a machine-processable form (Navathe, 1992). The DML is used to specify the retrieval, insertion, deletion and modification of data. Languages for data models can also be distinguished in terms of *record-at-a-time*⁵ or *set-at-a-time*⁶ (Navathe, 1992).

In the following section, the criteria related to the key considerations of a conceptual data model are applied to show that each of the well-known data modelling approaches, ER, ORM and UML, can be regarded as completed data models.

2.7 Conceptual Modelling Approaches

In the past, a number of data models, called semantic or conceptual models have been proposed for conceptual modelling, e.g. the Semantic Data Model (SDM) (Hammer and McLeod, 1981), the Functional Data Model (Shipman, 1981), the Nijssen's Information Analysis Methodology (NIAM) approach (Falkenberg, 1976; Nijssen, 1976, 1977; Nijssen and Halpin, 1989) and the Entity-Relationship (ER) model (Chen, 1976). For a comprehensive survey of these, refer to Hull and

⁵Record-at-a-time, the low level or procedural language, requires an elaborate control structure typically provided by a host programming language within which the DML commands are embedded (Navathe, 1992).

⁶Set-at-a-time, the high level or non-procedural DML, can specify and retrieve sets of elements (e.g. sets of tuples in the relational model) in a single DML statement. This retrieval of a tuple is said to be Set-at-a-time or Set-oriented.

King (1987) and Potter and Kerschberg (1988). Amongst these, one of the first and most powerful semantic data models was the ER model proposed by Chen (1976). Since this time, several conceptual modelling researchers have continued to evolve the model by trying to express it using semantically enriched modelling formalisms. A number of new abstraction mechanisms have been proposed, such as specialisation, aggregation and association (Batini et al., 1992; Elmasri and Navathe, 2007) to enrich its modelling capabilities. Another well-known fact-oriented model for conceptual modelling is the NIAM approach (Nijssen, 1976, 1977; Nijssen and Halpin, 1989). Although it is less popular than the ER model, it has also gained some attention from the database community and several papers have been presented (Leung and Nijssen, 1988; Creasy, 1989; Song and Forbes, 1991; Laender and Flynn, 1993; Puntheeranurak and Chittayasothorn, 2002). This model is now widely known as Object Role Modelling (ORM) (Halpin, 1998, 2001). More recently, Unified Modelling Language (UML) class diagrams (Muller, 1999; Halpin, 1998-1999a) have also gained popularity for data modelling.

There are many modelling approaches that provide for diagrammatic representation of conceptual models and much of the literature on data modelling is devoted to different modelling techniques. This thesis provides a discussion of three of these well-known approaches, ER, ORM and UML class diagram that are used for conceptual modelling. The following explanation of this model draws particular attention to *data structures*, *integrity constraints* and *languages* that are the key considerations of the model.

2.7.1 The Entity-Relationship (ER) Model

“(The Entity-Relationship model) incorporates some of the important semantic information in the real world...”

Peter Chen (1976)

The ER model was the first conceptual data model that was proposed by Peter Chen in 1976, and is still the most widely used data modelling approach for the conceptual design of databases and information systems. It can be described as a top-down approach that views the real world as entities (that have attributes) and relationships. The ER model exemplified semantic data models and has been a precursor of much subsequent development (Navathe, 1992). The key elements of a (conceptual) data model as they apply to the ER model are described below:

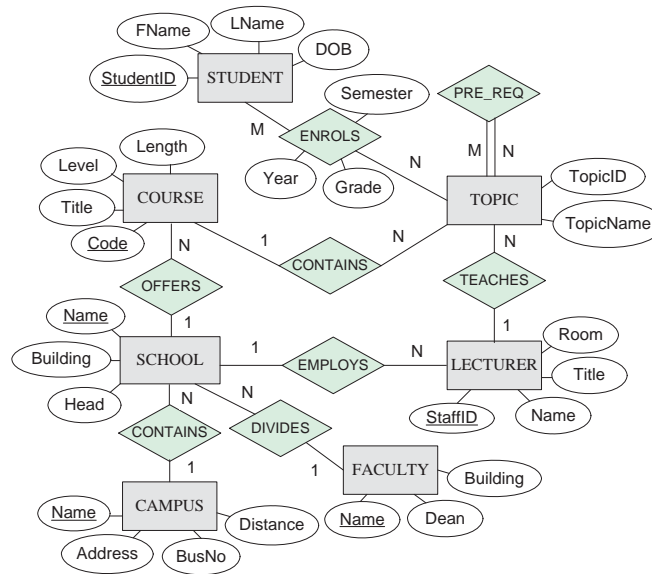


Figure 2.3: An example of an ER diagram for a university database.

Data Structures. The main constructs in the ER model are *entity types*, *relationship types* and *attributes* which are represented graphically as an entity-relationship diagram.

1. **Entity types.** In the ER model, the real world is modelled into entities, that are characterised by attributes and interrelated through relationships. An *entity*, or more precisely an *entity instance*, represents an object of interest that may be concrete or abstract. A collection of similar entities forms an *entity type*. An entity type is similar to a class of objects, however, it is not involved with the operations on data and thus there are no methods associated with an entity type.
2. **Relationship types.** A relationship type represents a meaningful association between two or more entity types.
3. **Attributes.** An attribute is a property of the entity. The set of values an attribute can take is often termed a *domain*.

Figure 2.3 shows an example ER diagram for a university database. In the ER diagram, entity types, relationship types and attributes are represented by rectangles, diamonds and ellipses respectively. The ellipse with an underlined label indicates a unique identifier attribute which uniquely identifies instances of an entity.

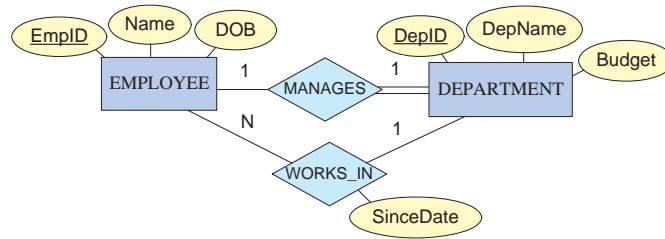


Figure 2.4: Cardinality ratio and participation constraint in the WORKS_IN and MANAGES relationship types.

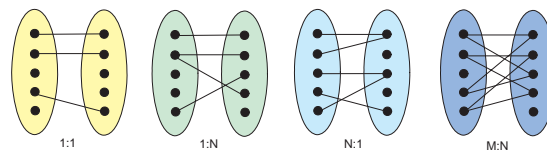


Figure 2.5: Cardinality ratios in the ER model.

Integrity Constraints. There are two types of constraints on relationship types in the ER model: *cardinality ratios* and *participation constraints* that fall under the schema-based constraint category.

1. **Cardinality ratios.** A cardinality ratio specifies the maximum number of relationship instances that an entity can participate in (Elmasri and Navathe, 2007). For example, consider the WORKS_IN relationship type in Figure 2.4. A cardinality ratio of EMPLOYEE:DEPARTMENT is N:1, meaning that an employee can work in only one department while a department can have many employees. The possible cardinality ratios for relationship types in the ER model are 1:1, 1:N, N:1 and M:N as shown in Figure 2.5, where the N and M notations refer to any number of instances (refer to Elmasri and Navathe (2007)).
2. **Participation constraint.** A participation constraint specifies whether the existence of an entity depends on its relationship to another entity via the relationship type (Elmasri and Navathe, 2007). There are two types of participation constraints — *total* and *partial*. Consider the MANAGES relationship type in Figure 2.4. If we do not expect every employee to manage a department, then the participation of EMPLOYEE in the MANAGES relationship type is *partial* as depicted by a single line connecting EMPLOYEE and MANAGES and if we expect that every department has a manager, then the participation of DEPARTMENT in the

MANAGES relationship is *total* as depicted by a double line connecting DEPARTMENT and MANAGES.

Languages. A discussion of query languages for the ER model ranges from procedural languages (Campbell and Embley, 1985; Demo, Di Leva and Giolito, 1985), descriptive languages such as GORDAS (Elmasri and Wiederhold, 1981) to graphical languages (Campbell, Embley and Czejdo, 1987; Elmasri and Larson, 1985) such as GRAQULA (Sockut, Burns, Malhotra and Whang, 1993) or GQL/ER (Zhang and Mendelzon, 1983). There are also query languages such as SQL/ER (Gogolla and Hohenstein, 1991) and others that are based on the Extended Entity-Relationship (EER) model such as SQL/EER (Hohenstein and Engels, 1992).

However, there are no DBMSs that use the ER model directly. As most commercial DBMS uses the relational model, the ER model is thus converted to a relational schema in the data-specification language of the relational DBMS. Subsequently, a RDBMS language, such as SQL, is used to create a schema in the conceptual design phase. The ER model integrates seamlessly with the relational data model and can be mapped into a relational database schema which guarantees the first normal form (1NF).

To capture the real-world semantics in the ER model, one main problem that needs to be overcome is how to distinguish between attributes and entity types. The distinction between entity types and attributes tends to be fuzzy (Navathe, 1992). Also, there seems to be not just one correct ER diagram for a given situation. For example, one may consider the Country in which a person was born to be modelled as an attribute of the PERSON entity type Figure 2.6(a). Conversely, it could be argued that the COUNTRY entity type should be modelled with the attributes of, for example, Name, Population and Zone. The latter argument is supported by promoting relationship types between the PERSON and COUNTRY entity type as shown in Figure 2.6(b).

Given the term Country from the example in Figure 2.6, this thesis suggests that the following guideline should be used to assess whether Country should be modelled as an attribute or entity type.

- Country should be modelled as an attribute if it is required to record only one piece of information about a Country (i.e. the Country's name) and Country is related to only one entity type.

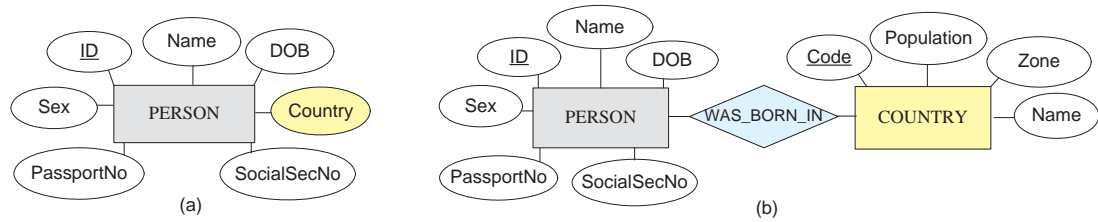


Figure 2.6: Two design choices of Country (a) attribute of PERSON entity type and (b) entity type COUNTRY.

- Country should be modelled as an entity type if it is required to record multiple pieces of information about Country (e.g. Name, Population, Zone) and where Country is also associated with any other entity types through relationship types.

2.7.2 Object Role Modelling (ORM)

ORM is a method for modelling and querying an information system at the conceptual level (Halpin, 1998). ORM's precursor is often referred to as NIAM, which was initially developed by Nijssen and others in Europe in the early 1970s (Nijssen and Halpin, 1989). NIAM was originally an acronym for Nijssen's Information Analysis Methodology (Nijssen, 1976, 1977; Nijssen and Halpin, 1989), but more recently, it has been revised to Natural Language Information Analysis Method (Verheijen and van Bekkum, 1982). At present, a more general name for NIAM is Object Role Modelling (ORM) (Halpin, 2001).

ORM expresses the information in terms of elementary relationships which means that a base concept of ORM makes no use of attributes, but uses a relationship instead (Halpin, 2000). This differs from other modelling techniques such as the ER or UML approaches in that these approaches use attributes as their main constructs. The key elements of a (conceptual) data model as they apply to the ORM approach are described as follows:

Data Structures. The main constructs of ORM are *entity types* (types of object), *label types* (type of values or names), *reference types* and *fact types*.

1. **Entity types.** An entity type is the set of all possible values. Each entity is an instance of the particular entity type. For example, the entity type STUDENT is the set of all students. That is, an *entity type*

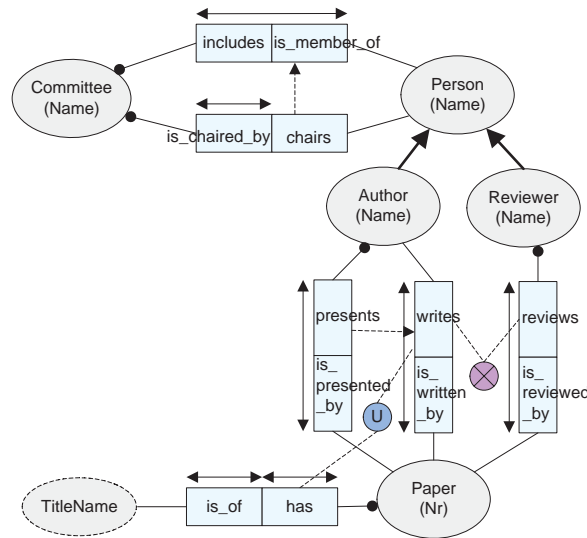


Figure 2.7: An example of an ORM diagram (adapted from Jarrar et al. (2003)).

is a generic collection of abstract or real entities. Note that in ORM the word ‘entity’ refers to ‘entity instance’. The semantics of the constructs are similar to that of the ER model, however the graphical representation is different. In ORM, an entity type is depicted as a named ellipse whereas in ER a named rectangle is used.

2. **Label types.** A label type is used to denote a particular object. In other words, it is a naming of an entity type and is usually depicted as a named, dotted ellipse.
3. **Reference types.** Relationship types in ORM are known as reference types and fact types that are depicted as a named sequence of one or more roles, where each role appears as a box connected to the object type that plays it. A *reference type* is an association between entity types and label types. An entity type may have several label types and its unique identifier is chosen from a one-to-one reference type.
4. **Fact types.** A *fact type* is an association between entity types. Each fact type is in the form of elementary facts (Halpin, 1993). This property enables the ORM conceptual schemata to be mapped into the 5NF relational schemata (Leung and Nijssen, 1987; Pornphol and Chittayasothorn, 2004).

Consider Figure 2.7 that depicts an example of an ORM diagram. Object types are shown as a named ellipse, with solid lines for entity types and

dotted line for label (value) types. Fact types appear as a named sequence of roles, where each role appears as a box connected to the object type playing it. Entity types are Committee, Person, Author, Reviewer and Paper. A label type is TitleName. Committees, Persons, Authors and Reviewers are identified by their name while Papers are identified by their paper number. A named bracket is used to concisely represent a unique identifier, which is placed below the entity type name. There is one reference type (i.e. a Paper has a TitleName) while the remaining are fact types (e.g. an Author writes a Paper, an Author presents a Paper, a Reviewer reviews a Paper, etc.).

Integrity Constraints. ORM is typically more expressive than ER in terms of its wider variety of constraints. Examples of such constraints that come under the schema-based constraint category include uniqueness, mandatory role, entity type, subtype and set comparison.

1. **Uniqueness constraints.** A uniqueness constraint is a constraint on the fact types to indicate that each fact instance is unique. Arrow-tipped bars over one or more roles represent uniqueness constraints. There are two types of uniqueness constraints: intra-fact-type and inter-fact-type.

- *Intra-fact-type constraints.* An intra-fact-type constraint, sometimes called *internal constraint*, declares that the role in the relationship type must be unique. These are shown as arrow tipped bars and are placed over one or more roles in a predicate⁷. For example, adding a uniqueness constraint over the first role of fact type ‘Committee is chaired by Person’ in Figure 2.7 declares that each committee is chaired by at most one person and that each entry in the Committee column must be unique (no committee’s name can be duplicated in that column). The predicate ‘...is chaired by...’ is *many to one* (N:1). The inverse predicate ‘...chairs...’ is said to be *one to many* (1:N).
- *Inter-fact-type constraints.* An inter-fact-type constraint, sometimes called an *external constraint*, indicates that instances of the

⁷A predicate (as discussed by Halpin (1998), Halpin and Morgan (2008) and Nijssen and Halpin (1989)) is basically a declarative sentence with object holes in it, one for each role. The number of role is called the *arity* of the predicate. ORM allows predicates of any arity (e.g. 1=unary, 2=binary, 3=ternary etc.) in which the name of the predicate can be written either in or beside the first role box. For example, the fact type **an Author writes a Paper** involves the predicate ‘...writes...’.

combination of the roles in the join of those predicates are unique. These are depicted as a circled U applied to two or more roles from different predicates and are interconnected with dotted lines. For example, to identify that a paper is defined by a combination of a title name and authors, an inter-fact-type constraint is used that joins the two role boxes shown in Figure 2.7. This indicates that for each paper the combination of a title name and authors is unique. This means that when querying any title name and author name, there is at most one PaperNr which is paired with both.

2. **Mandatory role constraints.** A mandatory role constraint declares that every instance in the population of the role's object type must play that role. These are represented graphically by a black dot (Figure 2.7). For example, each Paper must have a Title name.
3. **Entity type constraints.** An entity type constraint, sometimes called a *domain constraint* or *value constraint*, restricts an object type's population to a given list. The relevant values may be listed in braces. For example, the set of possible sexes may be listed or enumerated as {M,F}, and placed alongside the entity type. If the values are ordered and have a continuous range, the range can be declared by separating the first and last values by '..' which abbreviates the integers in between. For example, a range of employee age may be indicated as [18..65].
4. **Subtype constraints.** A subtype constraint indicates that the specified object type is a subtype of another. These are depicted by a solid arrow. For example, in Figure 2.7, the solid arrows connecting the object types Author and Reviewer to Person denote a subtype (is-a) relationship, i.e. both author and reviewer are a subtype of person.
5. **Set comparison constraints.** A set comparison constraint restricts the way the population of one role, or role sequence, relates to the population of another. These constraints can be applied between compatible role sequences where the corresponding roles have the same object type. Set comparison constraints declare a *subset*, *equality* and *exclusion* relationship between the populations of role sequence. An example of exclusion and subset constraints is described below:
 - *Exclusion constraints.* An exclusion constraint indicates that the

populations are mutually exclusive. This is depicted by the symbol \otimes . For example, in Figure 2.7, the \otimes symbol between the fact types ‘Reviewer reviews Paper’ and ‘Paper is written by Author’ indicates that an author who writes a certain paper is not allowed to be a reviewer of the same paper.

- *Subset constraints.* A subset constraint declares that the population of the source role sequence must be a subset of the target role sequence. This is depicted by a dotted arrow that connects any pair of compatible role sequences. For example, consider the subset constraint between the Person-Committee role pair in Figure 2.7. This constraint declares that any person who chairs a committee must be a member of that committee.

A detailed discussion of equality constraints including other constraints such as *ring* and *frequency* constraints provided by ORM, can be found in Halpin (2001) and Nijssen and Halpin (1989).

Languages. A powerful query language is important for successful database modelling. To allow queries and updates to be performed at a conceptual level requires interaction with conceptual structures, rather than with relational databases. ConQuer (Bloesch and Halpin, 1996) is a conceptual query language based on ORM allowing users to formulate queries naturally in terms of elementary relationships. Operators such as ‘and’, ‘not’ and ‘maybe’ can be used and it does not require the user to have any understanding of how the information is stored in the underlying data structures. Similarly, the conceptual query language RIDL (De Troyer, 1989) is based on NIAM (ORM’s precursor methodology) (Verheijen and van Bekkum, 1982; Demey, Jarrar and Meersman, 2002). ORM-Markup Language (ORM-ML) (Demey et al., 2002) has also been proposed as a method to express ORM schemata using an XML-based markup language. The ORM approach has well-defined semantics and uses elementary fact types which can be easily mapped into fifth normal form (5NF) relational schema. Once the conceptual schema has been transformed to a relational schema, it then can be supported by SQL which is widely used in commercial DBMSs.

2.7.3 Unified Modelling Language (UML) Class Diagrams

The Unified Modelling Language (UML) is a relatively new object-oriented analysis and design method. The Object Management Group (OMG) (OMG, 2008) is in charge of developing and standardising UML and this sponsorship aims to promote its universal acceptance and for it to become an emerging standard for conceptual data modelling in object-oriented domains (OMG, 2005). Several types of diagrams are provided by UML to assist developers of object-oriented programming (Booch et al., 2005; Halpin, 1998-1999*a*; Muller, 1999). As far as conceptual data modelling is concerned, this thesis is only interested in the UML *class diagram* which is used for data modelling purposes.

The key elements of a (conceptual) data model as they apply to the UML class diagrams are described below:

Data Structures. The main modelling constructs of the UML class diagram that is used for conceptual modelling purposes are *classes*, *associations* and *attributes*. These constructs closely correlate to those associated with the ER model.

1. **Classes.** A class is a description of a set of objects that share the same attributes, associations and methods. In ER terminology, the term *class* has an equivalent meaning to that of *entity type*. Classes have properties in the form of attributes, provide abstract services in the form of operations, and can be related to other objects using associations. Classes in UML are typically depicted as named rectangles with three sections: the top section for the name of the class; the middle section for the attributes of the class; and the bottom section for the operations (methods) of the class. When used in conceptual analysis and design, implementation details such as methods are omitted. The nature and usage of UML class diagrams closely mirrors equivalent processes in the ER model.
2. **Associations.** An association is the semantic relationship between two or more classes involving links or connections among their instances. These are depicted as solid lines connecting classes. In ER terminology, the term *association* has an equivalent meaning to that of *relationship type*.
3. **Attributes.** Attributes are properties of classes that describe a range of values that instances of this class may hold.

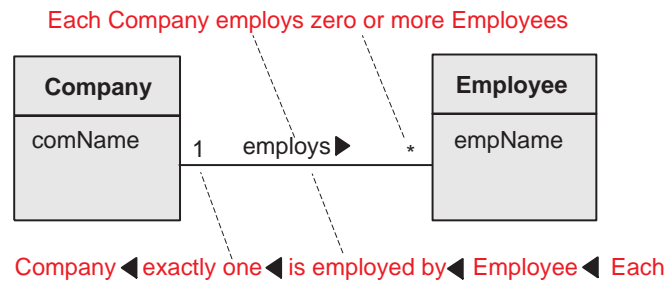


Figure 2.8: An example of a binary association with the expression of multiplicity constraints in a UML class diagram.

Integrity Constraints. The UML class diagram includes *multiplicity* constraints on the association roles. The concept of *multiplicity* in UML class diagrams corresponds to *cardinality* in ER terminology. The multiplicity specifies the number of target instances (at minimum and maximum) of one class that may be associated with a given *single* instance of another class. Each multiplicity constraint is placed on the *far role* in the direction in which the association is read (Halpin, 2001; Génova, Llorens and Martinez, 2001).

Figure 2.8 shows a multiplicity constraint on a binary association meaning that ‘each Company employs zero or more Employees’ while ‘each Employee is employed by exactly one Company’. The association is simply shown as a line connecting the two classes, named with a verb that describes the action. An arrow shows which way the association is read. Notations at each end of the line (i.e. ‘1’ and ‘*’) represents the multiplicity of the association. The ‘*’ character symbol abbreviates ‘0..*’, meaning ‘zero or more’. A ‘1’ abbreviates ‘1..1’, meaning ‘exactly one’. The multiplicity at each end of the line in Figure 2.8 is called a one-to-many association. Multiplicity is important in the data model since it maps directly into the structure of the foreign key in the logical schema. Table 2.1 shows some examples of multiplicity notations. Refer to Halpin (2001) for possible constraint patterns in UML for binary associations.

UML supports subclass and superclass, where each instance of a subclass is also an instance of its superclass. A subclass inherits all the attributes, associations (and operations/methods) of its superclass. The symbol for a subclass association in UML is an open arrowhead that points to the superclass. Rather than the usual multiplicity constraint, the subclass association line is labelled with four predefined constraints to indicate whether

Table 2.1: Examples of Multiplicities.

Multiplicity	Meaning
0..*	Zero or more objects
0..1	No more than one optional object
1	Exactly one object
1..*	One or more objects
2..10	At least two but not more than ten objects
1,3,9-10	At least one object but possibly three, nine or ten objects

subclasses are exclusive or exhaustive (Halpin and Morgan, 2008). Constraints are described along two dimensions: incomplete versus complete, and disjoint versus overlapping (Jewett, 2006a).

- *Incomplete versus complete.* These constraints can be considered as participation⁸ constraints.
 - *Incomplete.* The incomplete superclass/subclass relationship specifies that only *some* instances of the superclass belong to any of its subclasses. An incomplete superclass/subclass relationship is also called a *partial* participation. To represent an incomplete superclass/subclass relationship, the label {incomplete} is placed next to the subclass association line.
 - *Complete.* The complete superclass/subclass relationship specifies that *all* instances of the superclass must also be a member of a subclass. A complete superclass/subclass relationship is also called a *total* or *exhaustive* participation. To represent a complete superclass/subclass relationship, the label {complete} is placed next to the subclass association line. For example, in Figure 2.9 the subclasses of the Owner are complete, which means that every member of Owner must be either a private owner or a business owner.
- *Disjoint⁹ versus overlapping.* These two constraints, respectively indicate whether the subclasses are mutually exclusive or overlapping.

⁸Participation constraints determine whether every member in the superclass must participate as a member of a subclass (Connolly and Begg, 2004).

⁹Disjoint constraints describe the relationship between members of the subclasses and indicates whether it is possible for a member of a superclass to be a member of one or more subclasses (Connolly and Begg, 2004).

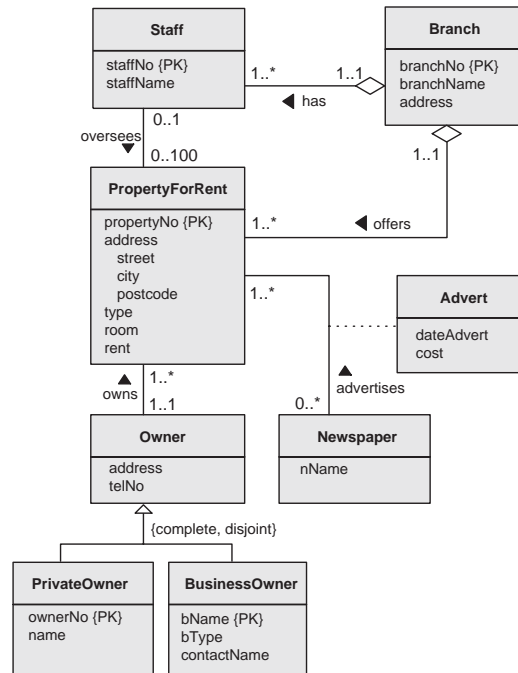


Figure 2.9: An example of a UML class diagram (adapted from Connolly and Begg (2004)).

- *Disjoint*. If subclasses are *disjoint*, then an entity instance of the superclass can be a member of *only one* of the subclasses. The disjoint constraint only applies when a superclass has more than one subclass (Connolly and Begg, 2004). A disjoint constraint also called an *exclusive* constraint. To represent a disjoint superclass/subclass relationship, the label {disjoint} is placed next to the subclass association line. For example, in Figure 2.9 the subclasses of the Owner are disjoint, which means that a member of Owner can be a private owner *or* a business owner, but not both.
- *Overlapping*. If the subclasses are *overlapping*, then an entity instance of the superclass may be a member of *more than one* subclass. To represent an overlapping superclass/subclass relationship, the label {overlapping} is placed next to the subclass association line.

The disjoint/overlapping and complete/incomplete constraints of a superclass/subclass relationship are distinct, giving rise to four categories: ‘complete and disjoint’, ‘incomplete and disjoint’, ‘complete and overlapping’ and ‘incomplete and overlapping’ (Connolly and Begg, 2004).

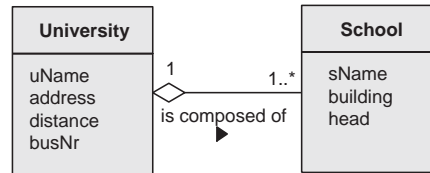


Figure 2.10: UML aggregation.

In addition, UML provides for an *aggregation*¹⁰ shown by an open diamond on the end of association line that points to the aggregated class to represent a collection of their component objects. The relationship between the primitive objects and their aggregate object is called a whole/part (Halpin, 2001) relationship or is described as *is-part-of*; the inverse is called *is-a-component-of* (Elmasri and Navathe, 2007). For example, a university is an aggregation of schools. Although this pattern can be modelled by an ordinary association, aggregation provides a more semantically correct way. Figure 2.10 describes that each university is composed of one or more schools and each school is part of one university.

Arbitrary constraints in UML can be declared by writing a comment or constraint in a note attached to the notations involved.

Languages. The OMG (OMG, 2008) has adopted UML as the standard notation for object methods (Muller, 1999). Several additions to the language are incorporated into the UML including the Object Constraint Language (OCL) and action semantics. As part of the UML, OCL provides the possibility of expressing constraints in a conceptual model unambiguously and enabling query expressions that can be used in conjunction with UML data models. Based on the OCL, a Unified Query Language (UQL) (Grinev and Kuznetsov, 2002) is also proposed. The mapping of OCL expressions to SQL (Demuth and Hussmann, 1999; Akehurst and Bordbar, 2001) as well as the mapping of a UML class diagram schema into a relational schema (Muller, 1999; Shah and Slaughter, 2003) are also provided.

Figure 2.9 shows an example of the UML class diagrams as discussed by Connolly and Begg (2004). This depicts superclass Owner, with a PrivateOwner and BusinessOwner as subclasses. The specialisation/generalisation of the Owner entity is complete and disjoint (shown as {complete, disjoint}) as an Owner must

¹⁰An aggregation represents ‘has-a’ or ‘is-part-of’ relationships between entity types, where one represents the ‘whole’ and the other the ‘part’ (Connolly and Begg, 2004).

be either a private owner *or* a business owner, but cannot be both. Figure 2.9 also depicts two examples of aggregations, namely ‘Branch has Staff’ and ‘Branch offers PropertyForRent’. In both relationships, the Branch entity represents the ‘whole’ and therefore the open diamond shape is placed beside this entity. In addition, Figure 2.9 includes an *association class* that is connected to the association by a dotted line. This occurs as there are new attributes dateAdvert and cost that result from a many-to-many association between Newspaper and PropertyForRent (the maximum multiplicity in each direction is ‘many’). If there are no attributes that result from a many-to-many association, there is no association class (Jewett, 2002-2006b).

2.7.4 A Comparison of ER, ORM and the UML Class Diagram

While the NIAM/ORM approach may not be as popular as the ER/EER model, it is a semantically powerful conceptual model as it is fact-oriented in nature that allows for the capture of more business rules about the application domain in diagrammatic form and is not impacted by changes that causes attributes to be remodelled as relationships. Its conceptual schema are presented in an easy-to-read graphical form which can be easily understood by everyone involved in the design stage (Leung and Nijssen, 1987). Additional work by Leung and Nijssen (1987, 1988) also reflects on an additional strength of NIAM/ORM in that its well-formed conceptual schema can be transformed into an SQL Optimal Normal Form (ONF)¹¹ database schema.

Several researchers have also addressed the strengths of this model. Halpin and Proper (1995) stated that ORM has advantages over the ER model in terms of a populated and rich semantic notation. Laender and Flynn (1993) claimed that the NIAM approach is more expressive in terms of a rich set of integrity constraints. Song and Forbes (1991) also suggested that the NIAM approach is more rigorous in its definition of constraints on the data represented in the model. Additional work by Halpin (2001) concluded that the ORM method has several advantages over the ER and UML approaches with respect to the number of business rules that can be captured, greater stability when dealing with application domain changes and easier verbalisation and population.

¹¹ONF is a fifth normal form (5NF) with a minimum number of 5NF tables in the overall schema (Nijssen and Halpin, 1989; Halpin, 1998).

Table 2.2: Equivalent data structures in ER, ORM and the UML class diagram.

ER	ORM	UML
Entity type	Entity type	Class
Entity	Entity	Object (or Data value)
Attribute	— ^a	Attribute
Relationship type	Relationship type	Association

^aNo corresponding concept; ORM uses relationship types rather than attributes.

As discussed by Halpin and Bloesch (1999) and Halpin (1998, 2001, 2004), UML class diagrams are clearly superior to both ER and ORM for the detailed design of object-oriented code e.g. Java or C++ programs. However, they are less suitable for conceptual analysis since they lack a standard for the identification of schemata e.g. uniqueness constraints on attributes and external uniqueness constraints between association roles and attributes. In addition, UML does not predefine any data types (Halpin, 2000) and does not impose any compatibility constraints (Parent et al., 2006a), leaving these up to modellers to define their own type systems and notations for the constraints.

Although each model possesses a construct which another lacks, this thesis has found that both the ER and ORM models are significantly similar in their modelling power but distinctly different in their modelling methodologies (i.e. top-down versus bottom-up) and diagrammatic representation. Laender and Flynn (1993) also asserted that the similarities between the two models are that they both can be implemented by the relational model.

Each model has its own peculiarities and vagaries in the way that the products are used and implemented by designers/developers. Some prefer to use the ER model rather than the ORM due to its simplicity. However, as design work cannot be completed by only using the ER model, the designer must also possess sufficient understanding of normalisation techniques after applying a process to transform an ER schema to a relational schema. The use of these techniques removes the relationships between the attributes that cause redundancies in the relational database schema.

For beginners who lack normalisation knowledge, the ORM design method is a suitable substitute. The strength in the ORM design method is in its use

Table 2.3: Equivalent integrity constraints in ER, ORM and the UML class diagram.

ER	ORM	UML
Key attribute	Unique identifier (chosen from 1-1 reference type)	— ^a
Participation (total/partial)	Mandatory role	Multiplicity (1..1, 1..*/0..1, 0..*) ^b
Cardinality (1:1,1:N,N:1,M:N)	Uniqueness (Intra-fact-type, Inter-fact-type)	Multiplicity (1,*) ^c
— ^d	Subtype	Subclass
— ^e	Exclusion	XOR

^aNo standard notation.

^bThe multiplicities ‘1..*’ for mandatory-to-many and ‘1..1’ for mandatory-to-one while the multiplicities ‘0..1’ for optional-to-one and ‘0..*’ for optional-to-many.

^cEach role multiplicity consists of a specification of integer value, ‘*’ and ‘1’ are the most common. The asterisk ‘*’ represents an unlimited upper bound.

^dNo equivalent concept or term; this constraint is established by using subclasses in EER.

^eNo equivalent concept or term; this constraint is established by using disjoint classes in EER.

Table 2.4: Equivalent languages in ER, ORM and the UML class diagram.

ER	ORM	UML
GORDAS, GRAQULA GQL/ER, SQL/ER	ConQuer, RIDL	OCL, UQL
SQL (ER-to-relational mapping available)	SQL (ORM-to-relational mapping available)	SQL (UML-to-relational mapping available)

of well-defined transformation algorithms and elementary relationships. In other words, the ORM conceptual schema can be easily transformed into a fifth normal form relational database schema (Leung and Nijssen, 1987, 1988; Puntheeranurak and Chittayasothorn, 2002), with no resultant redundancy in the schemata.

However, a transformation of an ER schema only guarantees a first normal form relational database schema which may need further normalisation. This is a consequence of the way that relationship types in the ER model exist between entity types and thus the relationships between attributes which can cause redundancy in a schema are not considered.

The basic similarities regarding data structures, constraints and languages in

Table 2.5: The data models in ten historical eras.

Era	Data Model	Main Data Structure	Types of Data Model	Based on DBMS/Technology
late 1960s and 1970s	Hierarchical	Record type Parent-child relationship (PCR) type	Logical	IMS
1970s	Network (CODASYL)	Record type Set type	Logical	IDMS
1970s and early 1980s	Relational	Relation	Logical	RDBMS
late 1970s and 1980s	Semantic - ER - ORM/NIAM - UML	Entity type Relationship type Entity type Label type Fact type Reference type Class Association	Conceptual	—
mid 1980s and early 1990s	Object-oriented	Object Class Method	Logical	ODBMS
late 1980s and early 1990s	Object-relational	Object+Relation	Logical	ORDBMS
1980s and 1990s	Temporal relational	Temporal relation Timestamp attribute	Logical	RDBMS
late 1980s and 1990s	Graph	Graph	Conceptual	—
late 1990s and early 2000s	Multidimensional	Fact Dimension	Logical	ROLAP MOLAP HOLAP
late 1990s to present	Semi-structured - XML - RDF	Tree Graph (Subject-predicate-object)	Logical	Web

ER, ORM and UML class diagrams is summarised in Tables 2.2, 2.3 and 2.4, respectively.

2.8 Evolution of Data Models

Data modelling plays an important role in the process of information system development. The two main purposes of data modelling are to assist in the understanding of the meanings (semantics) of data and to facilitate communication about the information requirements (Connolly and Begg, 2004). Data models were first proposed in the late 1960s and since this time, a great number of new

and extended data models have been proposed.

The purpose of this section is to summarise this evolution of data models. Broadly speaking, the development of data models has occurred in distinctive eras as shown in Table 2.5. The principal models developed during each of these eras is described as follows:

Hierarchical. A hierarchical data model represents data as a *record type* that is arranged into a tree-like structure. The hierarchical data model supports two main types of structures: record types which are a collection of data items; and parent-child relationship (PCR) types which define a 1:N relationship between two record types. The most recognised example of a hierarchical model database is an Information Management System (IMS) designed by IBM, which was released in the late 1960s. An IMS database is a collection of instances of record types, such that each instance, other than root instances, has a single parent of the correct record type. Every record in an IMS database has a hierarchical sequence key (HSK). The IMS has a data manipulation language, DL/1, which is a ‘record-at-a-time’ language based on discrete and sequential record processing.

Network. The network data model represents data as *record types* and *set types*. Each set type defines a 1:N relationship between one instance of a record to many record instances using pointer¹² linking mechanisms. A record type can participate as an owner or member in any number of set types. A network data model organises a collection of record types, each with keys, into a directed graph, rather than a tree. Thus, a given record instance can have multiple parents, rather than just a single one, as is evident in the hierarchical model. This approach provides more flexibility than the hierarchical model, but the programmer still has to know the physical representation of the data to be able to access it. Similar to the hierarchical model, every time the structure of the database changes, the physical storage and application software also needs to be modified (Danielsen, 1998). The network data manipulation language also involves one-record-at-a-time processing. The standards for the network model were published in 1971 by the Conference on Data Systems Languages (CODASYL) Consortium and presented in the CODASYL Data Base Task Group (DBTG) report (DBTG, 1971).

¹²Pointers provide fast access, but they also embed certain forms of data access in the system, making it difficult to change the system to accommodate new requirements or to take advantage of new data structures (Raymond et al., 1996).

Relational. A relational data model (Codd, 1970) presents all data as *relations*, which can be accessed using a high-level non-procedural (set-at-a-time) language. The use of a high level language can provide a high degree of physical data independence. Hence, there is no need to specify a storage proposal, as was required in both IMS and CODASYL. As this model negates the need for pointers, the use of tables and records becomes much easier to understand. This makes the development of programs more effective and less dependent on changes in the physical representation of data. Most significantly, a set-at-a-time (high-level relational query) language such as SQL could give an efficient performance comparable to any one-record-at-a-time language.

Semantic. A semantic data model is a data modelling technique that is used to represent the meaning of data within the context of its interrelationships with other data (Hammer and McLeod, 1981; Hull and King, 1987; Peckham and Maryanski, 1988). A semantic data model is sometimes called a *conceptual* data model. The logical data structure of a database management system (DBMS), whether hierarchical, network, or relational, cannot totally satisfy the requirements for a conceptual definition of data as it offers little to aid database designers and users in interpreting the contents of a database. Consequently, the need for better modelling approaches to capture more of the semantics of an application to define data from a conceptual view has led to the development of semantic data models. Database designers can represent objects and their relationships in a natural and clear manner (similar to the way users view an application) by using high level abstraction concepts such as aggregation, classification and instantiation, subclass and superclass, attribute inheritance and hierarchies (Navathe, 1992). Examples of well-known semantic data models include:

- **the ER model** (Chen, 1976);
- **Object Role Modelling (ORM)** (Halpin, 2001) (Its precursor was called **NIAM** (Falkenberg, 1976; Nijssen, 1976, 1977; Nijssen and Halpin, 1989; Verheijen and van Bekkum, 1982)); and
- **the Unified Modeling Language (UML)** (Booch et al., 2005; Muller, 1999).

A detailed description of these models is previously given in Section 2.7.

Object-oriented. An object-oriented data model is an adaption of the object-oriented programming language paradigm for database systems that is used to represent a collection of objects that are organised into classes defined by complex values and methods (Kim, 1990). The model is based on the concept of an encapsulation of data and code in an object (Silberschatz et al., 1996). The behaviour of these objects is controlled by methods and each method consists of code that manipulates or returns the state of the object. As discussed by Bachman (1996), the major components of the object-oriented concept are as follows:

- objects (and their relationships);
- inheritance and multi-type objects;
- abstract data type and operator overloading;
- encapsulation and entity methods; and
- tight integration of the database and programming language.

Object-relational. An object-relational data model is a combination of the object-oriented and relational data models (Silberschatz et al., 1996). Based on the solid foundation of the relational model, this hybrid object-relational data model has an extended modelling power that supports object-oriented concepts in both the data schema and the query language. The introduction of abstract data types allows for attribute states to be defined by more complex types.

As discussed by (Malinowski and Zimanyi, 2008), this hybrid object-relational data model supports:

- complex and/or multi-valued attributes;
- user-defined types with associated methods;
- system-generated identifiers; and
- inheritance among types.

Object-relational database management systems (ORDBMSs) are used to support object-relational data models and are able to process the various object-oriented features such as inheritance, polymorphism, embedded object, complex objects, sets, lists and bags. ORDBMSs are mainly based on the criteria defined by Stonbraker et al. (1990) that provides suggestions on how to extend the capabilities of an RDBMS to include support for rich object structures and rules (Danielsen, 1998).

Temporal relational. A temporal data model includes timestamp attributes in its schema and provides special semantics for the values of these attributes for processing in its query language. A number of research efforts that have added time to the relational model have been proposed (Jensen and Snodgrass, 1996). A *temporal relational* data model is a relational data model that uses temporal relations as the underlying data structure, with operators that are all temporal (Böhlem, Busatto and Jensen, 1998). These temporal relations are also defined by temporal attributes.

Research interest into time modelling has increased dramatically in the 1980s and 1990s. A recent bibliography contained 331 temporal database papers from 1995 to 1998 (Wu, Jajodia and Wan, 1998). Since the advent of semantic data models that try to capture more real-world meaning than the relational model, a number of research proposals have appeared in the literature that have attempted to add temporal aspects into the ER model (Gregersen and Jensen, 1999) (refer to Chapter 3).

Graph. A graph data model is a model in which the data structures for the schema and/or instances are modelled as a directed (and possibly labelled) graph or as a generalisation of the graph data structure (Angles and Gutierrez, 2008). Data manipulation within the model is expressed by graph-oriented operations and type constructors. Additionally, appropriate integrity constraints can be defined over the graph structure (Angles and Gutierrez, 2008). Graph data models first appeared in the late 1980s for representing complex structures of knowledge called G-Base (Kunii, 1987), with more proposals appearing in the 1990s (Amann and Scholl, 1992; Gemis, Paredaens, Thyssens and den Bussche, 1993; Güting, 1994; Levene and Loizou, 1995).

Multidimensional. A multidimensional data model structures data into *facts* and *dimensions*. A fact can be considered as data of interest which contains some *measures*, and a dimension provides a set of attributes that characterise the dimension. Two common multidimensional schemata are the star schema and the snowflake schema. The star schema consists of a fact table with a single table for each dimension while the snowflake schema is a variation of the star schema in which the dimensional tables from a star schema are organised into a hierarchy through normalisation (Elmasri and Navathe, 2007).

Data warehouse and On-Line Analytical Processing (OLAP) systems are

based on a multidimensional data model. OLAP is a technology that processes data from a data warehouse into multidimensional structures to provide rapid response to complex analytical queries. The multidimensional model supports OLAP functionality, i.e. querying, restructuring, classification and summarisation.

OLAP technologies are used to implement a multidimensional model. Different architectures are used to store and process multidimensional data as discussed by Malinowski and Zimanyi (2008) as follows:

- *Relational OLAP (ROLAP)*. In ROLAP systems, multidimensional data is implemented as relational tables organised in specialised structures called star schemata or snowflake schemata. ROLAP supports extensions to SQL and special access methods to efficiently implement the multidimensional data model and the related operators. ROLAP systems provide better storage capacity than MOLAP systems.
- *Multidimensional OLAP (MOLAP)*. In MOLAP systems, multidimensional data is directly stored as special data structures, for example, arrays. OLAP operations are implemented over these structures. MOLAP systems provide better performance when processing multidimensional data queries.
- *Hybrid OLAP (HOLAP)*. HOLAP systems combine both technologies, benefiting from the storage capacity of ROLAP and the processing capabilities of MOLAP. For example, HOLAP systems may store large volumes of detailed data in a relational database, while aggregations are kept in a separate MOLAP store.

Multidimensional data models appeared in the late 1990s (Chaudhuri and Dayal, 1997; Franconi and Sattler, 1999; Gyssens and Lakshmanan, 1997; Golfarelli, Maio and Rizzi, 1998; Vassiliadis and Sellis, 1999) and then later into the early 2000's (Jensen, Kligys, Pedersen Bach and Timko, 2004; Lechtenbörger and Vossen, 2003; Martyn, 2004).

Semi-structured. A semi-structured data model is designed to model data with flexible structures, e.g. documents and web pages (Buneman, 1997). Examples of well-known semi-structured data models include XML and RDF.

- **XML.** The eXtensible Markup Language (XML) data model structures data as an XML document consisting of the two concepts of element and attribute (Elmasri and Navathe, 2007). Data is represented

using hierarchical tree structures, which are represented as elements. With the use of tags, data can be nested to create complex hierarchical structures (Elmasri and Navathe, 2007). XML was developed by an XML working group (originally known as the SGML editorial review board) formed under the auspices of the World Wide Web Consortium (W3C) in 1996 (Bray, Paoli, Sperberg-McQueen, Maler and Yergeau, 2006). As discussed in the W3C recommendation report (Bray et al., 2006), XML documents are made up of storage units called *entities*, which contain either parsed or unparsed data. Parsed data is made up of *characters*, some of which form *character data*, and some which form *markup*.

As discussed by Stonebraker and Hellerstein (2005), the data model presented in XML schema has XML records that can (a) be hierarchical, as in IMS, (b) have ‘links’ (references) to other records, as in CODASYL and semantic data models, (c) have set-based attributes, as with semantic data models, and (d) inherit values from other records in several ways, as with semantic data models. Hence, the structure of a document can be very complex.

- **RDF.** The resource description framework (RDF) is a model of *meta-data* that relies on the XML standard which has statements in the form of subject-predicate-object expression. It provides interoperability between applications that exchange machine-understandable information on the Web. The underlying structure of any expression in RDF is represented as sets of triples, each triple consisting of a subject, a predicate (verb) and an object of an elementary sentence (Fromm, Polikoff, Obrst, Daconta, Murphy and Morrison, 2005; Klyne and Carroll, 2004). Each triple represents a statement of a relationship between the subject and the object (Angles and Gutierrez, 2008). A set of such triples is called an RDF graph, which can be illustrated by a node and directed-arc diagram (Klyne and Carroll, 2004). In summary, RDF is a directed, labelled graph data format for representing information in the Web (W3C, 2008).

Table 2.5 (Page 51) presents the data model proposals in ten historical epochs, along with their main data structure, types of data model, and the DBMS or technology on which they are based.

2.9 Summary

Conceptual modelling is a semantically rich discipline aimed at capturing the meaning of an application domain at a high level of abstraction. The importance of conceptual modelling has been recognised by practitioners and researchers as it provides a plan for building a database that can be used to capture user requirements and to understand system complexity. Conceptual modelling is a notoriously difficult activity that can not be treated algorithmically and requires both ingenuity and experience (Badia, 2000).

The main focus of database technologies lies in the database design process. This process consists of three phases, namely conceptual, logical and physical design. The goal of conceptual design is to produce a high-level conceptual schema for a database that is independent of a specific DBMS configuration. To achieve this, conceptual (or semantic) data models are used and are expressed by means of diagrams with a rich set of modelling constructs. The most popular conceptual model for relational database design is the ER model and its extensions. The logical design maps this high-level conceptual schema to a logical schema formulated according to the data model of the DBMS used for implementation.

Chapter 3

ER Modelling Extensions: A Survey and Comparative Review

Over the past three decades since the original ER model was first published by Chen in 1976, the ER modelling approach has gained worldwide acceptance in database design, information system development and software engineering, and has been extended by several authors. In line with Objective 1 of this thesis as outlined in the Section 1.5 of Chapter 1, this chapter continues with the review of new and extended data models for conceptual modelling. It provides background information on the need for extensions to the ER model and delivers a survey on various proposed extensions to this model, with the objective of defining a survey framework proposals that can be used to categorise and compare the various proposals. From this study nine common aspects and four criteria have been identified that form a basic **Classification of ER Modelling Extension (CERME)** framework.

This survey is organised as follows. Initially, an overview is provided that introduces the central concept and key aspects of the CERME framework (Section 3.1). Next, the deficiencies and weaknesses of the original ER model are examined (Section 3.2). The proposals within each of the CERME aspects are then presented (Section 3.3). Using a common comparative framework, each proposal is then assessed and compared with other ER modelling extensions (Section 3.4). This chapter concludes with a summary of the lessons and understanding learned from the exploration of the CERME framework (Section 3.5).

3.1 Introduction

The ER model is the most influential conceptual model in the database community. As discussed by Chen, Song and Zhu (2007), many different extensions of the ER model have been developed in order to extend the original ER model to achieve more semantic power. The widely researched extensions have included the EER model (Elmasri and Navathe, 2007), the E²R model (Embley and Ling, 1989), the HERM approach (Thalheim, 2000), the *TIMEERplus* model (Gregersen, 2005), the starER model (Tryfona, Busborg and Christiansen, 1999) and the MDER model (La-Ongsri, Roddick and de Vries, 2008). These models have been categorised according to an ER modelling extension classification framework as described in the subsequent discussion.

Minimal research has been directed towards explaining and analysing the available modelling methodologies and their extensions. This author has not located any survey similar to that presented in this thesis that explores the area of extending the semantics of the ER model, and any comparative reviews of any ER modelling extensions based on the key criteria of (conceptual) data models.

This chapter presents such a survey and conducts a comparative review of various ER extension proposals published between 1976 and 2008. Using the results of this research, the information was analysed to identify trends or common themes within the survey results. These results showed that it was possible to classify these extensions according to the specific area that each extension was designed to deal with. These areas, or aspects, were able to be identified as structural, data abstractions, temporal, spatio-temporal, data warehousing, domain-specific applications, knowledge base, fuzzy data and XML data. As a means of providing an overall descriptive term covering all these aspects, the term Classification of ER Modelling Extensions (CERME) has been devised. Following on from the review of the various proposals, a comparison of the proposals under each main CERME aspects is presented. In addition, earlier survey and bibliography papers as well as books that are relevant to each CERME aspect are included.

The number of extensions to the original ER model that have appeared in the literature is estimated to be more than 100 (cf. Patig (2006)) and have been designed to deal with a wide range of applications requiring various levels of semantics. This survey of the ER modelling extension population was conducted to identify common extension aspects and criteria which can be used to categorise and compare all various proposals. The CERME framework has identified the

following nine aspects.

1. **Structural Aspect.** This category encompasses structural extensions of the ER model. This broad group covers a wide range of ER structures that are focused on overcoming the weaknesses of the basic ER structures (entity types, relationship types and attributes) in order to enhance its expressiveness. Extensions covering behaviours, events and constraints are also included in this category.
2. **Data Abstraction Aspect.** This category deals with abstraction mechanisms to capture (a) more complex relationships between entity types such as superclass/subclass relationships and union types, (b) complex objects and (c) category and inheritance and other items pertaining to the concepts of generalisation and specialisation. Although this category has much in common with the **structural aspect**, it is classed separately due to the relatively large number of extensions grouped under this category and also since many of the features are *object-oriented* in nature.
3. **Temporal Aspect.** This category discusses those extensions to the ER model dealing with *time*.
4. **Spatio-temporal Aspect.** This category deals with those modelling aspects that relate to *space and time* information.
5. **Data Warehousing Aspect.** This category presents the conceptual modelling of data warehouses that enrich the ER model with the features of *multidimensional* views.
6. **Domain-Specific Application Aspect.** This category includes extensions of the ER model that caters for the needs of modelling specific applications such as geographic data, multimedia, superimposed information, electronic commerce and manufacturing.
7. **Knowledge Base Aspect.** This category focuses on data modelling of knowledge in the ER/EER model that are expressed using natural language or inference rules.
8. **Fuzzy Data Aspect.** This category concentrates on fuzzy extensions to ER/EER models to represent uncertainty and imprecision in data and semantics at a conceptual level.

Table 3.1: Overview presentation of the CERME survey framework.

CERME Aspect	Proposal Name	Citation	Special Name	Based on Model
Structural	E ² R model	Embley and Ling (1989)	E ² R	ER
	Higher-order Entity-Relationship Model	Thalheim (1990, 2000)	HERM	ER
	MesoData Entity-Relationship model	La-Ongsri et al. (2008)	MDER	ER
Data Abstraction	Object base entity relationship approach	Spaccapietra and Parent (1992)	ECR+	ER
	Enhanced Entity-Relationship model	Elmasri and Navathe (1994, 2007)	EER	ER
Temporal	Time Extended EER model	Gregersen and Jensen (1998, 2004)	TIMEER	EER
	Time Extended EER model	Gregersen (2005)	TIMEER _{plus}	EER
Spatio-Temporal	Spatio-Temporal ER model	Tryfona and Jensen (1999)	STER	ER
	Modeling of Application Data with Spatio-temporal data	Parent et al. (1999, 2006)	MADS	EER
	DIStributed design of SpaTio-temporal data	Ram et al. (2001)	DISTIL	ER
Data Warehousing	Multidimensional Entity-Relationship model	Sapia et al. (1998)	ME/R	ER
	starER model	Tryfona et al. (1999)	starER	ER
	MultiDimER model	Malinowski and Zimanyi (2006)	MultiDimER	ER
Domain-Specific Application	Hypertext Design Model 2	Garzotto et al. (1994)	HDM2	ER
	Security Enhanced Entity-Relationship model	Oh and Navathe (1995)	SEER	EER
	Geographic Entity-Relationship model	Hadzilacos and Tryfona (1997)	Geo-ER	ER
Knowledge Base	Knowledge-based Entity-Relationship model	Kerschberg et al. (1990)	KORTEX	EER ^a
	Deductive Entity-Relationship model	Han and Li (1992)	Deductive-ER	ER
	Refined Entity-Relationship model	Shimazu et al. (2003)	RER	ER
Fuzzy Data	Fuzzy ER model	Zvieli and Chen (1986)	Fuzzy ER	ER
	FuzzyEER model	Galindo et al. (2006)	FuzzyEER	EER
	MesoData Entity-Relationship model	La-Ongsri et al. (2008)	MDER	ER
XML Data	EReX model	Mani (2004)	EReX	ER
	XSEM-ER model	Necasky (2007)	XSEM-ER	ER

^a EER here is the Extended Entity-Relationship model described by Teorey et al. (1986)

9. XML Data Aspect. This category includes the current research on the conceptual models for XML based on the ER model.

Where possible, previous surveys, bibliographies and book references that provide a substantial contribution to each CERME aspect are referred to in each of the later discussions.

Table 3.1 provides an overview of the CERME framework that covers each of these CERME aspects, along with their proposal name, main citation, special name (or identifier), and the models on which they are based. All of these CERME aspects can be attributed to increasing expressiveness of the ER model. By introducing new constructs to the basic ER model, increased knowledge and understanding can be captured through the addition of these new constructs that can be assigned particular semantics.

This study provides a summary of 32 years worth of research into ER modelling extensions as proposed in the literature. This survey systematically analyses the various aspects of each reviewed ER modelling extension, categorises each proposal according to its CERME aspect and presents a comparative summary of the features of each proposal. A detailed description of some of the proposals is given in Appendix B.

3.2 Limitations of the ER Model

Both the classical ER model and other database models have significant inherent limitations, so it is expected that a number of extension versions will be required to overcome these deficiencies. Research into the basic limitations of the ER model prove useful in raising awareness of these faults and to create the impetus to consider approaches that can overcome these limitations. These ideas and contributions can thus lead to the development and delivery of tangible specifications of ER model extensions.

Research into the limitations of the ER model has been covered by a number of researchers including Badia (2000, 2004), Embley and Ling (1989), Kroenke and Gray (2006), Rolland and Cauvet (1992), Shekhar et al. (1999), Thalheim (2000) and Chen (2006).

In the research by Badia (2000, 2004), it was pointed out that the basic components of the ER model can only be combined in certain ways, not freely. Specifically, these constraints dictate that only entities and relationships can have attributes and only entities can be associated with relationships. Thus, the limitations of the ER model can be summarised as below:

- Attributes cannot be defined to have attributes.
- Attributes cannot be associated with relationships.
- Relationships cannot be associated with other relationships.

Additionally, Thalheim (2000) presented the deficiencies of the ER model as follows:

- The ER model is unable to represent hierarchical and higher-order relationships. Only first-order relationships can be modelled.

- Is-a relationship cannot be modelled naturally.
- The concept of weak entities is not theoretically based.
- The classical ER model does not use n -ary relationship.
- The basic approach (Teorey, 1990) in defining new entities as clusters of entities and relationships leads to a loss of information.
- Sets, sequences and null-valued relationships cannot easily be represented.
- During database design, the type system requires the introduction of artificial and abstract types that do not carry any semantics in the application. For example, if relationship types are restricted to binary types then n -ary types ($n > 2$) are represented by entity types and connecting relationship types that are not independent and do not have their own significance.
- ER concepts often lack a clear statement of purpose for semantics. This can result in different semantics being applied to the same concept, and the intermixing of semantics of different constructs.

Embley and Ling (1989) further discuss the limitations of both the ER and EER models as follows.

- The ER/EER models require designers to distinguish between attributes and entities. This can cause downstream redesign to accommodate schema integration such as where attributes and entities are mismatched or where there is a need to accommodate any subsequent discovery of relationships among items designated as attributes.
- Design work can not be completed in the ER model alone, and thus designers have to use two different types of abstraction. In the first instance, designers work with the ER diagrams. After applying a process to map the ER schema to a relational schema, they work with relational schemata using normalisation techniques in order to extract functional and join dependencies from the real world semantics.

In the research by Rolland and Cauvet (1992), it was suggested that traditional conceptual models are limited by their emphasis on modelling static aspects of the real world, thus requiring dynamic aspects to be integrated into static conceptual models. Kroenke and Gray (2006) stated that human beings

do not naturally consider and rationalise the relationships between entities and their cardinalities in the same way that they are represented in the ER model. It becomes unrealistic to expect users to understand the cardinality of relationships unless the ER model can accurately reflect user perceptions. A further limitation has been identified by Shekhar et al. (1999) who discussed that the ER model is unable, at least intuitively, to capture some important semantics inherent in spatial modelling.

The general practice of using conceptual models has proven that the underlying theory and constructs are sound and have been effective in capturing the basic semantics of applications. As applications become more complex and sophisticated, such as in handling fast time-varying and time-dependent changes in world states, modelling modifications and extensions become essential in order to overcome the inadequacies of the basic model. On reflection of his own modelling approach, Chen (2006) has identified the following weaknesses of existing conceptual modelling, methodologies and techniques that need addressing:

- Using the constructs of existing conceptual models, it is very difficult to model a wide spectrum of situations resulting from different degrees of importance of relationships due to different perspectives.
- Current state-of-the-art techniques focus on pre-defined entities of interest and their static relationships.
- As current database/knowledge-based systems only model snapshots of part of the world of interest, there is no support for information and schema changes or the storage of historical information.
- Virtually no constructs in the existing conceptual models are available for modelling changes of the entity behaviours (e.g. weather pattern changes) and the dynamic and time varying relationships between them.
- The schemata of the current data models are difficult to change dynamically.

The restrictive problems in the ER model or conceptual models are significant and should be addressed through the enhancement or extensions of these original models so as to produce better conceptual modelling methodologies and techniques.

3.3 Exploring ER Modelling Extensions

The original ER model (Chen, 1976) has attracted much research over the last 30 years and has been extended by more than 100 proposals (cf. Patig (2006)). In the past decades, many papers on extending and modifying the original ER model have been presented at the annual International Conference on Conceptual Modeling (the ER conferences). These have also been posted on the main forum for conceptual models and modelling at <http://www.conceptualmodeling.org/>.

In the exploration of ER extensions, this thesis has referred to the main series of relevant conferences (ER, VLDB, ACM, SIGMOD, ICDE) and journals (ACM Transactions on Database Systems, ACM Computing Surveys, ACM SIGMOD Record, Communications of the ACM, Data & Knowledge Engineering, IEEE Transactions on Knowledge and Data Engineering, Information Systems, Information and Software Technology, Information and Management, Journal of Database Management) up until 2008. Additionally, publications relating to the enhancement of ER models were accessed through various sources such as the scientific literature digital library (<http://citeseer.ist.psu.edu/>), the scholarly search engine (<http://scholar.google.com/>) and the ACM digital library (<http://portal.acm.org/>). Relevant textbooks, technical reports and various other papers were also included in the study.

For each of the nine CERME aspects, various proposals that extend the ER model are examined and discussed. This examination covers a total of 23 proposals. Whilst the sampling of these proposals was unbiased, the selection techniques used aimed to ensure that all of the CERME aspects were covered equally. Some of these proposals are described in detail in Appendix B. Earlier survey studies, published bibliographies and books that are relevant to each of the CERME aspects are also included to provide details on trends, ideas, interesting views and relevant research for further reference. Where the subject matter of a proposal spans several CERME aspects, it will appear in multiple categories. For example, the MDER model (La-Ongsri et al., 2008) appears under both the fuzzy data and structural aspects.

3.3.1 Structural Aspect

There is a large variety of structural extensions of the ER model whose main aim is to attempt to overcome the weaknesses of the ER model with regard to its basic structures (entity types, relationship types and attributes). Other extensions

relevant to structures include (a) behaviours (Schrefl, 1991), (b) events such as the Deterministic Event-Tuned Entity-Relationship Modeling (DETERM) approach (Falkenberg, 1993) that enhanced the ER model by considering not only static phenomena, but also dynamic phenomena of the universe of discourse, viz. events, and (c) constraints such as the Methods enhancement of the EER (MEER) model (Balaban and Shoval, 2002) that extends the EER model with structure-based update methods which are fully defined by cardinality constraints.

The limitation of the ER/EER model proposed by Embley and Ling (1989) is subject to the following two problems; the ER/EER models require designers to distinguish between attributes and entities, and the database design cannot be completed by using the ER model alone. Thus, designers have to use two different types of abstraction, namely, the ER model and a transformation to a relational schema which may then need normalisation. These problems were addressed in an improved ER approach called E²R (Embley and Ling, 1989).

The remaining weaknesses of the ER model are concerned with the way that those structures are built and their composition. For example, the classical ER model does not support the view of having attributes over attributes, relationship types over attributes, or relationship types over relationship types. To deal with the limited power of the model in supporting these basic *relationship types*, various structural extensions to the ER model have been suggested. Tu and Wang (1993) developed the Attribute-Relationship (AR) extension that allowed relationships to be built at the attribute level. Thalheim (1990, 2000) presented the Higher-Order Entity-Relationship Model (HERM) to allow for the definition of new relationship types based on existing relationship types. Badia (2000) proposed the concepts of Generalised Quantifiers (GQs) that provided for higher-order operators which allowed relationships to be involved in other relationships. Limitations of the ER model concerning *relationship types* were also discussed by Camps Pare (2002) and Badia (2004). Their discussion indicates that there are properties of ternary relationships that can not be represented in the ER model nor captured in a relational model in the form of a key or integrity constraint.

In addition, recent extensions to the ER model have been suggested to deal with its structural limitation with regard to *entity types*. For example, Jiménez (2006) promoted the Reenhanced Entity-Relationship Model (REERM) that re-defines the concept of entity and adds the construct event.

A further limitation of the ER model with regard to *attributes* is that there is *no support for data domains* (Kroenke and Gray, 2006). The need to capture

more domain semantics is growing due to the requirement to support more complex and sophisticated applications. When designing models, basic attributes are assigned to base data types. Complex attributes can be constructed by applying constructors such as list, set and bag (also called multisets) to attributes that have already been constructed. Most of these extensions are based on the notion of complex types of attributes.

However, a recent extension to attribute domains, the Mesodata Entity-Relationship (MDER) model (La-Ongsri et al., 2008), uses complex *domains* based on complex structures of mesodata type, such as tree and graph, to provide more advanced semantics to the domain of an attribute, instead of using complex *attributes* like other approaches. This maintains the usual basic attribute data types such as integer, string and real whilst also accommodating attributes whose values are based on complex domains. It also provides greater versatility by allowing domains of an attribute in MDER to be defined according to an organisation's data type standards when data models are created.

Examples of ER modelling extensions with this CERME aspect include the E²R, HERM and MDER models that are described below:

- 1. The E²R (E-squared-R) model.** The E²R proposal (Embley and Ling, 1989) is based on the ER model and includes notions of generalisation/specialisation and lexical entity types. In this model, designers do not need to distinguish between attributes and entities and it also supports normalisation at the model level. Synergistic database design occurs as a designer interactively manipulates an E²R diagram until the desired properties are attained. Design is performed by transforming a given E²R model into a normalised E²R model that is guaranteed to generate normalised relations. The main steps in this approach can be summarised as (a) capture the real-world semantics in an improved EER model, (b) transform the EER model into a normalised EER model, and (c) generate the normalised relations.

E²R modelling treats an attribute as a point of view rather than a basic model construct and thus does not prematurely impose structure on the database being designed. That is, time consuming decisions on whether to use an entity type or attribute become inconsequential. Extensions to the ER model that are necessary to add additional semantic-modelling power, such as participation constraints, generalisation/specialisation, aggregation/decomposition and power-sets, are included in the E²R model. The implementation of this approach allows a user to store the E²R model

with constraints that include both functional and multivalued dependencies. Also, transformations allow a user to remove or reduce relationship types that have redundancy.

- 2. The Higher-Order Entity-Relationship Model (HERM).** The HERM methodology (Thalheim, 1990, 2000) extends the ER model by using relationships of higher degrees and relationships of relationships. This approach introduces the new concept of higher-order relationships which are additional generalisations of the ER constructs to handle both the multiple entity set participation in a given role of the relationship, as well as the subclass, superclass and clustering concepts. This concept further extends the scope and concepts related to data abstraction and incorporates a supporting set of operations and integrity constraints. HERM algebra is introduced that defines the available manipulation operations in the model. HERM uses set semantics for the definition of entity instances and relationship instances and also supports a mechanism for the translation to relational, network and hierarchical schemata.
- 3. The Mesodata Entity-Relationship (MDER) model.** The MDER model (La-Ongsri et al., 2008) is a new approach proposed in this thesis (Chapter 4) that extends the ER model and is also applicable to the EER model. This approach provides richer semantics to attribute domains by incorporating the complex domain structures using mesodata concepts. New constructs and constraints are proposed to capture domain semantics of an attribute. The MDER model is based on the mesodata concept (de Vries and Roddick, 2004; de Vries, Rice and Roddick, 2004). This approach provides the *mesodata languages* (MDDL and MDML), which are extensions to SQL to define and manipulate the structure of domains (mesodata types). The mechanism to transform a MDER schema to a relational schema is also discussed (Chapter 8).

This thesis survey has identified the following relevant resource that covers the ways that the ER model has been extended to enhance its **structural aspect**.

Thalheim's book (2000) provides a theoretical basis for database modelling and proposes a new ER model extension termed the HERM model. This model supports relationships with higher degrees and relationships of relationships. The model allows the specification of structures, behaviour and interaction. The book also presents techniques for the translation of the ER model into object-oriented

models and classical database models such as relational, hierarchical and network models.

3.3.2 Data Abstraction Aspect

One of the most popular ER modelling extensions is the Enhanced Entity-Relationship (EER) model proposed by Elmasri and Navathe (1994, 2007). This model adds abstraction mechanisms such as superclass, subclass and category, including attribute and relationship inheritance. These properties favour an *object-oriented* approach. Other works include the Entity-Category-Relationship (ECR) model (Elmasri, Weeldreyer and Hevner, 1985) that presents the concept of categories to handle both multiple entity set participation in a given role of a relationship, as well as subclass and superclass concepts, including multi-valued attributes.

Other contributions add object-oriented features (e.g. generalisation and inheritance, enforcement of the information hiding principle, abstract data type encapsulation and message passing) in order to adapt the original ER model to object-oriented database design. Examples include the Object-Oriented ER (OOER) model (Navathe and Pillalamarri, 1988), the Behavior-Integrated ER (BIER) model (Kappel and Schrefl, 1988), the Object-Oriented ER Model (OO-ERM) (Gorman and Choobineh, 1991) and the object base entity relationship (ERC+) approach (Spaccapietra and Parent, 1992). A summary of some of these models can be found in the survey presented by Saiedian (1997).

The proposals of ERC+ and the most recent publications of EER models are described below:

1. **An object base entity relationship (ERC+) approach** . The ERC+ approach (Spaccapietra and Parent, 1992) is an extended entity-relationship model specifically designed to support complex objects and object identity. Two generalised relationships are supported, the classical **is-a** and an additional **may-be-a** relationship. ERC+ thus provides for the choice between optional and mandatory status (for attributes, for roles, and for generalisations, using **may-be-a** relationships). Formal definitions and Data Manipulation Languages (DML) of the ERC+ model are also discussed.

The ERC+ algebraic query language is presented and consists of ten primitive operations which may be combined in any order to form algebraic expressions. Using this algebra, entity types with complex attributes are constructed. Tools to support the graphical description of database schemata

and for graphical data manipulation are presented to aid the design and development of database applications. A method of mapping from ERC+ to the relational model and object-oriented model is also demonstrated.

- 2. The Enhanced Entity-Relationship (EER) model.** The EER model (Elmasri and Navathe, 2007) enhances the ER model by incorporating the concepts of superclass/subclass (or generalisation/specialisation), superclass/subclass relationships, type inheritance (including various types of constraints on specialisation/generalisation) and category. The concept of category is used to deliver the UNION construct.

A class/subclass relationship proposed in EER is often called an **is-a** relationship providing generalisation/specialisation. The constraints of total/partial and disjoint/overlapping can be applied to specialisation, generalisation and category. The EER-to-relational mapping algorithm is provided that can be used to create a relational schema from an EER schema.

This thesis survey has identified the following relevant resources that cover the ways that the ER model has been extended to enhance its **data abstraction aspect**.

Survey Paper:

Saiedian (1997) surveyed major extensions to the ER model, in particular, enhancements related to generalisation and object-oriented adaptations. This research also provides an overview of ER models and points out the close relationship between ER modelling and object-oriented data modelling. This research has led to the adoption of object-oriented design techniques that have significantly improved the modelling power of the original ER model. These object-oriented features include generalisation and inheritance, enforcement of the information hiding principle, abstract data type encapsulation and message passing. Saiedian's research discusses three object-oriented ER models (the OOER, BIER and OOERM) that further explores this CERME aspect.

Book:

Elmasri and Navathe's book (2007, 5th edition) introduces the fundamental concepts necessary for designing, implementing and using database systems and applications. It presents clear explanations of theory and design, broad coverage of models and real systems, and an up-to-date introduction to modern database systems and technologies. Sections of this book focus on data abstractions, semantic data modelling concepts and an integration of those concepts leading to

the EER model and EER diagrams. A discussion about relational database design using ER-and EER-to relational mapping is also presented.

3.3.3 Temporal Aspect

Temporal aspects of the real world are pervasive and important for most applications. Not surprisingly, several enhancements to the data models have been proposed in an attempt to support the modelling of time. Time has been incorporated into many data models, such as semantic data models, object-oriented data models, relational data models and deductive databases. As this thesis survey is focused on the ER model, the attention of this survey is only on those ER model extensions that facilitate the capture of time-varying information.

The modification of the ER model to capture time-varying information has gained increasing popularity within the database research community over the last two decades. A number of the temporally enhanced ER models have been presented, for example:

- the Temporal Entity-Relationship Model (TERM) (Klopprogge, 1981; Klopprogge and Lockeman, 1983);
- the Relationships, Attributes, Keys, and Entities (RAKE) model (Ferg, 1985);
- the Model for Objects with Temporal Attributes and Relationships (MOTAR) (Narasimhalu, 1988);
- the Temporal EER (TEER) model (Elmasri and Wu, 1990; Elmasri et al., 1993);
- the Semantic Temporal EER (STEER) model (Elmasri et al., 1990; Elmasri and Kouramajian, 1993);
- the Entity-Relation-Time (ERT) model (Theodoulidis et al., 1991*a,b*) and its refinement (McBrien, Seltveit and Wangler, 1992);
- the Temporal ER (TER) model (Touzovich, 1991);
- the TempEER model (Lai et al., 1994); and
- the TERC+ model (Zimanyi et al., 1997).

All of these models are extensions to the ER/EER model. For a detailed description of all the above models, refer to the comprehensive survey by Gregersen and Jensen (1999). This thesis examines a number of more recent models that capture temporal aspects of information, such as those detailed below:

- 1. The Time Extended EER (TIMEER) model.** The existing temporal ER models represent quite diverse approaches to capture temporal aspects of data at the conceptual level. The TIMEER approach (Gregersen and Jensen, 1998, 2004) retains the existing ER constructs and their associated semantics and simply makes them temporal. The model introduces new temporal constructs that provide implicit temporal support and includes snapshot-reducible attribute types (temporal single valued, temporal multi-valued, temporal composite and temporal derived attribute types) as well as snapshot-reducible participation constraints. The model also supports both valid time and transaction time including lifespans.

The TIMEER model extends the EER model to support four distinct types of temporal aspects, namely, valid time, lifespan, transaction time and user-defined time. Two additional participation constraints, snapshot and lifespan, are also proposed and the method of mapping the TIMEER model to the relational model is also presented. The research does not, however, provide for a query language that can be used to query the temporal ER database.

- 2. The Time Extended EER (TIMEER_{plus}) model.** The TIMEER_{plus} (Gregersen, 2005) model is a recent refinement of the TIMEER model offering enhanced modelling constructs applicable to attributes in four aspects; time sequence attributes, an update pattern for attributes, an observation pattern for attributes and an explicit notation for specifying changes to the database schema.

TIMEER_{plus} extends TIMEER by including time sequence attributes, update and observation patterns for attributes (Jensen and Snodgrass, 2000) and schema changes (Roddick, Craske and Richards, 1994). All modelling structures and integrity constraints are the same as those of the TIMEER model, but no query language for this temporal ER database is provided.

This thesis survey has identified the following relevant resources (listed in alphabetical order by first author) that cover the ways that the ER model has been extended to enhance its temporal aspect.

Survey Papers:

Gregersen and Jensen (1999) surveyed ten temporally enhanced ER models, provided a comprehensive list of 19 design properties of temporal ER models, and evaluated the models according to those properties. Their research explores different ways of conveniently capturing the temporal aspects of data at the conceptual level and consolidates these concepts and ideas so as to assist with future research in temporal ER modelling. As both their research and this thesis are focused on surveying extensions to the ER model, their research provides valuable reference material that complements this thesis survey. Rather than re-examine these previously proposed ER model extensions, this thesis refers the reader to Gregersen and Jensen's survey.

Özsoyoğlu and Snodgrass (1995) surveyed many temporal and real-time data models. Their research covers the time domain, temporal queries, real-time data and query languages, and temporal and real-time DBMS implementation. Their survey attempts to capture and summarise the major concepts, approaches and implementation strategies that have been discovered through temporal and real-time database research. The evaluation of temporal and real-time query languages along several dimensions are also discussed. Their work mainly examines the inclusion of time into relational data models and object-oriented data models. While their research does not consider those ER model extensions that incorporate time, it does cite a reference to some of these extensions.

The survey by Roddick and Patrick (1992) covered those information systems that incorporated the concept of time and identified potential impacts on temporal data modelling, artificial intelligence and various practical implications of these time concepts. Their research investigates the handling of time in data modelling which includes only two of the temporal ER models, the TERM and RAKE models. Their research does not consider the evaluation and comparison of the models.

Roddick and Spiliopoulou (2002) surveyed various aspects of temporal data mining and reviewed research contributions that are related to temporal knowledge discovery. Their survey includes a discussion of temporal rules and their semantics, temporal mining environments and the discovery of temporal rules that can identify key items of interest. As their research is more directed towards the area of using temporal databases for data mining, its value for data modelling is limited.

Theodoulidis and Loucopoulos (1991) surveyed nine approaches to specify

and use time in conceptual modelling. The approaches were evaluated using a devised comparative framework based on time semantics, model semantics and temporal functionality. Their research included only two of the temporal ER models, namely, TERM and the Entity, Relationship, Attribute, Event (ERAE) model. The likely reason for this is that the focus of the research was on the investigation of ontologies and the properties of time in the broader context of information systems and conceptual modelling.

Bibliographies:

Wu et al. (1998) collected 331 temporal database papers, most of which were published between 1995 and 1998. Adding to the six previous bibliographies (Bolour, Anderson, Dekeyser and Wong, 1982; Kline, 1993; McKenzie, 1986; Soo, 1991; Stam and Snodgrass, 1988; Tsotras and Kumar, 1996), this is the seventh bibliography concerning temporal databases. Their bibliography adopts a different classification method that divides papers into Models, Database designs, Query languages, Constraints, Time granularities, Implementations, Access methods, Real-time databases, Sequence databases, Data mining, Concurrency and Other papers.

Books:

A recent book devoted to temporal data and relational model is by Date, Darwen and Lorentzos (2003). This book provides an in-depth description of the foundations and principles on which temporal DBMSs are built. These foundations and principles are firmly rooted in the relational data model.

Snodgrass's book (2000) provides a general introduction and extensive coverage of temporal data with a great number of examples from real application database systems that have been designed and built to record information over time.

3.3.4 Spatio-Temporal Aspect

The spatio-temporal concept is created by combining the concepts of space and time, and in practical terms, this is achieved through recording spatial views (e.g. objects and layers) in time (e.g. time point and time interval). As described by Pelekis, Theodoulidis, Kopanakis and Theodoridis (2004), the development and research into spatio-temporal databases started in the early 1990s and has principally dealt with applications characterised by both spatial and temporal semantics. Some research has concentrated on the conceptual modelling of geographical

applications, mainly dealing with space, location and dimensionality of objects, spatial relationships and space-depending attributes, such as the Geo-ER model (Hadzilacos and Tryfona, 1997). For a summary of the Geo-ER model refer to the **domain-specific application aspect** (Page 81).

Since the spatial and temporal database models were integrated into the spatio-temporal database models, a number of extensions have been proposed that use object-oriented approaches. These have included Spatio-temporal UML (STUML) (Price, Ramamohanarao and Srinivasan, 1999), Extended spatio-temporal UML (Ext. UML) (Price, Tryfona and Jensen, 2000), an object-oriented data model for geographic applications (OMT-G) (Borges, Davis Jr and Laender, 2001) and the Tripod spatio-historical data model (Griffiths, Fernandes, Paton and Barr, 2004). As these models are extensions to the UML and Object Modeling Technique (OMT) methodologies, they are outside the scope of this thesis survey.

Examples of the extensions of the ER/EER model that include space and time are described below:

- 1. The Spatio-Temporal ER (STER) model.** The STER model (Tryfona and Jensen, 1999) is an extension of the basic ER model that includes spatio-temporal entities, attributes and relationships for modelling spatio-temporal information. STER offers support for the spatial data types (point, line and regions) and geometries (and various combinations thereof), and for temporal aspects (existence time (**et**) for objects, valid time (**vt**) for attributes and relationships, and transaction time (**tt**) for all constructs). Support for both valid time and transaction time is represented as bitemporal (**bt**). However, STER omits to provide integrity constraints or any supporting language.
- 2. The Modeling of Application Data with Spatio-temporal features (MADS).** The MADS approach (Parent et al., 1999, 2006a) is a conceptual spatio-temporal model based on the EER model. It caters for multi-associations, a larger number of semantic descriptors for relationship types (in particular, generation and transition) and supports diversity in the way constraints are handled. The orthogonality principle is an important aspect of MADS that adds different modelling dimensions (i.e. spatial and temporal characteristics). MADS also allows for additional multi-representation functionality resulting in further potential model extensions (Parent, Spaccapietra and Zimányi, 2006b). However, MADS does not yet support transaction time. Integrity constraints, query languages and the translation

process to facilitate an implementation of current DBMSs and GISs are provided.

3. DIstributed design of SpaTio-temporaL data (DISTIL). DISTIL (Ram, Snodgrass, Khatri and Hwang, 2001) is another recent design tool that assists in the design and modelling of spatio-temporal databases. DISTIL classifies spatio-temporal conceptual design into two steps:

- Firstly, capture the current reality of an application using an ER based conventional conceptual model, without consideration of any spatial aspects.
- Secondly, annotate the schema with spatial and temporal semantics of the application.

DISTIL is an annotation-based approach to spatio-temporal conceptual modelling that captures various aspects related to temporality and spatiality such as valid time, transaction time, events, states, position, geometry and shape. In particular, DISTIL provides a mechanism to capture semantics related to granularity and indeterminacy. Furthermore, schemata developed via DISTIL can be saved as XML schemata (Khatri, Ram and Snodgrass, 2006). However, a detailed description of the associated constraints and languages is not covered in their proposal.

This thesis survey has identified the following relevant resources (listed in alphabetical order by first author) that cover the ways that the ER model has been extended to enhance its **spatio-temporal aspect**.

Survey Papers:

Abraham and Roddick (1999) surveyed spatio-temporal models proposed in the literature and also included other aspects covering representation, spatial access methods, conceptual models, spatial database languages, scaling and accuracy issues, query optimisation and visualisation. However, no ER model extensions that capture spatio-temporal information are introduced in their survey.

Friis-Christensen, Tryfona and Jensen (2001) surveyed six existing proposals for the modelling of geographic and spatio-temporal data of which five are based on the object-oriented approach and the remainder, STER, that is based on the ER model. Their research presents a list of requirements that is used to identify critical properties of geographical data models. These surveyed models

are assessed using the list of requirements and a summary of the evaluation is presented.

Pelekis et al. (2004) surveyed different types of spatio-temporal data models and used a comparative framework to evaluate the benefits of each approach. An overview of notable achievements within spatio-temporal database research together with suggestions for future spatio-temporal database research are also provided. Their research discusses the extended ER model, STER, which caters for the capture of spatio-temporal information.

Wang, Zhou and Lu (2000) surveyed various aspects of spatio-temporal data management covering data models, indexing structures, query evaluation and architectures. Their research only refers to one of the spatio-temporal ER models, STER.

Bibliographies:

Al-Taha, Snodgrass and Soo (1994) collated a number of reference papers that consider spatio-temporal aspects. The bibliography is an updated version of the previous bibliography produced by Al-Taha, Snodgrass and Soo (1993).

Roddick, Hornsby and Spiliopoulou (2001) collated those papers primarily concerned with *mining* temporal, spatial and spatio-temporal data. As the bibliography uses broader categories covering Theses, Surveys, Books and Previous Bibliographies, it provides valuable reference material. Additional considerations and directions for further research into spatial and temporal databases can be found in Roddick, Hoel, Egenhofer, Papadias and Salzberg (2004).

Books:

Ott and Swiaczny's book (2001) deals with the integration of temporal information in Geographical Information Systems (GIS). It reflects upon theoretical ideas on the interrelations between space and time and includes practical examples taken from various types of applications (spatial/environmental analysis, demographics, history and business data warehousing).

Parent et al.'s book (2006*a*) focuses on modelling spatial and temporal information, presenting the MADS data modelling approach that covers both data modelling and data manipulation features. This presented material serves as a valuable reference in its discussions of how these concepts relate to the traditional data modelling approaches. Visual notations and examples are extensively used to illustrate how various constructs can be used. The book is of major importance and interest in the areas of spatio-temporal databases and geographical information systems.

3.3.5 Data Warehousing Aspect

From the discussions by Malinowski and Zimányi (2006), the structures of data warehouses (DWs) are re-examined using a *multidimensional* view of data including dimensions, hierarchies and measures. These concepts are applied to OLAP systems that allow for interactive data warehouse querying using operations such as drill-down and roll-up, which need hierarchies in order to automatically aggregate the measures to be analysed.

It is well known that data warehouses are centred upon decision support rather than transaction support, and that they are based on multidimensional modelling requiring specialised design techniques. This is reflected in the literature which has focused on the multidimensional modelling facets of conceptual modelling for data warehouses. Current research is thus directed towards advanced multidimensional features with proposals that have extended UML (Abelló, Samos and Saltor, 2006; Luján-Mora, Trujillo and Song, 2006) and the ER model (Malinowski and Zimányi, 2006; Sapia, Blaschka, Höfling and Dinter, 1998; Tryfona et al., 1999). A more recent model, the Generalising Conceptual Multidimensional Data (CGMD) model, extends the ER model for data warehouses through additional constructs of aggregated entities that allow for their interrelationships with the other parts of the schema (Kamble, 2008).

Examples of extensions of the ER model that support multidimensional views of data are described below:

1. The Multidimensional Entity-Relationship (ME/R) model. The multidimensional element plays a major role in the design of a data warehouse. ME/R (Sapia et al., 1998) extends the ER model to express the multidimensional structure of the data. ME/R includes two specialised relationship types and a specialised entity type allowing the adequate conceptual representation of the multidimensional data view to be compliant with OLAP schemata. These are as follows:

- a special entity type, termed the dimensional level,
- two special relationship types connecting dimensional levels:
 - a special n -ary relationship type, termed a *fact* relationship type,
 - a special binary relationship type, termed a *classification* relationship type.

However, the ME/R model does not discuss the features of the models with regard to integrity constraints and languages.

- 2. The starER model.** StarER (Tryfona et al., 1999) addresses the modelling requirements of a data warehouse and incorporates the star structure into the constructs of the ER model. Their research presents new special relationship types for hierarchies and includes an evaluation of the starER model. Examples from a mortgage data warehouse environment are used to illustrate how the model can be used to represent complex information at the conceptual level. However, starER omits to define the features of the model to address integrity constraints and languages.
- 3. The MultiDimER model.** MultiDimER (Malinowski and Zimányi, 2006) is a conceptual multidimensional model based on the ER model that includes constructs for data warehouse and OLAP modelling. The model offers some important features for representing different kinds of hierarchies, levels and fact relationships. Both graphical and textual notation, as well as a formal definition are included. Hierarchy features are important for analysis and form part of the advanced features for a multidimensional model. The model provides exclusive integrity constraints and introduces a mechanism to allow for the mapping of these hierarchies to the relational model. However, a language for the model is not provided.

In addition to the multidimensional view of data, a temporal extension of the MultiDim model (Malinowski and Zimanyi, 2008) has been recently introduced and provides temporal support (valid time, transaction time and lifespan) for levels, attributes, hierarchies and measures. Refer to Malinowski and Zimanyi (2008) for a detailed discussion on the implications of this proposal on temporal data warehouse design.

As new data warehouse applications and architectures move towards web-based technologies, data modellers and designers must deal with the challenges of automating the conceptual design process when some or most data sources reside on the web (Rizzi et al., 2006). Some attempts have been made in this direction, mainly aimed at building a web warehouse conceptual schema from XML data (Vrdoljak, Banek and Rizzi, 2003).

This thesis survey has identified the following relevant resources (listed in alphabetical order by first author) that provide the necessary data modelling and

design techniques for the data warehousing aspect.

Books:

Imhoff, Galemno and Geiger's book (2003) thoroughly describes the data modelling techniques used to construct multipurpose, stable and sustainable data warehouses used to support business intelligence.

Malinowski and Zimanyi's book (2008) explains the conventional data warehouse design in detail, particularly complex hierarchy modelling. Additionally, it addresses two innovative domains recently introduced to extend the capabilities of data warehouse systems, namely the management of spatial and temporal information. Its presentation covers different phases of the design process, such as requirements specification, conceptual, logical and physical design.

3.3.6 Domain-Specific Application Aspect

This category focuses on a variety of extensions to the ER models that are relevant to specific application domains such as:

- geographical information systems (Hadzilacos and Tryfona, 1997; Vert, Stock and Morris, 2002),
- hypermedia (Garzotto, Mainetti and Paolini, 1994; Bowers, Delcambre and Maier, 2003),
- superimposed information (Murthy, Delcambre and Maier, 2006),
- web applications (Feyer and Thalheim, 1999),
- electronic commerce (Karlalalem, Dani and Krishna, 2001),
- multimedia (Velez, 1985), and
- manufacturing (Flory and Giard, 1988; Moyne, Teorey and Leo C. McAfee, 1991).

Examples of the proposals for hypermedia, security and geographical applications are summarised below:

1. **The Hypertext Design Model 2 (HDM2).** HDM2 (Garzotto et al., 1994) is the evolution of the Hypertext Design Model (HDM) (Garzotto,

Paolini and Schwabe, 1993) and is an extension of the classical ER model. The concept of access structure (i.e. index and guided tour), the notion of anchors, and the definition of the dynamic behaviour of guided tours are extremely important in hypermedia applications but are not covered in the traditional ER approach. The main features of HDM2, with respect to HDM, are an improvement of the access mechanisms, a generalisation of the notion of link, an extension of the notion of derivation, definitions of *hyperviews* and a refinement of the definition of browsing semantics. Neither integrity constraints nor languages are defined in HDM2.

2. **The Security Enhanced Entity-Relationship (SEER) model.** The SEER model (Oh and Navathe, 1995) extends the conceptual level of the EER model to deliver a model that handles security schemata and authorisation history details. This model proposes a two-layered representation of data with the first layer based on the traditional ER model and the secondary layer used to deal with security schemata and authorisation histories. This framework aims to serve as a common data model that provides independence from other different access control mechanisms supported by the participating DBMSs. Neither integrity constraints nor languages are defined in SEER.
3. **The Geographic Entity-Relationship (Geo-ER) model.** This model is based on the study of spatial aspects that call for special modelling techniques and constructs to deliver conceptual designs for geographic applications. Geo-ER (Hadzilacos and Tryfona, 1997) is presented as a model that extends the ER model to integrate the concepts of aggregation and grouping. Geo-ER includes spatial entity types and spatial relationship types, geographic entities' position, and space-depending attributes to express the semantics of space. In addition, two new constructs, spatial aggregation and spatial grouping, are added to express the spatial dimension of complex geographic entity types. Neither integrity constraints nor languages are defined in Geo-ER.

This thesis survey has identified the following relevant resources (listed in alphabetical order by first author) that provide the necessary data modelling and design techniques for the domain-specific application aspect.

Books:

DeMers's book (2005) discusses spatial and mapping concepts, the components of GIS systems and the process of designing and implementing a GIS system.

Elangovan's book (2006) discusses the fundamentals of GIS, database creation and analysis in GIS and advanced GIS applications.

Longley, Goodchild, Maguire and Rhind's book (2005) gives a comprehensive treatment of the field of GIS ranging from the fundamental principles to the advanced features.

Shekhar and Chawla's book (2002) discusses issues and approaches providing a wide range of applications and methods for spatial data management that are at the core of GIS.

3.3.7 Knowledge Base Aspect

Since the 1990s, a series of research achievements that are relevant to knowledge bases have been reported from such fields as statistics, database management and machine learning (Shimazu, Momma and Furukawa, 2003). For example, in the field of database management, deductive databases have been proposed to combine logic programming with relational databases to construct systems that support applications with very large datasets. As the ER model provides an effective tool for organising the design of relational databases, it is a natural extension that it should also be used to assist in the design of deductive databases (Han and Li, 1992). In the field of machine learning, a logic approach called Inductive Logic Programming (ILP) is a machine learning technique that has been effective for deducing rules based upon qualitative and structural data.

This aspect includes ER extensions that are concerned with knowledge based modelling or knowledge organisation within deductive databases (Battista and Lenzerini, 1993; Han and Li, 1992; Kerschberg, Baum and Hung, 1990; Storey, Goldstein, Chiang and Dey, 1994) and inductive learning based on predicate logic (Shimazu et al., 2003).

Examples of proposals for ER extensions that deal with knowledge, deductive database and inductive learning are discussed as follows:

1. **The Knowledge-based Entity-Relationship model (KORTEX).** KORTEX (Kerschberg et al., 1990) is a prototype expert database system that

supports knowledge-based extensions of the Extended ER (EER) model (Teorey, Yang and Fry, 1986) and incorporates knowledge-based concepts such as rules and virtual objects. The main KORTEX knowledge components are inferred attribute values and virtual objects. For inferred attributes, their value may be determined by a function calculation or through inference rules. In the case of virtual objects, they can represent entities or relationships and are also referred to as inferred views. For virtual entities, their actual values are inferred by the system from a related entity. From a users' perspective, these virtual objects appear as distinct objects in the system.

A variety of tools have been provided that assist in the maintenance of KORTEX such as in verifying the internal validity of inference rules or in supplying schema information to the user. Although KORTEX utilises a variety of constructs within the model, no supporting language has been provided with this prototype.

2. The Deductive Entity-Relationship (Deductive-ER) model. This proposal applies the ER approach to the design of deductive databases and delivers a deductive-ER model (Han and Li, 1992) that is an integration of two data models:

- a typical ER model and its refinements, and
- a typical Horn-Clause-Based logic data model.

A major motivation for the development of a deductive-ER model is the need to organise knowledge in deductive databases. As a deductive database is a deductive extension of a relational database, the model inherits many features from relational and ER models. The outcome is an integrated language called Deductive-ER, which takes advantage of both relational and logic data languages and facilitates the construction of a structured deductive database.

The Deductive-ER model provides the capability to define and manipulate real, virtual and hybrid components, generalisation hierarchies, integrity constraints and complex data objects (including tuple-valued, list-valued, text-valued, set-valued and null-valued attributes). A data definition and manipulation language is built into this model and is termed Deductive-ER. The proposal also caters for the specification and use of constraints.

3. The Refined Entity-Relationship (RER) model. RER (Shimazu et al., 2003) is an extended ER model that accommodates an additional feature for each attribute (of an entity) and each relationship, which is used to determine whether the value of these has been derived from another attribute or relationship. These features are compliant with the input data identification rules for Inductive Logic Programming (ILP) systems. As ILP is one of the most expressive machine-learning algorithms, the RER model facilitates adaptive data mining by directly connecting relational databases and ILP systems. Their research provides specifications on the interface based on the RER model that enables ILP systems to access relational databases. However, the proposal does not include a discussion of constraints or query languages that are specific to the RER model.

This thesis survey has identified the following relevant resources (listed in alphabetical order by first author) that provide the necessary data modelling and design techniques for the knowledge base aspect.

Books:

Ohsuga, Kangassalo, Jaakkola, Hori and Yonezaki's book (1992) collated conference papers that have a main theme of information modelling and knowledge bases. Their article list is classified according to the following topics: Theory of Concepts and Conceptual Modelling, Acquisition and Elicitation of Modelling Knowledge, Knowledge Representation I and II, Database Design, Knowledge Base Design and Software Engineering, and Different Approaches to Conceptual Modelling.

A revised version of this book (Kangassalo, Jaakkola, Ohsuga and Wangler, 1995) includes a survey of European-Japanese research on information modelling and knowledge bases.

3.3.8 Fuzzy Data Aspect

Fuzzy data (imprecise or uncertain data) can arise whenever subjective judgments or evaluations are part of the stored database (Yazici, George, Buckles and Petry, 1992). The concept of fuzzy data has been incorporated into database technologies in order to deal with ambiguous and uncertain data, provide support to queries based on natural linguistics, and to allow for the modelling of data that is inherently fuzzy. Extending data models to cater for imprecise and uncertain

information is of particular interest given the incomplete nature of information in the real world. Consider the example where you are given the information ‘*David may be 62 years old*’. Whilst this information retains some element of validity, the exact value is indeterminant.

Within the context of relational databases, research into the use of fuzzy data has predominantly occurred during the last two decades. The aim of this has been on allowing the storage of imprecise or fuzzy data and developing query constructs that can process and extract this information (Galindo, Urrutia and Piattini, 2006). More recent efforts have focused on fuzzy object-oriented databases. Research into extending conceptual models to deal with fuzzy data has included the Fuzzy IFO model (Ma, Ma and Zhang, 2000), which has focused on the modification to objects to deal with different levels of fuzziness, especially for *is-a* relationships.

Other research that extends conceptual models has included the Fuzzy Extended Entity-Relationship (FEER) (Ma, Zhang, Ma and Chen, 2001), Fuzzy ER (Zvieli and Chen, 1986) and Fuzzy Enhanced Entity-Relationship (FuzzyEER) (Galindo et al., 2006) models. The aim of these models is to deal with different types of uncertainty in order to improve the expressiveness and usefulness of the base ER/EER models. More recent proposals such as the AR-enriched-ER (AR-EER) model (Chen, Ren, Yan and Guo, 2007), have attempted to extend the ER model based on association rules (AR) to deal with more general, flexible and linguistic based knowledge in fuzzy association rules. The transformation from an AR-EER schema to a relational schema is also discussed in this proposal.

The major concepts of the Fuzzy ER and FuzzyEER models considered in this thesis include:

1. **The Fuzzy ER model.** Fuzzy ER (Zvieli and Chen, 1986) is an extension of the ER model to incorporate fuzzy set theory, where fuzzy entities, attributes and relationships are represented graphically in the model. Their proposal considers fuzziness at three levels:
 - (a) The first level refers to the fuzziness at the model level, relating to fuzzy entity types, fuzzy relationship types and fuzzy attributes.
 - (b) The second level concerns the fuzziness at the occurrence/instance level of entities or relationships.
 - (c) The third level is related to the fuzziness in attribute values.

Whilst fuzzy participation constraints and cardinality constraints have been discussed in their research, the model does not provide any query language or the method of mapping the Fuzzy ER data model to a relational database.

- 2. The FuzzyEER model.** FuzzyEER (Galindo et al., 2006) extends the EER model with fuzzy semantics and fuzzy notations to represent imprecision and uncertainty in entities, attributes and relationships. FuzzyEER presents various fuzzy features for fuzzy modelling, e.g. fuzzy values in the attributes, the degree of fuzziness of the value of an attribute, fuzzy entities, fuzzy relations, fuzzy aggregation, fuzzy constraints, and so on. The possibility of expressing constraints by using the power and flexibility of fuzzy set theory is a key feature that distinguishes their approach from various other fuzzy data treatments (Zvieli and Chen, 1986; Chen and Kerre, 1998; Ma et al., 2001; Ma, 2005, 2006). Their approach considers many of the essential modelling facilities including participation constraints, fuzzy cardinality constraints and the method of mapping the FuzzyEER model to the relational data model. A query language termed a Fuzzy SQL (FSQL) is also provided for use with the fuzzy database.

Note that the **fuzzy data aspect** also includes the MDER model (La-Ongsri et al., 2008) as some information of complex structures that are used to describe the domains of an attribute in MDER is inherently imprecise or fuzzy. For example, location attributes such as City or Zip code that are constructed as a weighted graph usually involve various forms of uncertainty such as *close to*, *adjacent to* or *in proximity*.

This thesis survey has identified the following relevant resources (listed in alphabetical order by first author) that cover the ways that the ER model has been extended to enhance its **fuzzy data aspect**.

Survey Papers:

A recent review paper by Ma and Yan (2008) introduces fuzzy database models based on fuzzy relational and object-oriented databases. The paper also presents an overview of imperfect information and fuzzy set theory. The paper precludes any examination of fuzzy conceptual data models.

Yazici et al. (1992) surveyed two conceptual modelling approaches, the ER model and the IFO model, that have been extended to incorporate fuzzy and imprecise data. Their research also proposes relational models that may exist

in either first normal form (1NF) or non-first normal form (Non-1NF) which are capable of representing imprecise information.

Books:

The recent book by Galindo (2008) contains a collection of chapters devoted to research on fuzzy information processing in databases. The book delves into two critical challenges of fuzzy databases, namely what processes can be used to extract fuzzy data and secondly, how can fuzzy data be stored in a database. The book also offers topics about fuzzy data mining such as the extraction of fuzzy association rules.

Galindo et al.'s book (2006) examines extensions of EER models to provide fuzzy capabilities. Their research forms the basis of a proposal, termed the Fuzzy-EER model. Some of the extensions of the proposal include fuzzy attributes, fuzzy aggregations, and other assorted specialisations such as fuzzy degrees and fuzzy constraints.

Ma's book (2005) covers the then current research and practical applications of fuzzy conceptual models (in particular, in ER/EER and UML model), fuzzy databases (mainly, object-oriented databases and relational databases) and the fuzzy XML model. Processes to transform fuzzy XML and fuzzy EER models into fuzzy databases are also presented.

Another book by Ma (2006) presents further advances for imprecise and uncertain engineering information from a fuzzy database modelling perspective.

3.3.9 XML Data Aspect

The evolution of XML from a simple data exchange format to the native data format of application components has led to its widespread use within many application areas and its gradual acceptance as a fundamental element of any system (Sengupta and Wilde, 2006). In recent years, application designers have adopted XML schema to describe their requirements as a logical data model. This cannot be achieved through the use of traditional conceptual models since they do not have the modelling capacity to accommodate XML features.

Over the last decade, a number of extended conceptual models for XML data have been proposed in the literature. These extensions are based on ORM (Bird, Goodchild and Halpin, 2000), UML (Combi and Oliboni, 2006; Lu, Yang and Liu, 2006; Routledge, Bird and Goodchild, 2002), and ER models such as XSEM-ER

(Nečaský, 2007), EReX (Mani, 2004), XER (Sengupta, Mohan and Doshi, 2003), X-Entity (Lósio, Salgado and Galvão, 2003) and ERX (Psaila, 2000). These latter XML-based conceptual models that are based on the ER model are of particular relevance to this thesis survey.

Examples of these XML based ER models are described below:

- 1. The EReX (ER extended for XML) model.** EReX (Mani, 2004) extends the ER model with additional XML features that accommodate a specification of structures (i.e. categories) and constraints (i.e. coverage and order) which can be modelled using the ER model. The proposal provides a mechanism to translate an EReX schema into XML, but difficulties arise in dealing with the notion of document order. Document ordering is part of the overall ordering of elements covered by XML, but in the case of EReX, there is no corresponding notion of what these global documents represent in real world applications. While the model uses standard XML, it does not provide any additional operators or inferencing rules for querying and manipulating XML data.
- 2. The XSEM-ER model.** The main features of XML data are hierarchical and irregular structures, ordering and mixed contents. XSEM extends the ER model to represent these features through a two phase approach (Nečaský, 2007). Firstly, XSEM-ER is used to produce a model representing the overall non-hierarchical conceptual schema of a domain and from this first step non-hierarchical XSEM-ER constructions are generated. In the second phase, operators transform these constructions into XSEM-H constructions to provide a hierarchical organisation of all the structures. Whilst the model uses standard XML and provides for integrity constraints within the model, it omits to provide any additional operators or inferencing rules for querying and manipulating XML data.

Another recent proposal for conceptual representation of XML content structures is the ER-XML model (Al-Kamha, Embley and Liddle, 2007) that extends the XER model proposed by Sengupta et al. (2003). Their study discusses the requirements necessary for the conceptual modelling of XML which address ordering, irregular and heterogeneous structure, document-centric data, constraints and logical level mapping.

This thesis survey has identified the following relevant resources (listed in alphabetical order by first author) that cover the ways that the ER model has been

extended to enhance its XML data aspect.

Survey Papers:

Nečaský's survey (Nečaský, 2006) describes five existing conceptual models for XML, two of which are based on the ER model with the remainder are based on the hierarchical approach. The survey proposes a list of requirements that are considered as an essential feature of any XML based conceptual model, and compares these various surveyed models against these requirements.

Sengupta and Mohan (2003) surveyed eight data models for XML, only one of which is related to extensions of conceptual modelling for the ER model. The study classifies the models according to three characteristics, namely physical, formal and conceptual.

Book:

Chaudhri, Rashid and Zicari's book (2003) provides useful reference material on general XML data management. One section of this book provides an XML-based data model that is of particular relevance to this aspect.

3.4 A Comparative Study of ER Modelling Extensions

A conceptual model is a type of a data model. From a formal viewpoint, a data model possesses at least three components, namely a structure component, an integrity component and a manipulative component (Codd, 1980; Date and Darwen, 1992). These components can be applied to any application created for an organisation (Date and Darwen, 1992).

The CERME framework has identified four key criteria that can be used as a common framework for comparing all the surveyed models. These include data structures, integrity constraints, languages and transformations. The first three have been previously identified in the key considerations of a conceptual data model (Chapter 2). A detailed description of these criteria is as follows:

Data Structures. A data structure of a data model is a representation of the modelling constructs used to describe a database schema. Fundamental modelling constructs of conceptual data models are represented by entity types, relationship types and attributes.

Integrity Constraints. Integrity constraints are general statements and rules that define the set of consistent database states, or rules for determining how a state may change, or both (Codd, 1980).

Languages. In the context of standard data models, languages define the algebra of operations, calculus, DDL and DML. In general, these are commonly referred to as query languages. A query language is a collection of operators or inferencing rules that can be applied to any valid instances of the data structure types of the model, with the objective of manipulating and querying data in those structures to achieve the required result (Codd, 1980).

Transformations. As the ER model is not supported by any DBMSs, another mechanism is required to translate the ER schema into a logical model representation (such as relational, object-relational or XML schemata) that can support implementation. This mechanism is referred to as a *transformation*. Transformation processes are crucial in creating logical schemata that serve as the basis for any database implementation. For this reason it is included as a necessary characteristic of the comparison framework.

The need for customising ER-based applications has led to a great deal of research on developing extensions to the traditional ER model, which has aimed at providing increased expressiveness and incorporating a richer set of semantics into databases. This thesis survey shows that during the 1980s and 1990s, research into extending the ER model was predominantly focused on **temporal aspects**. In the current era, research has shifted more towards the use of **XML data aspects** and developing modelling techniques to support this.

This thesis survey also illustrates how few of these proposals for ER extensions are able to fulfil all the key considerations or criteria (data structures, integrity constraints and languages) of conceptual data models. The main proposals that have been considered in the CERME framework are shown in Table 3.2, along with a tabular comparison of whether each can support the essential features of data models. As discussed by Codd (1980), any extension that omits to define integrity constraints or languages in their data models should be regarded as being incomplete. Table 3.2 shows these deficiencies in the surveyed models. Thus, the focus of any research into new extensions of conceptual data models must clearly address these two criteria as they are essential in providing the key to understanding how structures behave.

Table 3.2: A comparison of the main CERME proposals (“√” indicates support).

No.	Proposal	Year	Data Structure	Integrity Constraint	Language	Transformation
Structural Aspect						
1.	E²R	1989	√	√		√
2.	HERM	1990/2000	√	√	√	√
3.	MDER	2008	√	√	√	√
Data Abstraction Aspect						
4.	ERC+	1992	√	√	√	√
5.	EER	1994/2007	√	√		√
Temporal Aspect						
6.	TIMEER	1998/2004	√	√		√
7.	TIMEERplus	2005	√	√		√
Spatio-Temporal Aspect						
8.	STER	1999	√			
9.	MADS	1999/2006	√	√	√	√
10.	DISTIL	2001	√			√
Data Warehousing Aspect						
11.	ME/R	1998	√			
12.	starER	1999	√			
13.	MultiDimER	2006	√	√		√
Domain-Specific Application Aspect						
14.	HDM2	1994	√			
15.	SEER	1995	√			
16.	Geo-ER	1997	√			
Knowledge Base Aspect						
17.	KORTEX	1990	√	√		
18.	Deductive-ER	1992	√	√	√	
19.	RER	2003	√			
Fuzzy Data Aspect						
20.	Fuzzy ER	1986	√	√		
21.	FuzzyEER	2006	√	√	√	√
3.	MDER	2008	√	√	√	√
XML Data Aspect						
22.	EReX	2004	√	√		√
23.	XSEM-ER	2007	√	√		√

It is also expected that the key criteria (data structures, integrity constraints and languages) that have been identified as representing the core properties of conceptual data model can also be used as a guideline in identifying the new modelling constructs or components that must be considered whenever a new data model is proposed. These criteria have also been used to form the common basis for the comparative examination of all the CERME proposals.

This study includes previous survey papers, bibliographies and books that are relevant to each of the CERME aspects. Through this survey, it is apparent that research is predominantly focused on expanding the expressiveness and applicability of models, rather than in surveying existing proposals or undertaking further research into the fundamental theory of modelling. For instance, there

Table 3.3: Resources available for each CERME aspect.

No.	CERME Aspect	Resources		
		Survey Paper	Bibliography Paper	Book
1.	Structural			Thalheim (2000)
2.	Data Abstraction	Saiedian (1997)		Elmasri and Navathe (2007)
3.	Temporal	1) Gregersen and Jensen (1999) 2) Ozsoyoglu and Snodgrass (1995) 3) Roddick and Patrick (1992) 4) Roddick and Spiliopoulou (2002) 5) Theodoulidis and Loucopoulos (1991)	1) Bolour et al. (1982) 2) Kline (1993) 3) McKenzie (1986) 4) Soo (1991) 5) Stam and Snodgrass (1988) 6) Tsostras and Kumar (1996) 7) Wu et al. (1998)	1) Date et al. (2003) 2) Snodgrass (2000)
4.	Spatio-Temporal	1) Abraham and Roddick (1999) 2) Friis-Christensen et al. (2001) 3) Pelekis et al. (2004) 4) Wang et al. (2000)	1) Al-Taha et al. (1993) 2) Al-Taha et al. (1994) 3) Roddick et al. (2001) 4) Roddick et al. (2004)	1) Ott and Swiaczny (2001) 2) Parent et al. (2006)
5.	Data Warehousing			1) Imhoff et al. (2003) 2) Malinowski and Zimanyi (2008)
6.	Domain-Specific Application			1) DeMers (2005) 2) Elangovan (2006) 3) Longley et al. (2005) 4) Shekhar and Chawla (2002)
7.	Knowledge Base	Kangassalo et al. (1995)		1) Ohsuga et al. (1992) 2) Kangassalo et al. (1995)
8.	Fuzzy Data	1) Ma and Yan (2008) 2) Yazici et al. (1992)		1) Galindo (2008) 2) Galindo et al. (2006) 3) Ma (2005) 4) Ma (2006)
9.	XML Data	1) Necasky (2006) 2) Sengupta and Mohan (2003)		Ghaudhri et al. (2003)

has been much research on fuzzy databases and incorporating XML into conceptual modelling, but there have been limited survey reviews of these areas. This bias has provided a good reference point for identifying opportunities for further potential areas of research. Table 3.3 provides a summary of the resources that are available to support each of the CERME aspects.

3.5 Summary

Discussion on the enhanced power for modelling has accompanied the ER approach for three decades and this will continue into the future. As argued by Badia (2004) and Kroenke and Gray (2006), no conceptual model will ever be able to fully capture all the semantics of an application. Thus, the limitations of each ER model must be weighed up against their benefits, such as their ease of use, intuitive appeal and clarity of semantics. Establishing a balance between the expressiveness of a model and its complexity of use will always be a challenge for database developers and researchers.

The motivation behind various extensions to the ER model is to enhance data model expressiveness. In this context, the term expressiveness is used to represent any required meaning. This is achieved through proposed constructs that can be assigned given meanings (semantics).

This study proposes that various semantic features can be supported through a classification system referred to as the CERME framework. This framework classifies ER extensions according to the following nine CERME aspects:

- structural,
- data abstractions,
- temporal,
- spatio-temporal,
- data warehousing,
- domain specific applications,
- knowledge base,
- fuzzy data, and
- XML data.

It is clear that there are substantial weaknesses with the traditional ER model that require the enhancement of various new extensions to the model. Research must be directed to overcoming these limitations and to deliver a data model that is sufficiently adaptable to deal with emerging user needs and the complexity of

applications. As stated by Chen (2006), this vision must not be constrained by restrictive constructs, and must have sufficient versatility to deal with all aspects of the real world, its changes and activities under different perspectives.

Periodic surveys of new model proposals should be undertaken in an attempt to maintain this surveyed model list and to document any new CERME aspects and proposals that must be considered. For instance, any future research into expanding the ER model to handle the semantics of *ontologies* should be added into this list. Greater awareness of the potential for new semantics, content and approaches can lead to the development of more advanced, high-quality conceptual level information systems (Spaccapietra, March and Kambayashi, 2002).

This thesis survey contributes to the general understanding of the various potential uses and applicability of the various extensions of the ER model that have been promoted over the last three decades. This work reflects upon some of the common limitations of the extensions that have been proposed and provides guidelines on core elements that must be considered for all new conceptual models. This guide and survey will serve as a valuable reference for future research.

The CERME framework has been a useful mechanism to categorise those ER model extensions that have been designed to deal with specific aspects associated with modelling while also incorporating an assessment of their completeness according to the criteria of data structures, integrity constraints, languages and transformation.

This framework is a valuable tool that can assist developers in raising awareness of what needs to be considered when developing new proposals. Specifically, the comparison chart in Table 3.2 identifies which conceptual data model proposals can be considered as complete within each of the CERME aspects. Using this guide, researchers have a valuable point of reference for evaluating the completeness of their proposed data models and for identifying prior effective examples of modelling extensions that may inspire new proposals in their specific area of research interest.

Chapter 4

Mesodata in Conceptual Modelling

This thesis chapter serves to provide some of the answers to support Objectives 1 and 2 of the thesis as stated in Chapter 1 (particularly Section 1.5.1) and as indicated in Figure 1.3. The focus of this chapter is in exploring the conceptual modelling extensions to accommodate the concept of mesodata and how this concept can be used to increase the expressiveness of conceptual modelling approaches. Mesodata modelling is a recently developed approach for enhancing a data model's capabilities by providing for more advanced semantics to be associated with the domain of an attribute. Mesodata supplies both an inter-value structure to the domain and a set of operations applicable to that structure that may be used to facilitate additional functionality in a database. Through each of the following chapter sections, the thesis demonstrates that conceptual models are able to retain more meaningful information pertaining to their model of their real world application if they were able to incorporate the semantics of complex data types into the attribute domains. This chapter investigates the accommodation of mesodata into the entity-relationship and object role modelling techniques, presenting the Mesodata Entity-Relationship (MDER) model and Mesodata Object Role Modelling (MDORM), which show how the mesodata concept can be incorporated into conceptual modelling methodologies to include the semantics of complex domain structures.

The structure of this chapter is as follows. Section 4.1 provides an overview to this topic. Section 4.2 contains a discussion on the concept of mesodata approach. Section 4.3 shows how mesodata can be accommodated into the ER model. Section 4.4 discusses the accommodation of mesodata into the ORM technique. Sec-

tion 4.5 presents examples of MDER and MDORM schemata. Section 4.6 provides some general insights and outlines the directions of future work.

4.1 Introduction

The process of conceptual modelling plays a major role in assisting the description of some part of the real world, the Universe of Discourse (UoD), as a concise description of the user's database requirements. The conceptual model ignores physical level aspects such as the physical storage structure, indexes, clustering and access paths, as well as external level aspects such as user views, screen forms and report interfaces. The result is a conceptual schema that is independent of the DBMS but which can be implemented by mapping onto a logical schema supported by the chosen data model (relational, object-oriented, object-relational and so on) (Halpin and Proper, 1995).

While the entities, their attributes and the relationships between entities are explicitly denoted, in a typical conceptual model of an enterprise any understanding of the domains of the attributes is only implicitly included, at best. For example, attributes such as *DayofWeek*, *DiseaseCode* or *Colour* would rarely have their semantics recorded in the conceptual model. Mesodata makes a strong distinction between the type of an attribute and its domain (the open or closed set of values that an instance of an attribute is permitted to take). It should be noted that while complex *types* in the relational and object-oriented data models have been widely explored, complex *domain structures* have not been examined to the same depth.

Several extensions have been presented to enrich the semantic expressiveness of the original entity-relationship (ER) model (Chen, 1976) by providing, for example, the valence concept (Baumann, 1989), representation for security semantics (Pernul, Winiwarter and Tjoa, 1993), dynamic phenomena (events) (Falkenberg, 1993), fuzzy data types (Galindo et al., 2006), multidimensional data (Sapia et al., 1998) and abstraction mechanisms, such as specialisation, aggregation, association and categories (Batini et al., 1992; Elmasri et al., 1985; Elmasri and Navathe, 2007). The regular ER model is focused on the modelling of static structures and as a result, there are several temporal extensions (Klopprogge, 1981; Tauzovich, 1991; Theodoulidis et al., 1991a; McBrien et al., 1992; Zimanyi et al., 1997) which add dynamic aspects to the ER model. Refer to Chapter 3 for a discussion on other extensions to the ER model.

There have been similar extensions to Object Role Modelling (ORM). As with the ER model, ORM is a powerful semantic conceptual model. ORM has gained the attention of the conceptual modelling community and is now supported by a variety of modelling tools such as VisioModeler, Microsoft's Visio Enterprise 2000, DogmaModeler and NORMA. Research has explored its use in conceptual modelling, including ORM-based approaches for modelling contextual information (Henricksen, Indulska and McFadden, 2005), e-tutorial systems (Leelawatananon and Chittayasothorn, 2004), web systems (De Troyer, Casteleyn and Plessers, 2005) and in-house decision support systems (Pierson and Cruz, 2005). ORM has also been used for business rules as a markup language (Demey et al., 2002) as well as for an ontology engineering framework and tool (Jarrar et al., 2003). Reactive behaviour (Halpin and Wagner, 2003) and temporal extensions have been proposed (Pornphol and Chittayasothorn, 2004; Proper, Hoppenbrouwers and van der Weide, 2005). Extensions to ORM's precursor, NIAM (Nijssen, 1977, 1985), have also been proposed (Puntheeranurak and Chittayasothorn, 2002; Creasy, 1989; Chankuang and Chittayasothorn, 2004; Yuliana and Chittayasothorn, 2005), including a transformation from NIAM into Optimal Normal Form (ONF)¹ (Leung and Nijssen, 1987, 1988), a set of rules for schema conversion from NIAM to EER and vice versa (Song and Forbes, 1991), a semantic comparison of the ER model and NIAM (Laender and Flynn, 1993) and the EER model and NIAM (Kim and March, 1995), as well as an analytical evaluation of NIAM's grammar for conceptual schema diagrams (Weber and Zhang, 1996).

However, with the exception of work such as FuzzyEER (Galindo et al., 2006), most extensions to the ER/EER and NIAM/ORM models do not deal with the semantics of complex attribute domains. In the classical models, attribute domains are, in practice, restricted to simple scalar types such as the set of integers, reals, character strings and so on. This thesis argues that in order to reflect real-world domains, a conceptual modelling tool should be powerful enough to represent the semantics of complex attribute domains such as graphs, trees and lists. As a simple example, consider the extensive use of *reference* or *lookup* files — files that simply enumerate the values that an attribute may take. These reference files may either refer to information that can be designed as non-inclusion or inclusion, as shown in the example in Figure 4.1. In practice, as they require an additional relationship type and entity type to be included, these are often

¹ONF is 5NF with a minimal number of relations.

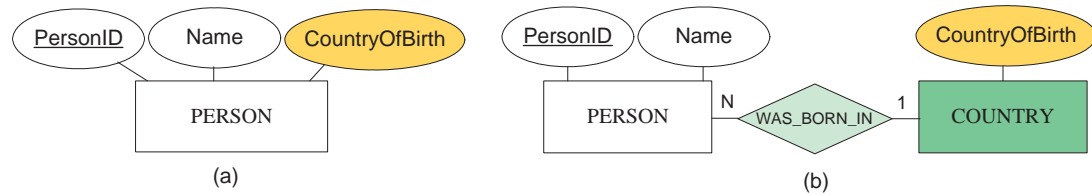


Figure 4.1: Examples of reference files (a) non-inclusion, and (b) inclusion.

left off conceptual diagrams with their existence noted elsewhere². More complex domains, such as hierarchies, cannot be easily included at all.

Mesodata adds power to the relational data model by providing greater semantics to the domain of an attribute by allowing attributes to be defined over complex domain structures (de Vries and Roddick, 2004). Importantly for backwards compatibility, the *type* of an attribute remains a simple scalar type; it is the *domain* of the attribute that allows the values taken by the attribute to be placed within some complex structure.

Consider the example of an attribute CountryofBirth that is defined as CHAR(30), where country names exist within the domain which is a weighted graph. This domain of CountryofBirth includes all country names (current and superceded), which are then accessible to the DBMS with the extended SQL operators. Assuming these exist in table `countryrel`, mesodata can be used to create a relational table as follows:

```

CREATE DOMAIN  COUNTRIES
              AS          wgraph
              OF          CHAR(30)
              OVER       countryrel;

CREATE TABLE person (
  PersonID    CHAR(6) NOT NULL,
  Name        CHAR(40),
  :
  CountryOfBirth COUNTRIES,
  :
  PRIMARY KEY PersonID,
  :

```

After defining CountryOfBirth over the mesodata domain COUNTRIES, the query

```

SELECT      PersonID
FROM        person
WHERE      CountryOfBirth EQUALTO 'USA';

```

would return those people where their country of birth is equivalent to any value in the domain that is synonymous with 'USA', namely 'U.S.', 'US', 'USA', 'America',

²This issue also leads to semantic ambiguity as discussed by Wand et al. (1999).

‘United States’, ‘United States of America’, etc. In this case, `EQUALTO` is an operation available to domains defined as `wgraph`. Without mesodata, the user would have to enumerate each of these values and may still remain unaware of changes or other variations to the country name.

The Object-Oriented Data Model (OODM) and the Object-Relational Data Model (ORDM) have both emerged in response to the increasing complexity of database applications. Moreover, semantic extensions such as *Data Blades* in Informix and *Data Cartridges* in Oracle enhance an RDBMS by providing a specific data type extension for some application domains, such as spatial, time series, text or image databases (Elmasri and Navathe, 2007; Stonebraker, Brown and Moore, 1999). A number of data types and various operators on the data types have been provided, for example, spatial data types include `point`, `line`, `polygon`, `path`, `circle` and so on and spatial operators including `overlap`, `contains`, `above`, `below`, `nearest` and so on. Such extensible data types have been implemented as add-on packages that can be included as abstract data type (ADT) libraries whenever users need to implement the types of application they support (Elmasri and Navathe, 2007). None of these provides the semantic extension offered by the mesodata concept. Furthermore, although the support for additional or extensible data types in ORDBMS seems similar to mesodata modelling, there is a distinct difference in that the former provides specific data type extensions for complex *attributes* while mesodata provides the ability to describe and use complex *domains*.

To date, the incorporation of mesodata into conceptual modelling has not been adequately investigated. Since the structure of the domain is an important design decision, it is believed that the conceptual schema itself should be modified to support domain semantics with respect to complex data types. To that end, this thesis shows how mesodata can be accommodated into two well-known but very different modelling techniques to demonstrate how the mesodata concept can be extended to enhance the semantics of complex domain structures in data modelling. The Mesodata Entity-Relationship (or `MDER`) model is presented here as an extension to the ER/EER model and the Mesodata Object Role Modelling (or `MDORM`) as an extension to the ORM model.

For `MDER`, the concepts of mesodata entity type, mesodata mapping and total mesodata domain participation constraint are introduced. A transformation algorithm from an `MDER` schema into a corresponding relational database schema is provided in Chapter 8. These proposed extensions are similar (and orthogonal) to those of the Enhanced Entity-Relationship (EER) model that extend the ER

model's representational capabilities. The EER model allows the use of additional concepts such as subclass, superclass and category (Elmasri and Navathe, 2007) while the MDER model allows the use of mesodata types, mesodata domains and the mesodata layer to represent the richer semantics of complex data types for attribute domains.

The second example, MDORM, extends ORM. ORM is a popular, fact-oriented model with several strengths with respect to richer business rule capture, greater stability given changes to any application domain and easier verbalisation and population (Halpin, 2001). MDORM introduces new constructs that are able to represent the mesodata concept within a conceptual schema. The proposed constructs are integrated with the existing features of the ORM approach without any conflict with current ORM diagrammatic notations. The concepts of mesodata value type and mesodata mandatory role constraint are defined. An algorithm to transform an MDORM conceptual schema into a relational database schema is provided in Chapter 8.

Some complex structures can also be expressed as complex objects formulated using the Unified Modelling Language (UML) (Booch et al., 2005; Muller, 1999). However, this thesis suggests that the distinction between objects and domain values is important since not every complex domain structure also meets the intuitive concept of an object. From a semantic point of view, the data component is the kernel of the object component because data types are commonly used as attribute domains as well as for query results (Engels et al., 1992). This thesis thus argues that complex data types should directly be modelled in the conceptual schema to reflect the structures and complexities of some real world domain to meet the challenges of new application development trends. As with the investigation of mesodata in ORM, this thesis suggests that mesodata can also be modelled in UML class diagrams to represent the structures of complex domains.

4.2 Mesodata Approach

Mesodata is a recent modelling approach to facilitate the definition of attributes over complex domain structures such as graphs, trees and lists thus providing for more advanced semantics to the domain of an attribute and enhancing a data model's capabilities (de Vries et al., 2004).

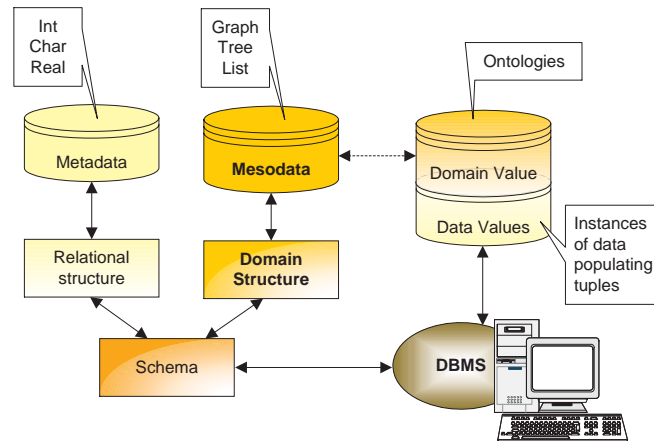


Figure 4.2: Mesodata layer between metadata and data (based on de Vries, 2006).

4.2.1 What is Mesodata?

Mesodata is a concept that provides an intermediate conceptual layer of domain definition between the metadata and data as shown in Figure 4.2 to accommodate the definition of complex domain structures within the database (de Vries, 2006; de Vries and Roddick, 2004). This new layer is introduced to the relational database which is traditionally a two-layered framework consisting of the metadata and the data (de Vries and Roddick, 2007). This middle layer, called mesodata, separates the schema definition of an attribute from the semantics of its domain (de Vries, 2006; de Vries and Roddick, 2004, 2007). Commonly used data structures, such as lists, graphs and trees (called *mesodata types*), and operations for the manipulation of these data structures are defined in the mesodata layer.

Earlier research by de Vries (2006) and de Vries and Roddick (2007) pointed out the important difference between a graph as a mesodata type and a graph as a user-defined abstract data type (ADT) as follow:

- In the former, an attribute in a relation would take as its value from an instance of an elementary type that exists within a graph.
- In the latter, the value of the attribute exists in the relation.

In other words, a mesodata type represents the structure of the *domain* whereas an ADT represents the structure of the *data value*. In addition, unlike the provision of libraries of user-defined ADTs, a mesodata type graph is not

Table 4.1: Examples of mesodata types and their operators (adapted from de Vries and Roddick (2007)).

Mesodata Type	Operator (Extended SQL Operator)
GRAPH	NEXTTO
WGRAPH	NEXTTO, CLOSETO, FAR, EQUALTO
DGRAPH	NEXTTO
DWGRAPH	NEXTTO, CLOSETO, FAR
TREE	INTREE, DESCENDENT, PARENT, CHILD, SIBLING, ANCESTOR
WTREE	CLOSETO, FAR, INTREE, DESCENDENT, PARENT, CHILD, SIBLING, ANCESTOR
LIST	NEXT, PREVIOUS, FIRST, LAST, INBETWEEN
CLIST	NEXT, PREVIOUS, INBETWEEN
SET	INSET
Tri-State Logical	MAYBE

directly accessible or manipulable through the attribute (de Vries, 2006). Rice, Roddick and de Vries (2006) further extend the mesodata concept to accommodate attribute domains defined over multiple types. Their work identifies the important distinction between mesodata techniques and object-oriented concepts, in that the former introduces the idea of storing complex *domain values* in the databases while the latter are concerned with complex *attribute values*. Earlier discussion regarding mesodata concepts can be found in de Vries (2006), de Vries and Roddick (2004), de Vries et al. (2004), de Vries and Roddick (2007) and Rice et al. (2006).

4.2.2 Key Components of Mesodata

As discussed by de Vries (2006), de Vries and Roddick (2004) and de Vries et al. (2004), mesodata includes the following key components to facilitate the implementation of semantically-rich domains.

Mesodata layers. A mesodata layer contains domain structures such as graphs and trees, including their values and inter-relationships and is able to accommodate domain variability by mapping between different representations.

Mesodata Types and Operators. Mesodata types are *domain structures* that are typically in the form of complex data types with built-in operators for

Table 4.2: Examples of mesodata types and the structure of source relations. (adapted from de Vries and Roddick (2007)).

Mesodata Type	Source Schema
GRAPH	R (<u>FROM</u> , <u>TO</u>)
WGRAPH	R (<u>FROM</u> , <u>TO</u> , WEIGHT)
DGRAPH	R (<u>FROM</u> , <u>TO</u>)
DWGRAPH	R (<u>FROM</u> , <u>TO</u> , WEIGHT)
TREE	R (<u>PARENT</u> , <u>CHILD</u>)
WTREE	R (<u>PARENT</u> , <u>CHILD</u> , WEIGHT)
LIST	R (<u>SEQUENCE</u> , <u>ITEM</u>)
CLIST	R (<u>SEQUENCE</u> , <u>ITEM</u>)
SET	R (<u>ITEM</u>)
Tri-State Logical	—

manipulation and comparison that can be used to develop mesodata domains. Table 4.1 presents examples of mesodata types with operators that can be executed over domain structures. These operators extend the currently available SQL operators and enhance querying power. Having defined a domain of an attribute over a mesodata type, these additional operators become available according to the mesodata type selected. Mesodata types are populated by values that are stored in source relations.

Source Relations. A source relation is used to hold the specific structural information for the domain. The structure of the source relation relies on the mesodata type selected as shown in Table 4.2. For example, the weighted graph (*WGRAPH*) mesodata type presents a structure of the source relation as R (FROM, TO, WEIGHT), where R is a name of a source relational schema and has a composite key consisting of both attributes (denoted by underlining both *FROM* and *TO* attributes) as a primary key for the schema.

Mesodata domains. A mesodata domain is built by matching a mesodata type with a base type and a source relational schema to hold the specific structural information for the domain. An example of the base type, such as *INTEGER* or *CHAR(12)*, could form a simple domain or it could in turn become a mesodata domain based on a mesodata type, for example, a weighted graph of integers or strings.

Table 4.3: A relation `zipcoderel` (an example of a source relation for the mesodata type weighted graph).

ZIPCODEREL		
Zipcode1	Zipcode2	Weight
5042	5042	0.0
5042	5050	0.1
5042	5043	0.1
5042	5047	0.1
5042	5051	0.2
5050	5051	0.1
5050	5043	0.15
5050	5047	0.15
.	.	.
.	.	.
.	.	.

4.2.3 Example Use of Mesodata

As discussed by de Vries and Roddick (2007), modelling data in traditional relational databases requires specifying attributes over a restricted set of data types. In general, these modelling techniques capture the format of the data value but do not capture information about the attributes domain or how a specific value may be related to other values within that domain.

Consider the example of an attribute within a database that is used to record a zip code. Basic data types of this attribute have standard collating sequences that may be defined as `CHAR(4)`. However, interpreting zip code to capture advance semantics such as *closeness* requires a domain knowledge of the unit of measure and its relationships to other values (de Vries and Roddick, 2007). For example, the zip code Bedford Park 5042 is adjacent to zip codes 5043, 5047 and 5050 (refer to Figure 4.6(b), Page 110).

The mesodata solution for the attribute `Zipcode` is to use the mesodata type `WGRAPH` for defining the mesodata domains of zip code. The operators such as `NEXTTO`, `CLOSETO` and `EQUALTO` are then available to operate over this domain structure. To define such domains based on mesodata type `WGRAPH`, it also requires the source relational schema that describes the specific structural information of the weighted graph. For example, assume this source exists in a relation `zipcoderel` (Table 4.3), which has a composite key (`Zipcode1` and `Zipcode2`) as a primary key for this relation. Thus, a mesodata type weighted graph and its source relational schema can be used to create the mesodata domain for attribute `Zipcode`, that can then be used to relate `Zipcode` to the mesodata domain as shown below:


```

CREATE DOMAIN  LOCATIONS
AS            wgraph
OF            CHAR(4)
OVER         zipcodere1;

CREATE TABLE  person
(
Emp_ID       CHAR(4) NOT NULL,
Name        CHAR(30),
:
:
Zipcode     LOCATIONS,
:
:
PRIMARY KEY ... );

```

To provide answers to questions such as *Find all persons who live in suburbs adjacent to zip code 5042*, the following type of query can be used:

```

SELECT      Name
FROM        person
WHERE       Zipcode NEXTTO '5042';

```

This would return those persons recorded with any of zip codes adjacent to '5042', which in this case are '5043', '5047' and '5050'. In this case, `NEXTTO` is an operator available to domains defined as `WGRAPH`. Note that the values of `Zipcode` that result from this query exist within a weighted graph of `LOCATIONS` domain, not in a relation `person`.

4.3 The Mesodata Entity-Relationship (MDER) Model

Conceptual data modelling is typically carried out using graphical tools allowing users to model enterprise data and their relationships in an intuitive way. The Entity-Relationship Model (ER) (Chen, 1976) is a high-level conceptual data model consisting of *inter alia* entity types, relationship types and attributes. These basic components, visually represented by the Entity-Relationship diagram, are used to view (and to reach some form of agreement on) the real world as a construct of entity types and associations between entity types.

This thesis presents an extension to the ER model, termed the **MesoData Entity-Relationship** (or **MDER**) model, which includes the concepts of mesodata types, mesodata domains and the mesodata layer. Such concepts are modelled as *mesodata entity types*. At the conceptual level, the focus is on the structure of the data used in an organisation, the relationships between values and the constraints imposed on their use. MDER also includes the idea of *mesodata mappings* and introduces a *total mesodata domain participation* (TMDP) constraint. All of the






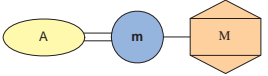
Symbol	Meaning
	Entity Type
	Relationship Type
	Attribute
	Mesodata Entity Type with two sections of a name of mesodata domain and a mesodata type
	Mesodata Mapping
	Total Mesodata Domain Participation Constraint of A in m

Figure 4.3: Symbols of major components in the MDER model.

original ER concepts and symbols remain available together with the new concept of mesodata entity type. Additionally, EER extensions are orthogonal to the MDER extension and can also be included. Figure 4.3 shows symbols for the major components in the MDER model comprising the basic ER constructs, the new mesodata entity types, the mesodata mappings and the TMDP constraint.

Formally, there are three main criteria for considering the representation of data models — data structures, integrity constraints and languages. The following sections describe how these criteria apply to the MDER model.

4.3.1 MDER Data Structures

The MDER model includes all the modelling constructs and concepts of the ER/EER models. In addition, it extends the attributes in the ER/EER model with the support of mesodata domains.

The basic mesodata construct is a complex domain structure termed the mesodata type, while the enumerated values, that can be taken from a simple data type or another mesodata type, are termed the mesodata domain. These constructs are modelled as *mesodata entity types*. In addition, mesodata includes mapping to convert a value from source relations to a domain value. This is modelled as *mesodata mappings*.



Figure 4.4: A **COUNTRIES** mesodata entity type with **WGRAPH** mesodata type (as per the example discussed in Section 4.1).

4.3.1.1 Mesodata Entity Types

A mesodata entity type is a complex-domain entity type that represents a *mesodata domain* for an attribute. Each mesodata entity type has a name and a mesodata type. For example, a **COUNTRIES** mesodata entity type may have the name **COUNTRIES** and a weighted graph (**WGRAPH**) mesodata type. This **WGRAPH** mesodata type has built-in operators such as **NEXTTO** and **CLOSETO** for manipulating values provided in the graph structure. Conceptually, all mesodata is stored in a *mesodata layer* (although in practice, they may be split between other system files and user relations as appropriate).

Mesodata entity types in **MDER** are depicted as hexagons: the top section holds the name of mesodata domain, the bottom section the mesodata type as shown in Figure 4.4.

Mesodata Type. The mesodata type defines the domain structure and is typically a complex data structure (such as a graph, tree or list). It is also likely to have associated operations (refer to **EQUALTO** in the example in Section 4.1 and **NEXTTO** in the example in Section 4.2). In general, the range of mesodata types is fixed by the database implementation although a variety of specialisations may be available (e.g. for graphs, users can choose from weighted or unweighted, directed or undirected, etc.). It is also possible to define a complex mesodata type in terms of another mesodata type (a directed graph of circular lists for example).

The weighted graph mesodata type is constructed with built-in operators (such as **CLOSETO** and **EQUALTO**) and stores relationships and paths between nodes providing the metrics for proximity and adjacency. The weighted arcs between two data values (nodes), with values from 0 to 1, represent the degree of similarity. An example of weighted graph is shown in Figure 4.5.

While an attribute's type is normally a simple scalar type, the domain from which the values are taken can have a complex internal structure (i.e.

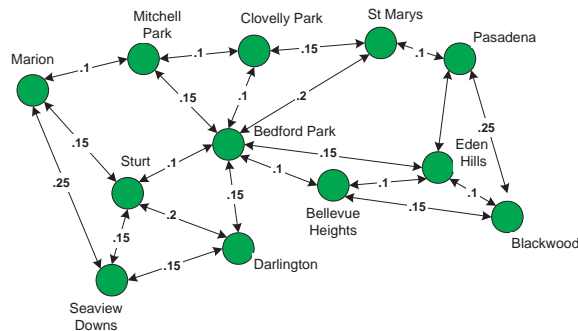


Figure 4.5: An example of a Suburb connectivity.

mesodata type) that is not only a collection of values but also characterises the way in which these values are related. That is, a mesodata type consists of a set of *values* which can be structured, for example, as a graph, a tree, a list, etc., together with a set of *operations (functions)*. The values are provided in a *source relation* (Figure 4.6(a)).

Mesodata Domain. A domain is a set of values representing some property. In most cases, the values that constitute a domain are implied through its data type. Similarly, the values that constitute a mesodata domain are specified by a mesodata type.

The mesodata domain is a set of atomic values taken from either a simple domain or another mesodata domain. For example, a weighted graph of characters refers to a weighted graph mesodata type being constructed from a base type in which its component nodes are characters. An attribute defined over such a domain would take an instance of an elementary value that exists within the mesodata structure.

In the example in Figure 4.6, the base type for the Suburb attribute would remain unchanged (e.g. `CHAR(25)`) as would the data values. The Suburb would take its value as an instance of a base type that exists *within a weighted graph* mesodata type, where the intrinsic operations such as `CLOSETO` are defined to operate over such specific domain structures.

Figure 4.7 shows an example in which the three of the five attributes of person (namely Dept, Suburb and Salary) are defined over mesodata domains `NEWDEPTS` and `OLDDEPTS`, `LOCATIONS` and `SALARIES` respectively (the second with total mesodata domain participation). An attribute with more than one associated mesodata domain can take its value from any of the linked domains (which

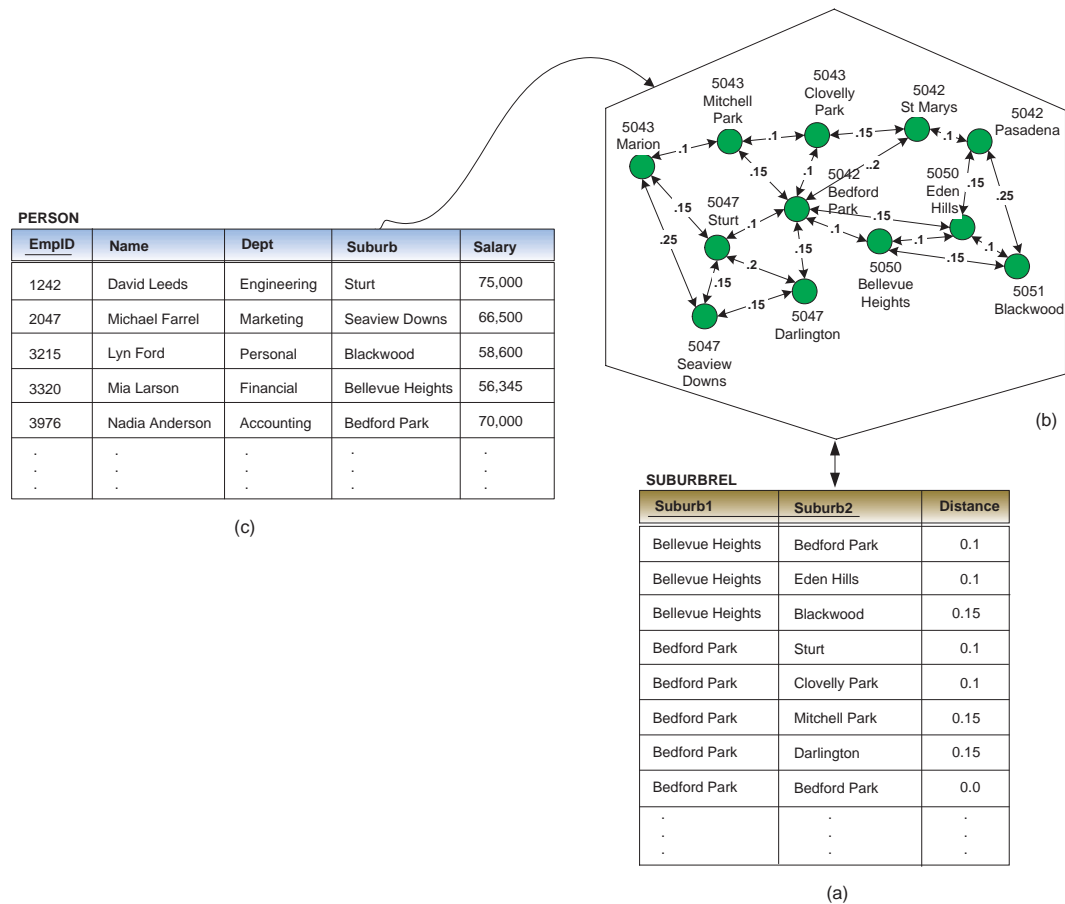


Figure 4.6: An example of Suburb attribute referencing a weighted graph mesodata type (a) a source relation for the weighted graph mesodata type, (b) a weighted graph mesodata type and (c) a person relation .

can be of use in dealing with changes to domain structure (cf. de Vries and Roddick (2004, 2007)).

The definition of mesodata domain can be specified by three associated components: (1) the mesodata type, (2) the simple (base) data type of an associated attribute, and (3) the source schema describing a specific structure of the mesodata type. These components can be explicitly declared to create a mesodata domain and to allow attribute specifications to refer to the mesodata domains (refer to a transformation from the MDER schema to a relational schema in Chapter 8).

4.3.1.2 Mesodata Mappings

A mesodata mapping between a mesodata entity type and an attribute defines a set of mappings required to interpret a data value within a specific and well

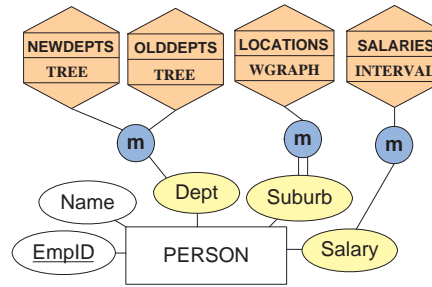


Figure 4.7: An MDER Example.

recognised concept. This includes mapping to convert a value from source relations to a domain value. Mesodata mapping, which is typically created and used internally by the system, is specified to convert a value to its meaning. Built-in mesodata mappings also allow regularly used mappings to be accommodated in the DBMS to provide additional functionality.

Consistent with other EER extensions, the MDER diagrammatic notation for mesodata mapping is represented as a circled **m** as demonstrated in Figures 4.3 and 4.7. A mesodata mapping connecting an attribute to more than one mesodata domain is shown by the mesodata mapping connected to attribute **Dept** in Figure 4.7.

4.3.2 MDER Integrity Constraints

A database schema is a formal and abstract definition of the semantics and constraints of the corresponding real-world system (Lukovic, Ristic and Mogin, 2003). The set of integrity constraints that arise from business rules are crucial to the structure of a database schema in order to maintain data consistency.

Constraints are specified on database schema to reflect restrictions on data values in the real-world. Constraints can be specified in various ways including their specification in the schema of the data model via the DDL so that the DBMS can automatically enforce them while others need to be checked within application programs.

The MDER model includes all the constraints of the ER/EER models. In addition, it includes one new constraint — the total mesodata domain participation or TMDP, requiring an attribute to take its value from one of the values recorded in the mesodata domain. If these constraints are not imposed then an attribute may take any value consistent with the attribute’s base domain. An attribute that references a mesodata type can be defined to participate either *totally* or

partially. The TMDP constraint specifies that every attribute value must correspond to values that exist within the mesodata domain. If TMDP is not specified the value must still adhere to the base type but may hold values not enumerated in the domain.

In MDER diagrams, TMDP is shown as a double line connecting the participating attribute to the mesodata mapping as shown in Figures 4.3 and 4.7.

4.3.3 MDER Languages

Each relational DBMS provides a Data Definition Language (DDL) for defining the conceptual and internal schemata, including their mappings, and a Data Manipulation Language (DML) for manipulating and querying the data. As the implementation of the mesodata was based on a relational database, the mesodata language specified by de Vries (2006) and de Vries et al. (2004) was given as an extension to SQL to define and manipulate the mesodata type.

The mesodata DDL (MDDL) extends SQL's DDL commands (such as `CREATE`, `ALTER` and `DROP`) to implement mesodata types. For instance, the `CREATE DOMAIN` command was extended to allow for the specification of the complex domain structure (the mesodata type) in the mesodata layer, and the `CREATE TABLE` command was extended to allow attribute specifications to refer to the mesodata domains. Similarly, the mesodata DML (MDML) extends SQL's DML commands to query records that are defined over the mesodata domains. For example, the `SELECT` statement was extended to allow for the new *mesodata operators* which are associated with the mesodata types.

4.4 The Mesodata Object Role Modelling (MD-ORM) Model

Like the ER model, ORM cannot handle the semantics of complex attribute domains. To capture such semantics, ORM must be extended to capture mesodata domains and types. This thesis introduces an extension to ORM, termed the Mesodata Object Role Modelling (or MDORM) which allows the mesodata concepts to be modelled with ORM's value types. It is assumed that the reader already has a basic familiarity with ORM; a detailed discussion of database design using ORM approach can be found elsewhere (Nijssen and Halpin, 1989; Halpin, 2001).


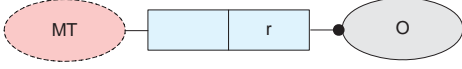
Symbol	Meaning
	Mesodata value type MT (where MT denotes mesodata type e.g. TREE, LIST, WTREE, GRAPH or WGRAPH)
	Mesodata mandatory role constraint (mesodata role r is mandatory for the population of O, specified as a black dot)

Figure 4.8: ORM Symbols Extension.

Compared with the top-down structured design methodology of the ER model, ORM can be considered as a bottom-up approach since it builds up from atomic objects in terms of elementary relationships. The ORM design method, sometimes known as a binary semantic model (Kent, 1984) and often referred to as NIAM (Nijssen and Halpin, 1989; Verheijen and van Bekkum, 1982), is a fact-oriented approach for modelling information at a conceptual level. It was developed in the early 1970s based on the binary approach (Senko, 1975, 1976) and is well established as a conceptual modelling tool for relational databases. With its emphasis on fact types, ORM is also called *fact-oriented modelling* (Nijssen and Halpin, 1989).

As discussed by Halpin (1998) and Halpin and Morgan (2008), ORM differs from both ER and UML, in that it makes no explicit use of attributes as a base construct but instead uses the relationship type to express all fact types. For example, instead of using `CountryOfBirth` as an attribute of `person`, ORM uses the relationship type `Person was born in Country`. This attribute-free approach leads to stable semantics in conceptual model and conceptual queries (attributes may evolve into entities or relationships). For example, if it is decided to record the population of a country, then `CountryOfBirth` needs to be reformulated as a relationship. To do this in ORM, the country type is simply declared to be independent. The object type `country` may be populated by a reference object that contains those country codes of interest.

As an extension of ORM, MDORM supports all the current ORM concepts and symbols together with the new concepts of mesodata value type and mesodata mandatory role constraint. Figure 4.8 shows the graphical notations for the additional constructs.

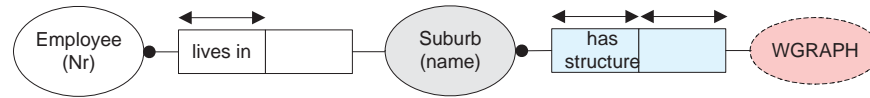


Figure 4.9: Example of a Suburb where the suburb names have the WGRAPH structure and are restricted by the mesodata mandatory role constraint.

4.4.1 MDORM Data Structures

MDORM includes all the modelling constructs and concepts of ORM. In addition, it extends value types in ORM with mesodata types. The idea behind the MDORM approach is to provide a seamless extension to the ORM model. Starting with the basic ORM model, a mesodata type is added to the value type, termed a *mesodata value type*. A mesodata value type is a value type whose values are associated with a mesodata type. This is indicated by placing a MT (mesodata type) in a dotted ellipse (refer Figure 4.8). The MT can be labelled as, for example, TREE, GRAPH or WGRAPH that represents a mesodata type.

Each mesodata value type is annotated as a name of mesodata type in a dotted ellipse. Figure 4.9 expresses a relationship between a suburb entity type and the WGRAPH value type. When verbalising information from this example as elementary facts, the elementary sentences are stated as follows:

- Employee with employeeNr ‘3320’ *lives in* the Suburb.
- Suburb with suburbname ‘Bellevue Heights’ *has structure* as WGRAPH.

4.4.2 MDORM Integrity Constraints

ORM has a powerful and extensive graphical notation for representing constraints. Several researchers have addressed the strengths of this model in terms of a rich set of integrity constraints (Halpin, 2001, 2004; Halpin and Bloesch, 1999; Laender and Flynn, 1993; Song and Forbes, 1991). In practice there are usually several other kinds of constraints that need to be considered since business rules representing a certain aspect of a business domain or policy may change regularly.

MDORM includes all the constraints of ORM. Additionally, it proposes a new constraint applicable to mesodata domains. Similar to the TMDP constraint as discussed in Section 4.3, mesodata modelling requires a constraint, termed the Mesodata Mandatory Role (MMR) constraint, to indicate that the values of the object type must be taken from one of the values recorded in the mesodata

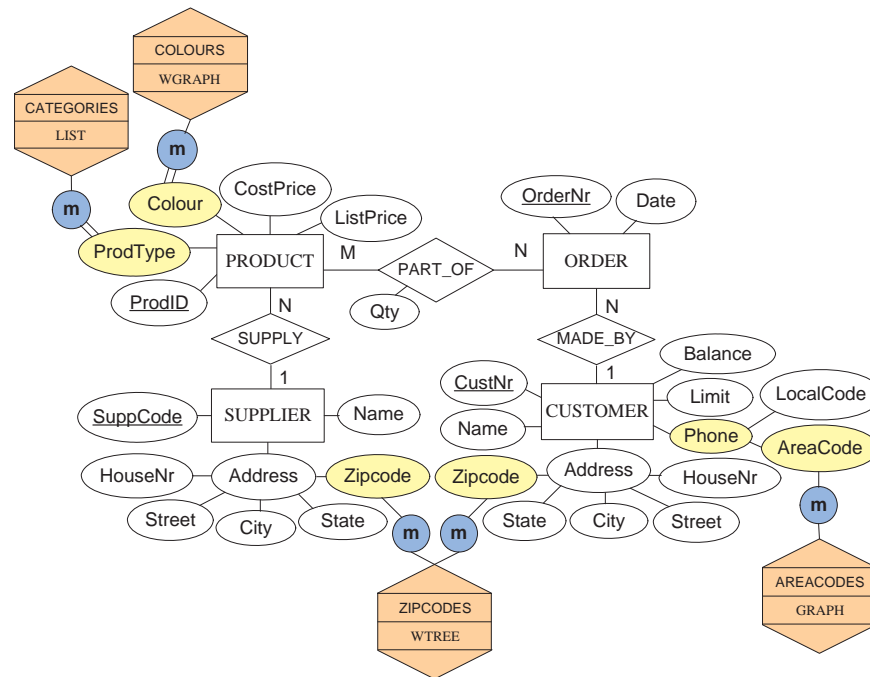


Figure 4.10: MDER diagram for an example INVENTORY database.

domain. This is specified as a black dot on the entity type that connects to the roles of mesodata reference types. The mesodata reference types are relationships between entity types and the mesodata value types. This constraint, as shown in Figure 4.9, declares that every instance in the population of the role's suburb involving the WGRAPH must play that role.

4.4.3 MDORM Languages

The first implementation of mesodata (de Vries, 2006) was based on the relational model (Codd, 1970, 1979), which is the most widely implemented data model in current commercial databases. Due to this popularity, this thesis continues this approach and uses SQL in providing support for languages in the MDORM model.

The use of the language support for MDORM is demonstrated in the transformation of an MDORM schema into a relational schema as discussed in Chapter 8. In particular, examples of the use of Mesodata DDL (MDDL) is demonstrated in the process associated with specifying schema definitions in that chapter.

4.5 Example MDER and MDORM Schemata

The motivation behind MDER and MDORM is to provide easy-to-use constructs with which to capture the semantics of an attribute's domain while keeping the graphical representation simple. Thus, the number of additional elements required is kept to a minimum and should be easy to learn and use. Note that the inclusion of mesodata may also have the effect of simplifying a conceptual model by replacing relationships with reference entities. For example, a **person** entity type linked with a relationship type of **was_born_in** to a **country** entity type (as described earlier in Figure 4.1) could be more appropriately described using an enumerated list for the **countries** mesodata domain. In practice, ER and ORM diagrams which include reference/lookup file entities are not uncommon.

As an example database application, consider the **INVENTORY** schema shown in Figures 4.10 and 4.11, based on the ER model and ORM approach respectively, which keeps track of products, orders, suppliers and customers. Typically discussions between a user and the analyst/modeller often contain examples of the queries that an organisation wishes to be able to execute over their data. Consider the following queries, none of which is particularly unusual:

List all orders for a chair or lounge (of any description) which is some shade of green,

List all products held which are similar, but not identical, in colour to the standard stock colours,

List all customers who live close to the Norwood store,

List all customers who live close to their suppliers, and

List all customers whose area code does not correspond to their zip code.

Each of these queries would require knowledge of the domain. Thus, the attributes **ProdType**, **Colour**, **Zipcode** and **AreaCode** in the **MDER** schema and the mesodata value types in the **MDORM** schema are identified as requiring a complex domain so as to assist with providing such answers.

4.6 Summary

This chapter shows that capturing semantics of complex domain structures has become increasingly important in dealing with emerging information demands

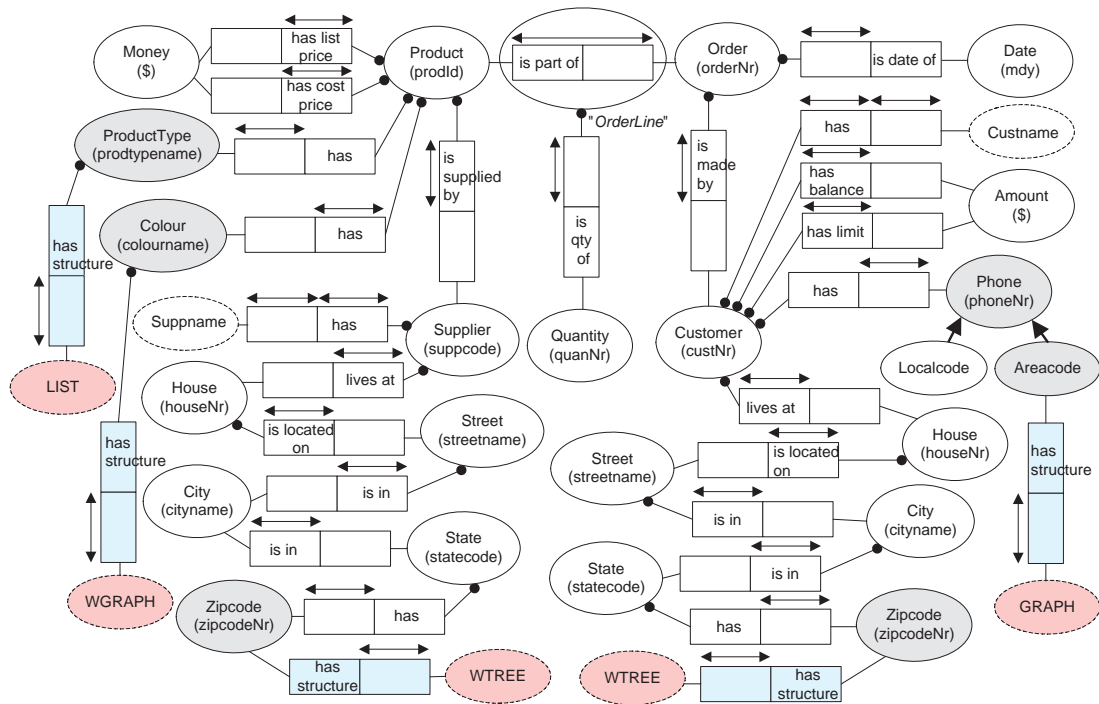


Figure 4.11: MDORM schema for an example INVENTORY database.

and that these semantics should be explicitly represented in conceptual schemata. Mesodata can be a very powerful mechanism to map such semantics directly into SQL.

A mechanism for accommodating mesodata into two conceptual modelling methodologies has been presented — the Mesodata Entity-Relationship (MDER) model and Mesodata Object Role Modelling (MDORM). The proposed constructs are simple and easy to use without introducing any conflict with current methods and mesodata modelling in ER and ORM and can be easily learned by database designers. Incorporating mesodata domains can also enhance the power of a DBMS’s query language by adding advanced semantic concepts to comparison operators used to retrieve information from databases.

While the focus of this research was on the Entity-Relationship model and Object Role Modelling, this thesis explores the wider modelling issues relating to mesodata as a conceptual tool. As well as investigating the accommodation of the mesodata in other modelling approaches (such as UML) this thesis suggests that the complexity of (and some of the restrictions relating to) current ontologies can be reduced. In some worked examples of larger ER models this thesis has found that either the ER model can be simplified using mesodata (by removing purely referential entities) and/or the semantics of the model can be enhanced through

the inclusion of additional constructs that can be assigned given meanings into the model.

In conjunction with this research that incorporates mesodata into two different modelling approaches, this thesis has found that the ORM approach (attribute-free approach) can be naturally extended and annotated with mesodata concepts and constraints more so than attribute-based modelling approaches such as ER and UML. As this research aim focuses on the broad issues relating to conceptual modelling, other challenges still remain in other modelling areas such as with the methodologies used for data warehousing.

The value of the suggested models that have been introduced in this chapter is ultimately measured in the practicability of their schemata that can be manipulated by a relational database system (refer to Chapter 8). This chapter has served to address Objective 2 of the thesis, namely to extend conceptual data models to enhance expressiveness. The practicability of these models is covered under Objective 4 that describes how these models can be mapped to the particular implementation platform to show the validity of the suggested conceptual models, which is discussed in Chapter 8.

Chapter 5

Ontology in Conceptual Modelling

This thesis chapter serves to provide some of the answers to support Objectives 1 and 2 of the thesis as stated in Chapter 1 (particularly Section 1.5.1) and as indicated in Figure 1.3. The focus of this chapter is in exploring the conceptual modelling extensions to incorporate the concept of ontologies and how this concept can be used to increase the expressiveness of conceptual modelling approaches. With the increasing complexity of applications and user needs, recent research has shifted from a data information level to a human semantic level interaction. Research has begun to address the increasing use and development of ontologies in various applications, strongly motivated by the semantic web initiative. However, existing conceptual models are not rich enough to incorporate ontologies in one single conceptual schema. To improve this situation, it is necessary to refine modelling formalisms and make them more expressive and semantically sound. It is argued that conceptual modelling methodologies would be semantically richer if they were able to express the semantics of a domain that arises in concrete application scenarios. This chapter investigates the incorporation of ontologies into three conceptual modelling methodologies, presenting the Ontological Entity-Relationship (OntoER) model, Ontological Object Role Modelling (OntoORM) and the Ontological Unified Modelling Language (OntoUML) Class Diagram. An extended conceptual framework for modelling ontologies is also discussed.

The structure of this chapter is as follows. Section 5.1 provides an overview to this topic. Section 5.2 introduces an insight into ontologies. Section 5.3 contains further discussion on the concept of ontological class hierarchy modelling. Sec-

tions 5.4, 5.5 and 5.6 show how ontologies can be accommodated into these conceptual modelling approaches — the ER, ORM and UML class diagram, respectively. Section 5.7 presents examples of *OntoER/OntoORM/OntoUML* schemata, followed by a summary in Section 5.8.

5.1 Introduction

Ontologies have been applied in a multitude of areas in computer science. The first significant growth of interest in the subject appeared in the mid 1990s which was motivated by the need to create principled representations of domain knowledge for the knowledge sharing and reuse community in the field of artificial intelligence (AI) (Guizzardi, 2006). A major challenge with ontologies is how to access, store and manage them. This research direction has included:

- supporting ontology-based semantic matching, querying and referential constraints in RDBMS (Das, Chong, Eadon and Srinivasan, 2004; Chong, Das, Eadon and Srinivasan, 2006; Necip and Freytag, 2003) leading to recent advances in ontology management in DBMSs such as those introduced by Oracle,
- implementing ontology systems or tools that support ontology-based applications (Corcho, Fernández-López and Gómez-Pérez, 2003; Cui and O'Brien, 2000; Dameron, Noy, Knublauch and Musen, 2004; Valo, Hyvönen and Komulainen, 2005), such as *Protégé*, which is an ontology and knowledge-base editor that allows the user to construct a domain ontology, customise data entry forms and enter data (*Protégé*, 2008).

Another challenge driving the database community is to create better data models. These research projects have attempted to use conceptual data modelling in supporting ontologies (Jarrar et al., 2003; Spaccapietra et al., 2004). For example, this has included research into methodologies for supporting database design creation and evaluation that makes use of domain-specific knowledge about an application stored in the form of domain ontologies (Sugumaran and Storey, 2006, 2002). In a further example, a theory of ontology was promoted that can be used to clarify the meaning of relationship constructs that are widely used to undertake conceptual modelling (Wand et al., 1999).

However, the conceptual modelling for ontologies and its transformation into relational database schema, has not been adequately investigated. This thesis

supports the argument that introducing ontologies into conceptual modelling can enhance the semantics of the model. This thesis suggests extensions to high-level conceptual models to represent the relationship between ontologies and the underlying conceptual schema.

With the increasing complexity of actual application scenarios and user needs, there is a requirement to shift from the data and information level to the human semantic level interaction. Consequently, semantic representation becomes important and to maximise the level of semantics requires that these representations become increasingly explicit. Humans learn to deal with the ambiguity of language by understanding the context in which terms are used. Data rich systems can emulate this by referencing data through structures such as *ontologies* that represent terms and their interrelationships (Das et al., 2004).

The main deficiency with traditional conceptual modelling practice is that ontologies are not semantically represented. This problem requires investigation to refine modelling formalisms to allow for the integration of ontologies into the conceptual schema. This thesis argues that conceptual modelling methodologies must be expanded to facilitate ontologies, including the reuse of existing ontologies, to enrich the semantic expressiveness of the data model. In addition, this approach caters for the ability to query the data in the context of its associated ontologies in the same way as querying simple relational data.

Consider a particular example of a medical database application that requires ontologies, specifically the knowledge associated with a hierarchical domain. For a simple application of a patient's visit to a doctor, a relational table with a Diagnosis attribute as shown in Figure 5.1 (a) can describe the type of diagnosis of a patient as identified by a physician at a visit date. The diagnosis attribute's domain is a hierarchical structure which can be represented as a diagnosis ontology shown in Figure 5.1(b).

Consider the following question: *how can those patients that have been diagnosed with immune deficiency conditions be identified?* The expected human response would be by identifying those patients with AIDS and so on. This response occurs as humans have the ability to combine data with the domain knowledge that AIDS is a type of immune deficiency condition, and in many instances this connection is made automatically at a subconscious level. However, the fact that AIDS is a type of immune deficiency condition is not explicitly represented in the data as shown in the Table in Figure 5.1(a), but belongs to the domain knowledge of diagnosis ontology as shown in Figure 5.1(b).

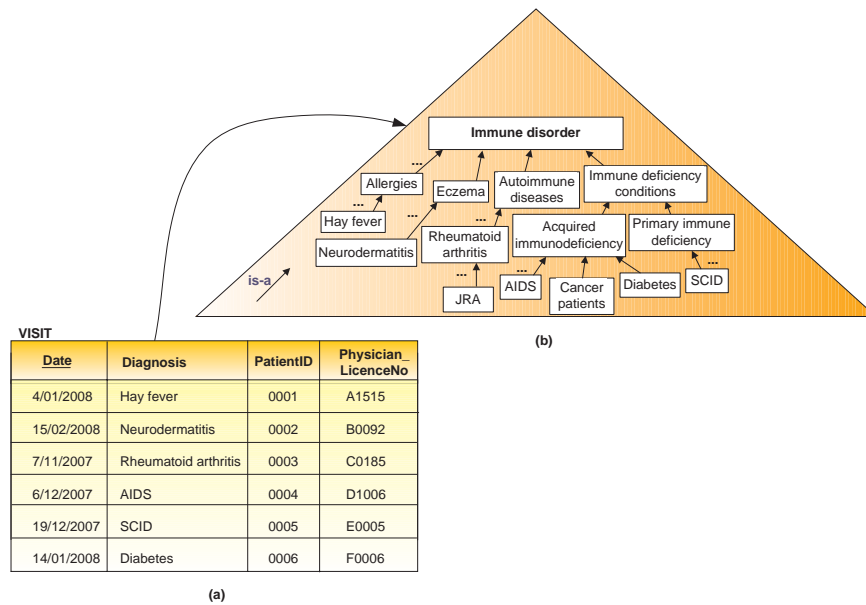


Figure 5.1: A portion of a medical database (a) the base table of patients visiting a doctor, and (b) a diagnosis ontology.

Within the medical database, the relational DBMS allows us to query on these attributes: Date, Diagnosis, PatientID and Physician_LicenceNo. This expressive power of traditional queries is relational complete. However, for querying such attributes whose domain, for example, is related in the ontology hierarchy (by relationships such as *is-a*), the traditional query is limited such as in the situation where it is necessary to identify those patients who are diagnosed with immune deficiency conditions. In this scenario, a query on a conventional database application could be constructed using the equality operator ($=$) as shown below:

```

SELECT      PatientID
FROM        visit
WHERE       Diagnosis = 'Immune deficiency conditions';
  
```

This traditional SQL query will fail to return any results that semantically satisfied the query condition since none of Diagnosis's values in the table will match 'Immune deficiency conditions'. In other words, the domain knowledge required to answer such queries is not present in the relational table in Figure 5.1(a). To provide semantically correct answers, the DBMS must not only know that Diagnosis denotes a disease but also the diseases' semantics, which is illustrated in the diagnosis ontology in Figure 5.1(b). This traditional query can be shifted from query-by-value to query-by-meaning to perform ontology-based semantics as follows:

```

SELECT      PatientID
FROM        visit
WHERE       Diagnosis INTREE 'Immune deficiency conditions';
  
```

The `INTREE` operator allows matching of the terms based on the specified relationship `is_a` in the diagnosis ontology. That is, the above query will return all patients who are diagnosed with 'Immune deficiency conditions', or any of its subclasses by consulting the referenced ontology.

This chapter explores how conceptual modelling can be improved by introducing ontologies into the model. To date, the incorporation of ontologies into conceptual modelling has not been adequately investigated. To show how ontologies can be accommodated into conceptual modelling methodologies, three well-known but different modelling techniques are used to demonstrate how the semantics of a specific domain knowledge can be incorporated into data modelling. The Ontological Entity-Relationship (`OntoER`) model is presented here as an extension to the ER/EER model, the Ontological Object Role Modelling (`OntoORM`) as an extension to the ORM and the Ontological Unified Modelling Language (`OntoUML`) class diagram as an extension to the UML class diagram.

This chapter further extends the use of mesodata concepts (Chapter 4) to introduce ontologies into conceptual modelling. Such ontologies include an existing ontology's class hierarchy of, for example, a relevant ontology of diseases (Wrong-Diagnosis.com, 2008) and the National Cancer Institute's (NCI) Cancer Ontology (Mindswap, 2008). It is anticipated that mesodata can provide *common domain structures* that can be used to accommodate the backbone ontology that is represented by the hierarchical organisation of concepts through the `is-a` relationship. This chapter thus defines the ontology's class hierarchy in the mesodata layer and use the `TREE` mesodata type to represent such ontologies. In addition, mesodata itself can be considered as an ontology (Subsections 5.2.2 and 5.2.3).

As this chapter uses common domain structures of the mesodata concept as a basis for the ontological modelling framework, it uses many of the concepts previously discussed in Chapter 4.

5.2 Review of Ontologies

As the concept of ontology is the key feature of this thesis research, it is necessary to define what an ontology is. This section provides a basic introduction to ontologies that includes different definitions of ontologies, type of ontologies and terminological clarifications of ontologies, and further comparisons between ontologies and conceptual data models.

5.2.1 What is Ontology?

Ontology is a term having different meanings in the disciplines of philosophy and computer science. The term Ontology (Greek. on = being, logos = to reason) in its original sense is a philosophical discipline (Sure, 2003) concerned with the nature of being and existence. In philosophy, ontology is the science of what is, of the kinds and structures of objects and properties in every area of reality (Smith, 2003). Ontology in this sense is often used by philosophers as a synonym of ‘metaphysics’ as defined by Aristotle. Ontology can be referred to as the study of conceptions of reality or the study of being or existence. It seeks to describe the basic categories and relationships of being or existence to define entities and types of entities within its framework.

In computer science, ontologies were developed in artificial intelligence to facilitate knowledge sharing and reuse (Fensel, 2001). More recently, ontologies are becoming widespread in fields such as knowledge engineering, knowledge representation, knowledge management, information integration, electronic commerce and the semantic web. Artificial intelligence and database management systems are fields where ontologies have been used as a means to add a formal and computable semantic dimension. The area of data and schema integration for heterogeneous databases is a recent example (Beneventano, Bergamaschi, Guerra and Vincini, 2003). Many definitions of ontologies have been given in the last decade; however, there is no agreed definition of what ontology is in computer science. A variety of definitions for ontologies are as follows:

— *An ontology is an explicit specification of a conceptualization* (Gruber, 1993).

Gruber’s definition has been widely cited in literature and accepted by most ontological engineers. Based on Gruber’s definition, many other variants have been proposed, adding emphasis to different features of the model.

— *An ontology is a formal, explicit specification of a shared conceptualization* (Borst, 1997).

The Gruber definition was modified slightly by Borst who added that the specification must be formal and the conceptualisation should be shared. This definition has also been clarified in Studer, Benjamins and Fensel (1998), Fensel (2001) and Gailly and Poels (2007) as follows: A ‘*conceptualization*’ refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. ‘*Explicit*’ means that the type of concepts used, and

the constraints on their use are explicitly defined. For example, in medical domains, the concepts are diseases and symptoms, the relationship between them are causal and a constraint is that a disease cannot cause itself. ‘*Formal*’ means that the ontology should be machine-readable. ‘*Shared*’ indicates that an ontology captures consensual knowledge, namely information that is not private to an individual, but accepted by a group.

— *An ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e., its ontological commitment to a particular conceptualisation of the world* (Guarino, 1998).

Guarino further refined the notion of conceptualisation by applying the theory of logic. This definition would be applicable to ontologies developed using logic.

— *An ontology defines a common vocabulary for researchers who need to share information in a domain. This includes machine-interpretable definitions of basic concepts in the domain and relationship among them.* (Noy and McGuinness, 2001).

This definition emphasises that ontology is the term used to refer to the shared understanding of some domain of interest.

— *An ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base* (Corcho et al., 2003).

This definition includes some highlights about the hierarchical representation of concepts and their relationships as well as knowledge bases. According to this definition, the same ontology can be used for building several knowledge bases, that would share the same skeleton or taxonomy. An ontology, together with all of the individual instances of its classes, constitutes a knowledge base; however, the boundary between the notion of ontology and the notion of knowledge base is somewhat blurred (Noy and McGuinness, 2001).

— *An ontology is a data model that represents a set of concepts within a domain and the relationships between those concepts.*

This definition covers fundamental concepts in the domain and relationships among those concepts.

— *An ontology is a formal specification of a perspective. If two people agree to use the same ontology when communicating, then there should be no ambiguity in the communication. To enable this, an ontology codifies the semantics used to represent and reason with a body of knowledge* (OwlSeek.com, 2008).

This is another recent definition provided by the website owlseek.com. It is understood to be an explicit formal description of a wider view that is used to refer to the shared understanding of a domain of interest. The important aspect of this definition is that it creates a shared understanding of a domain.

In addition, a set of constraints can be associated with the ontology to specify semantics of the concepts and/or relationships. Ontologies can be considered as conceptual schemata, intended to represent knowledge in the most formal and reusable way possible (Davies, Studer and Warren, 2006). Ontologies may also be thought of as semantically rich metadata capturing the information content of the underlying data repositories (Mena and Illarramendi, 2001).

In this section, the most relevant definitions of ontology have been collected. Based upon these various contributions, this thesis uses an informal understanding of ontology that defines it as a declarative model of a domain using abstractions called *concepts/terms* and the *relationship* among them. In other words, an ontology is a set of terms of interest in a particular information domain and the relationships among them.

More formally, an ontology can be represented as a pair $O:=(C,E)$ where C is a set of concepts and E is set of edges. $E \in (a, b, r)$ where $a, b \in C$, and r is the type of relationship between two concepts which defines the semantic (**is-a**, **syn-of**, **is-part-of**, etc.) and the algebraic properties (transitivity, symmetry, reflexivity, etc.) of the relationship. **Is-a** and **syn-of** are transitive (if A is-a B and B is-a C , it follows that A is-a C). **Is-a** is not symmetric (A is-a B excludes that B is-a A) while **syn-of** is symmetric (A syn-of B , then B syn-of A). **Is-a** and **syn-of** are reflexive (A is-a A , A syn-of A); both are semantically correct but ineffective in terms of data description. Thus, **is-a** and **syn-of** relationships can be expressed as a transitive property and **syn-of** can be expressed as a symmetric property as follows:

$$\begin{aligned} \textbf{Transitivity:} \quad & \forall x y z \text{ is-a}(x,y) \wedge \text{is-a}(y,z) \rightarrow \text{is-a}(x,z) \\ & \forall x y z \text{ syn-of}(x,y) \wedge \text{syn-of}(y,z) \rightarrow \text{syn-of}(x,z) \\ \textbf{Symmetry:} \quad & \forall x y \text{ syn-of}(x,y) \rightarrow \text{syn-of}(y,x) \end{aligned}$$

Semantics of other relationships which are dependent on the domain of the ontology always need to be specified (Necip and Freytag, 2005).

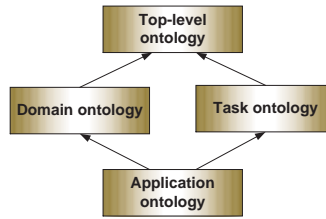


Figure 5.2: Different types of ontology. Arrows represent specialisation relationships (from Guarino, 1998).

5.2.2 Types of Ontology

The purpose of this section is to provide a general understanding of the vocabulary used to classify ontologies. There are several types of ontology which may be identified to fulfil different roles in the process to model the domain. Guarino (1998) proposed types of ontology based on their level of dependence on a particular task or point of view as shown in Figure 5.2.

Top-level ontologies. Top/upper-level ontologies (or generic ontologies) define very broad concepts or common objects that are generally applicable across a wide range of domain ontologies. They contain a core glossary in terms that can be used to describe a set of domains and provide general notions under which all root terms in existing ontologies should be linked. There are several standardised generic ontologies available for use, such as the Upper Cyc Ontology (Lenat, 1995), the Suggested Upper Merged Ontology (SUMO) (Niles and Pease, 2001), WordNet (Fellbaum, 1998) and a Description Ontology for Linguistic and Cognitive Engineering (DOLCE) (Gangemi, Guarino, Masolo, Oltramari and Schneider, 2002). These standards provide a structure and a set of general concepts from which domain ontologies (e.g., medical, financial, engineering, etc.) can be constructed. Top-level ontologies are also referred to as *foundation ontologies* and as *common sense ontologies* (Fensel, 2001).

Domain ontologies. Domain ontologies (or domain-specific ontologies) capture the knowledge valid for a specific/particular type of domain, or part of the world of interest. These represent the particular meanings of terms as they apply to that domain (e.g. electronic, digital, medical, etc.). For example, the term **mouse** has different meanings in different contexts. An ontology about the domain of **computer hardware** would model mouse as a

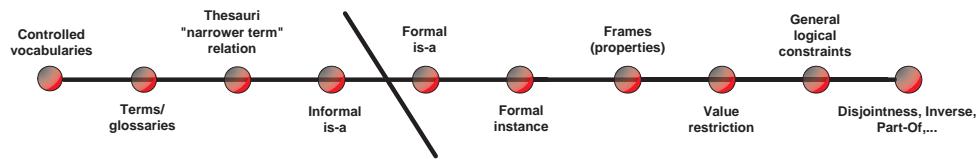


Figure 5.3: Ontological categories (from Lassila and McGuinness, 2001).

computer input device while an ontology about the domain of **creature** would model it as an **animal**. These ontologies provide vocabularies about the concepts within a domain and their relationships and the rules governing that domain. The concepts in domain ontologies are usually specialisations of concepts already defined in top-level ontologies.

Task ontologies. Task ontologies provide terms specific for particular tasks (e.g. ‘hypothesis’ belongs to the diagnosis task ontology), by specialising the terms in the top-level ontologies.

Application ontologies. Application ontologies describe concepts that depend upon both a particular domain and task, and often combine specialisations of both the corresponding domain and task ontologies. These concepts often correspond to roles played by domain entities while performing a certain task. Application ontologies are application-dependent containing all the definitions needed to model the knowledge required for a particular application.

In addition, Lassila and McGuinness (2001) classified ontologies based on the richness of their internal structure according to the following categories: controlled vocabularies, glossaries, thesauri, informal is-a hierarchies, formal is-a hierarchies, formal instances, frames, value restriction and general logical constraints as illustrated in Figure 5.3. The main categories and their meanings as described by McGuinness (2003), Corcho et al. (2003) and Lassila and McGuinness (2001) are summarised as follows:

- *Controlled vocabularies.* Controlled vocabularies (a finite list of terms) are one of the simplest notions of a possible ontology. A typical example of this category is a catalogue.
- *Glossaries.* Glossaries are lists of terms with their meanings. The meanings are specified as natural language statements. This provides a form of semantics or meaning since humans can read and interpret the natural language statements.

- *Thesauri*. Thesauri provide some additional semantics in their relationships between terms. They give information such as synonym relationships, but do not supply an explicit hierarchy. For example, *traveller*, *passenger* and *customer* could be considered as synonyms in the travel domain.
- *Informal is-a hierarchies*. Informal is-a hierarchies refer to specifications of term hierarchies such as those used by Yahoo. Such a hierarchy is not a strict subclass or **is-a** hierarchy. For example, the terms *car rental* and *hotel* are not kinds of travel but they could be modelled in *informal is-a hierarchies* below the concept *travel*, because they are key components of the travel and allow the user to select either a car rental for the trip or an accommodation.
- *Formal is-a hierarchies*. Formal is-a hierarchies include strict subclass hierarchies. In these systems, if A is a superclass of B and an object is an instance of B then it necessarily follows that the object is an instance of A. Strict subclass hierarchies are necessary to exploit inheritance. In the travel domain, subclasses of the concept *trip* could be *flight*, *train* and *cruise*.
- *Formal is-a hierarchies that include instances of the domain*. Formal is-a hierarchies that include instances of the domain are broad classification schemes with some schemata only including class names while others include individual content. For example, the flight description ‘QF720’ is an instance of *flight*.
- *Frames*. Frames are class hierarchies that include properties associated with each object class. Properties become more useful when they are specified at a general class level and then inherited consistently by subclasses and instances. Properties can be inherited by classes of the lower levels of the formal **is-a** taxonomy. For example, the *wine* class may associate with properties (*hasSugar*, *hasColor*, *hasMaker*, *madeFromGrape*) and may inherit one property (*locateIn*) from its superclass.
- *Ontologies that express value restriction*. Ontologies that express value restriction are a more expressive point in the ontology categorisation which includes value restrictions, or restrictions placed on the values that can fill a property. Each property is associated with a range class with values of the property being restricted to instances of the range class. For example, the *hasMaker* property takes values that are instances of the winery class.

- *Ontologies that express general logical constraints.* Ontologies that express general logical constraints are ontologies where the schema properties can expand dynamically according to the amount of gathered information. This method often takes the form of a mathematical formula where the value of the property is based on possible values of a different variable. For example, a logical constraint in a travel domain is that it is not possible to travel from Australia to New Zealand by train.
- *Ontologies that express logical constraints with more detailed relationships.* Ontologies that express logical constraints with more detailed relationships are very expressive ontology languages such as that seen in Ontolingua (Farquhar, Fikes and Rice, 1997) or Cyc's knowledge representation language (CycL) (Genesereth, 1991) that allow ontologists to specify first order logic constraints between terms and more detailed relationships such as disjoint classes, disjoint coverings, inverse relationships, part-whole relationships, etc.

Note that some researchers consider the previous categories (of catalogues, glossaries and thesauri) to be ontologies but most prefer to have an explicit hierarchy included before it is considered to be an ontology (Lassila and McGuinness, 2001).

In this chapter, mesodata can be considered as an ontology as it possesses an unambiguous interpretation of structures (e.g. a TREE mesodata type). Similarly, an existing class hierarchy can be considered as an ontology as it holds strict hierarchical subclass relationships between terms (e.g. an *is-a* relationship).

5.2.3 Terminological Clarifications of Ontologies

The range of ontology terminology as discussed by Smith and Welty (2001) is illustrated in Figure 5.4, which depicts a wide range of different types of specifications that have been termed ontologies in the literature. These specifications range from a simple catalog containing, for example, the products that a company sells, to a lexicon of terms with natural language definitions (thesaurus), to formal logical theories (Guizzardi, 2005).

The different types of specifications as described by Smith and Welty (2001) are summarised below:

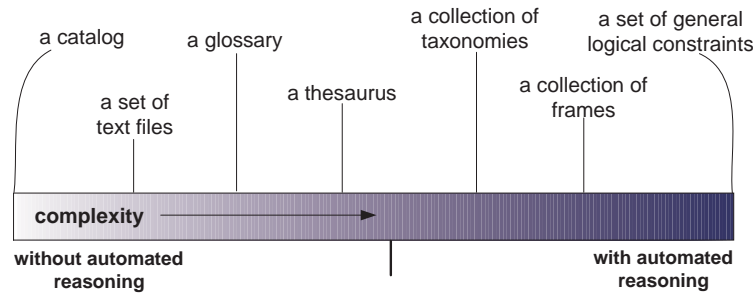


Figure 5.4: Different types of specifications classified as ontologies in the computer science literature (from Smith & Welty, 2001).

- *Catalogs.* Catalogs are the most simplest ontologies that provide simple natural language texts and allow string matching.
- *Text files.* Text files are sequences of readable characters such as letters, digits, punctuation or white space. This simplicity allows a wide variety of programs to display their contents.
- *Glossaries.* Glossaries provide natural language descriptions of terms, with an imposed structure in the text (indexing by terms).
- *Thesauri.* Thesauri provide, in addition to descriptions of terms, relationships to both general and more specific terms within a common hierarchy.
- *Taxonomies.* Taxonomies are structures in which properties of generic classes can be applied to those properties of more specific class definitions. The fields of knowledge representation, database and software engineering mostly use ontologies conceived as taxonomies.
- *Frames.* Frames extend the concept of taxonomies to include the relationships between objects and restrictions on classes of objects that can be related to each other.
- *Logical constraints.* Logical constraints are the most expressive and complex form of ontologies that use the axioms of full first order, higher order or modal logic.

Some types of these specifications that are classified as ontologies as described by Smith and Welty (2001) correspond to some of a list of ontological categories as previously discussed by Lassila and McGuinness (2001). Such types of specifications include glossaries, thesauri and frames. However, it is a debatable point

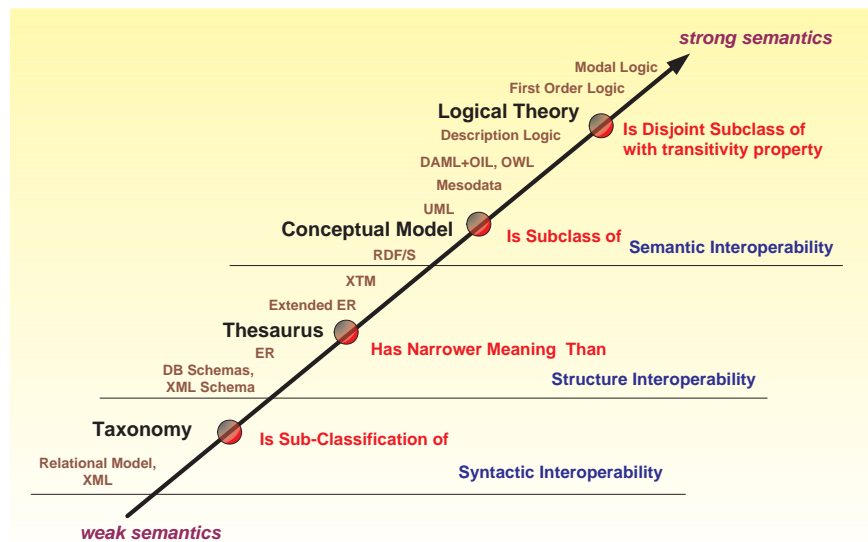


Figure 5.5: Ontology spectrum (adapted from Obrst, 2003).

whether the simple categories such as catalogs, text files and glossaries as discussed above are classed as ontologies. The argument is that such terms are insufficiently qualified to truly define an ontology, viz. an ontology as a representation of a conceptual system that is characterised by specific purposes. This results in a number of different interpretations for the term ontology and is discussed in depth by other authors. In the work of Obrst (2003) and Fromm et al. (2005), the authors provide four different interpretations (taxonomy, thesaurus, conceptual model and logical theory) for the term ontology as an *ontology spectrum* depicted in Figure 5.5. The ontology spectrum describes a range of semantic models of increasing expressiveness and complexity.

Included in the figure as described by Fromm et al. (2005) are models and languages, for example, the relational data model and XML on the lower left followed by XML schema, the ER model, the XML Topic Map Standard (XTM), Resource Description Framework/Schema (RDF/S), Unified Modelling Language (UML), Mesodata, Web Ontology Language (OWL), through to First Order Logic (the predicate calculus). From this point, the spectrum then extends beyond modal logic and becomes largely theoretical. Mesodata, which is a recently developed approach for enhancing a data model's capabilities by providing for more advanced semantics to be associated with the domain of an attribute, has also been included in the ontology spectrum as shown in Figure 5.5.

The four different interpretations of the term ontology are as described in Fromm et al. (2005) and are summarised as follows:

Taxonomy. Taxonomy is the simplest form of semantic model ranging at the low end of expressiveness. A common taxonomy can be considered as a hierarchy (categorising or classifying) of terms with each being a subclass of a more general term. The association between two terms becomes a hierarchical structure like the parent-child relationship. Taxonomies are useful for classifying things, but they are not useful for modelling the meaning of things.

Thesaurus. Thesaurus is a higher order form of semantic model than a taxonomy as its associations contain additional inherent meaning. In other words, a thesaurus is a taxonomy with some additional semantic relationships in the form of a controlled vocabulary. However, a thesaurus is not explicitly a form of taxonomy hierarchy. The nodes in a thesaurus are *terms* meaning words or phrases. These terms have *narrower than* or *broader than* relationships to each other. A thesaurus also includes other semantic relationships between terms such as synonyms.

Conceptual model. Conceptual model is a semantic model in which relationships are explicitly specified. Both conceptual model and logical theories can be considered ontologies, the former a weaker ontology, the latter a stronger ontology. Unlike a taxonomy which is commonly shown as a *tree*, i.e. a hierarchical structure with child nodes having only a single parent, an ontology typically takes the form of a *graph*, i.e. a network with branches across nodes (representing other relationships) and with some child nodes having links from multiple parents. This variability in connectivity provides tremendous flexibility in dealing with concepts since many conceptual domains can not be expressed adequately within either a taxonomy or a thesaurus.

Logical theory. Logical theory is the most expressive form of semantic model ranging at the high end of the spectrum. Simple ontologies are merely subclass hierarchies or a network of connections providing limited clues that a computer can use to determine meaning; richer ontologies can include, for example, rules and constraints governing these connections. Improvements in languages such as OWL that can express more meaning and approaches to model-based programming increase the ability to move from conceptual models to logical theories, enabling computers to do automated reasoning, without the need for human coding steps.

In summary, an ontology may have different definitions and may take a variety of interpretations from simple taxonomies with minimal hierarchy of knowledge (only parent/child relationships) to thesauri (extended vocabularies) including words and synonyms, to conceptual models having more complex relationships, and to logic theories including very complex, expressive and meaningful knowledge.

In addition, since ontologies are widely used for different purposes (e.g. knowledge management, e-commerce, semantic web, intelligent integration information, etc.) in different communities (e.g. knowledge engineering, databases and software engineering), the ontology community distinguishes ontologies that are mainly taxonomies as *lightweight ontologies*, while models offering stricter constraints and therefore more actionable semantics are known as *heavyweight ontologies*. Lightweight ontologies include concepts, concept taxonomies and relationships between concepts and properties that describe concepts. Lightweight ontologies are usually less restrictive. Heavyweight (also sometimes referred to as *fully fledged*) ontologies add axioms and constraints to the lightweight ontologies. Heavyweight ontologies usually provide complete definitions (of concepts, relationships, etc.). This models a domain in a deeper way and provides more restrictions on the semantic domain.

The existing ontology's class hierarchy presented in this chapter is analogous to the lightweight ontology that is defined as a set of elements connected by some structures, i.e. *is-a* hierarchy. The use of mesodata as an ontological concept extends beyond lightweight ontologies as it provides more complex relationships and expressive domain knowledge to conceptual models.

5.2.4 Ontologies versus Conceptual Data Models

Ontologies have both similarities and differences when compared with conceptual data models. In general, an ontology is an explicit specification of a conceptualisation that describes the semantics of data for sharing and reuse and should capture domain knowledge in a machine-readable form in order to provide a consensus and shared understanding of the domain. A conceptual data model is used to model the Universe of Discourse (UoD), entities and relationships corresponding to particular user requirements that are independent of their implementation.

Ontological modelling is concerned with capturing the relevant entities of a domain into an ontology of that domain using an ontology specification language

such as OWL or RDF that is based on a small set of basic, domain independent ontological categories (Guizzardi, Herre and Wagner, 2002). On the other hand, conceptual modelling is concerned with identifying, analysing and describing the essential concepts and constraints to capture a user's view of some domain in a formal conceptual schema with the help of diagrammatic modelling that is expressed in metamodels, like the ER/EER model, NIAM/ORM and UML class diagrams.

There are some similarities between ontologies and conceptual data models in that both are represented by a modelling grammar with similar constructs (El-Ghalayini, Odeh and McClatchey, 2006). For example, *concepts/classes/types* in ontologies correspond to *entity types* in conceptual data models. Thus, the methodologies of developing both models have common activities (Fonseca and Martin, 2007). However, as discussed by Spaccapietra (2008), there are two important differences. Firstly, an ontology targets data description relevant to the wider community whereas a conceptual data model targets data management for a given organisation. Secondly, an ontology describes what is known about the real world (*open world assumption*) while a conceptual data model prescribes how the world of interest is (*closed world assumption*). In the open world assumption, information in the ontology may be incomplete. Instances are accepted as long as they do not contradict the knowledge described, e.g. tertiary students should be linked to a university but if they are not, it could be assumed that this will be done later. Conversely, with the closed world assumption, instances have to comply with the schema and constraints, e.g. students must be linked to a university.

Another obvious point regarding the differences between ontologies and schema definitions as described by Fensel (2001) and Gómez-Pérez, Fernández-López and Corcho (2004) is that an ontology must be a shared and consensual terminology because it is used for information sharing and exchange. An elaborated comparison between database schemata and ontologies can be found in Fonseca and Martin (2007), Meersman (1999) and Spyns, Meersman and Jarrar (2002).

In summary, it is apparent that ontologies and database schemata are closely related. As both are abstract concepts in conceptual data models, the distinction between conceptual schemata and ontology representations is somewhat blurred. Through this thesis research, the major differences between ontologies and conceptual schemata have been identified as follows:

- Ontologies are developed to define the meaning of terms used in some domain while conceptual schemata are developed to model some data, and
- Some ontologies are used to describe a type level including instances at the basic ground level while conceptual schemata are used to describe the type level (no instances).

Without an inclusion of instances, ontologies can be considered as either a sub-type hierarchy or an aggregation hierarchy, which shows a significant similarity between ontologies and conceptual schemata in terms of descriptions of the type level.

5.3 Ontology-Based Conceptual Data Modelling

Conceptual modelling is a very important phase in designing a successful database application (Elmasri and Navathe, 2007). As conceptual modelling accommodates more of the concepts for specifying terms of interest in a particular domain and the relationships among them, it is clear that the information about the domains are strongly related to conceptual modelling. This section presents conceptual modelling that incorporates an ontology as a domain knowledge for database design.

5.3.1 Using Ontology for High-Level Conceptual Data Models

Ontologies can be considered as domain-oriented concepts consisting of terms denoting abstract concepts, their relationship and constraints. Concept definitions in ontology are similar to schema definitions in relational databases as well as classes in object-oriented databases. Ontologies are suited to represent high-level (conceptual) schema, which is independent of the implementation level. The relationships between three associated components; ontologies, conceptual schemata and relational schemata are shown in Figure 5.6.

Figure 5.6 shows a simplified description of the conceptual design that successfully captures a conceptual domain. Ontologies can be incorporated into conceptual design as a conceptual domain schema, extending a high-level conceptual data model, the ER/EER model, ORM and UML class diagram to represent the knowledge of application domain being modelled.

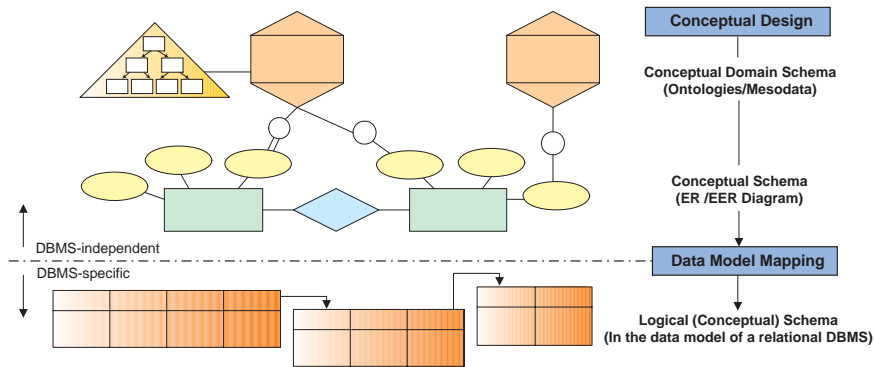


Figure 5.6: A simplified diagram to illustrate the relationship between ontologies and the underlying conceptual (and logical) schemata.

From this research, an ontology can be viewed as a structure of terms of interest in a particular information domain and the relationships among them as shown in Figure 5.1(b) (Page 122). It helps capture the semantics of a domain as it can suggest what terms might appear in an application domain and how they are related to other terms. The main purpose of an ontology is to make the information content explicit in a manner independent of the underlying data structures that may be used to store information in a data repository. Ontologies are thus abstractions and can describe different types of data organisations such as relational tables and textual and image documents (Mena and Illarramendi, 2001).

Another important feature of ontologies is that it must be machine-readable. In conforming with this principle, the conceptual domain schema can be correspondingly transformed from the high-level data model into the implementation data model for use in commercial relational or object-relational DBMSs. This step is called data model mapping or logical design, where the results become the database schema in the implementation model of the DBMS (Elmasri and Navathe, 2007). That is, the structured ontologies from the ER/EER, ORM and UML class diagrams can be processed and translated into SQL.

5.3.2 Ontology's Class Hierarchy Modelling

The concept of Ontology can be easily applied to the term domain as it describes the concepts and the relationships that hold between those concepts representing features of some domain of interest. This chapter suggests that the classical ontology's class hierarchy that is represented by the hierarchical organisation of concepts through the *is-a* relationship, can be introduced into conceptual models

to provide richer semantics and such ontologies can be modelled using mesodata concepts. This chapter uses common domain structures to represent ontological hierarchies. It is expected that a set of built-in mesodata operations would be used to facilitate semantic domain queries using modified database query language syntax.

The ontology's class hierarchy modelling is described in terms of *data structures*, *integrity constraints* and *languages* as follows:

Data Structures. The basic ontological class hierarchy constructs consist of *terms* (also called *concepts* or *classes*) and their *relationships*, basically represented by hierarchical structures. Note that *individuals* (or *instances*)¹ are the basic ground level objects of an ontology. However, an ontology need not include any individuals.

- **Terms.** Terms are the core concepts related to the domain and include abstract groups, sets, collections or type of objects. They may contain individuals, other classes or a combination of both. For example, in the publisher domain, obvious terms include *reviewer*, *author* and *paper*; the term *author* would contain all the individuals that are authors in the domain of interest. Terms are usually hierarchically organised through a structuring relationship such as **is-a** (superclass, subclass) or **part-of**. For example, consider the terms Person, Reviewer and Author — both Reviewer and Author might be a subclass of Person (with Person as superclass of Reviewer and Author). Terms are represented as squares as shown in Figure 5.7 and can be considered as entity types in the ER model. The word *concept* and *class* are sometimes used in place of *term*. Terms and classes are concrete representation of concepts.

¹The research on an ontology is sometimes particularly detailed such as having instances of classes within ontologies while another research suggests that the recording of instances or ground content should be done by the information system itself under the guidance of the conceptual schema (Fonseca and Martin, 2007). In general, ontologies should not include instances of its concepts corresponding to the assertion by Stevens, Goble and Bechhofer (2000) who stated that an ontology should not contain any instances as it is supposed to be a conceptualisation of the domain. The combination of an ontology with associated instances is what is known as a *knowledge base*. Obrst (2003) suggested that ontologies and knowledge bases are distinguished by their types of assertions. Ontologies define generic or class-level assertions, about entities, their properties and relationships. Knowledge bases are instance (or equivalent fact) bases, and define instance-level assertions based on class-level properties and relationships inherited from the ontological assertions. At the ontological level, a Person Lives at a Location Having an Address. At the knowledge base level, 'Lyn' is an instance of a Person and lives at 'the FMC Flats on Flinders Drive, Bedford Park, South Australia' is an instance of a Location Having an Address.

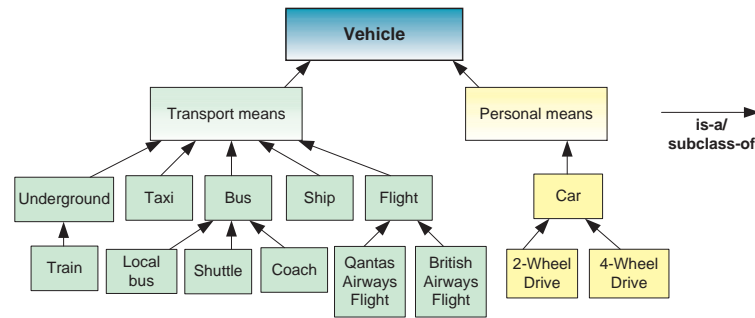


Figure 5.7: An example of a representation of terms and relationships.

- **Relationships.** Relationships represent a type of interaction between terms (or concepts) of the domain. Essentially, the hierarchical organisation of terms through the inheritance (**is-a**) relationship constitutes the backbone of an ontology. Other kinds of relationships like part-whole (**part-of**) or synonym (**syn-of**) or application specific relationships might exist. Furthermore, a set of logical axioms is often associated with the ontology to specify semantics of the relationships (Necip and Freytag, 2005). Relationships can also be either transitive or symmetric (refer to Section 5.2). Relationships are represented as arrows linking terms together. Figure 5.7 shows a representation of terms and relationships linking the terms.

The addition of the **is-a** relationship has created a hierarchical organisation which is also known as *taxonomy*, a tree-like structure (or more generally, a partially ordered set) that clearly depicts how objects relate to one another. In such a structure, each object is the ‘child’ of a ‘parent’ class (some languages restrict the **is-a** relationship to one parent for all nodes, but many do not). In addition to standard **is-a** and **part-of** relationships, ontologies often include additional types of relationship that further refine the semantics they model. These relationships are often domain-specific and are used to answer particular types of questions. An ontology consists of a number of relationships. However, the majority of semantic relationships in an ontology are concerned about the relation between two terms related in the hierarchy by the semantic relationships such as **is-a** or **is-subclass-of**.

The idea behind the modelling of semantic relationships in ontologies is to use the common domain structures provided by mesodata concepts in handling different types of relationship. For example, the hierarchical **is-a** relationship can be easily modelled by a **TREE** mesodata type. Table 5.1 shows an example of the corresponding usages of terms within semantic relationships in ontology and

Table 5.1: Corresponding use of terms between semantic domain relationships and common domain structures

Semantic (domain) relationships between terms	Common domain structures (Mesodata Types)
is-a is-subclass-of related-to part-of instance-of member-of kind-of locatedIn livesIn hasSibling hasChild hasParent hasAncestor Insubtree	TREE
is-a . . . Insubtree closeTo	WTREE (Weighted Tree)
syn-of	SYNONYM
adjacentTo	GRAPH
adjacentTo closeTo inProximity equalTo	WGRAPH (Weighted Graph)
first and last next and previous inbetween...and...	LIST
next and previous inbetween...and...	CLIST (Circular List)

common domain structures.

Integrity Constraints. Constraints provide a method of precisely defining the semantics of data and play an essential role in establishing the quality of a database and its correct evolution (Parent et al., 2006a). Most often constraints are restrictions placed upon attribute values, forcing conformance to application rules. Also, most data models come with integrity constraints that can be specified using predefined clauses of the associated data definition language. For example, the standard SQL allows the specification of uniqueness constraints (using `PRIMARY KEY` and `UNIQUE` clauses) and referential constraints (using `FOREIGN KEY` clauses) (Parent et al., 2006a).

Traditional integrity constraints are limited to dealing with attribute value domains that refer to ontologies. Thus, it is desirable that the deployment of the ontology's hierarchy model will enable attribute values to be validated against the ontology to ensure that only valid attribute values (those belonging to the domain or subset of terms) are stored in the ontology. This *ontology-based referential constraint* (Chong et al., 2006) can be

viewed as an extension to the traditional referential constraint mechanism in a RDBMS that enforces foreign key relationships between a referencing table and another table in the database. The key difference is that the domain of attribute values is identified by the referenced ontology.

The ontology-based referential constraint is similar to the *total domain participation (TDP) constraint* proposed in **OntoER** (Section 5.4) and thus the TDP constraint can be used to restrict attribute values by referencing ontologies. This can be applied to the *ontological mandatory role (OMR) constraint* in **OntoORM** (Section 5.5) and the *ontological type (OT) constraint* in **OntoUML** class diagram (Section 5.6) in the same way as the TDP constraint.

Languages. As an ontology should be machine readable with the data managed by DBMSs, it is desirable that the ontology can be managed using the same framework so that users are able to query ontologies (semantic information) in the same way as querying relational data (instead of dealing with multiple heterogeneous data repositories). Such queries can extract more information out of relational data if the relational data are associated with ontologies in the domain of the relational data. For example, if a column in a relational table contains names of diseases, a query asking for a match on ‘Immunodeficiency Syndrome’, will be able to retrieve rows containing the value ‘AIDS’ providing that this value of ‘AIDS’ from the NCI Cancer Ontology (Mindswap, 2008) is correctly interpreted as a type of ‘Immunodeficiency Syndrome’ (Chong et al., 2006).

As the Mesodata DDL (MDDL) and Mesodata DML (MDML) specified by de Vries (2006) and de Vries et al. (2004) provide extensions to the standard SQL necessary to implement the integration of complex domains into a relational database, it is also possible to utilise them to provide richer ontology-based semantic querying capability.

5.4 The Ontological Entity-Relationship (**OntoER**) Model

Introducing the semantics of the domain into conceptual models can be done by reusing pre-existing ontologies. It is worth considering pre-existing ontologies if they can be refined and extended from existing sources for a particular domain.




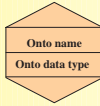



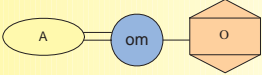
Symbol	Meaning
	Entity Type
	Relationship Type
	Attribute
	Ontological entity type with two Sections: Ontological name and ontological data type.
	Ontological entity type with the bottom section referring to an existing ontology's class hierarchy visually indicated by the  icon.
	Ontological Mapping
	Total Domain Participation Constraint of A in om

Figure 5.8: Symbols of major components in the OntoER model.

Many ontologies are available in electronic forms on the web. For example, a knowledge base of NCI cancer ontology (Mindswap, 2008) captures detailed semantic relationships among genes, diseases, drugs and chemicals, anatomy, organisms and proteins. A relevant ontology of diseases (WrongDiagnosis.com, 2008) provides information about symptoms, diseases and diagnoses.

This thesis presents the Ontological Entity-Relationship (OntoER) model to show how common domain structures can be used to accommodate an existing ontology's class hierarchy into the ER model, thus enhancing conceptual modelling techniques. Additionally, EER extensions are orthogonal to the OntoER extension and can also be included.

5.4.1 OntoER Data Structures

Consistent with MDER, all of the MDER concepts and symbols remain available together with a new concept of an ontological entity type for hierarchies. Ontological entity types and ontological mappings in OntoER are similar to mesodata entity types and mesodata mapping in MDER, respectively.

Figure 5.8 shows symbols for the major components in the OntoER model comprising the basic ER constructs, ontological entity types, ontological mappings and a TDP constraint.



Figure 5.9: A DISEASES ontological entity type with TREE ontological data type of an existing class hierarchy.

5.4.1.1 Ontological Entity Type

An ontological entity type is a complex-domain entity type that represents a domain for an attribute. Each ontological entity type has a name and an ontological data type. Ontological data types are common domain structures such as a tree, weighted tree, graph or weighted graph whose values/instances are structures of the attribute domain typically available together with operators for manipulation and comparison. A mesodata type is mainly used to define such common domain structures in OntoER.

Ontological entity types in OntoER are depicted as hexagons with two sections: the top section for the *name of the ontology (domain)*, the bottom section for *ontological data type*. Ontological entity types referring to an existing ontology's class hierarchy are modelled as TREE in the bottom section for ontological data type together with a triangle symbol enclosing a hierarchy as illustrated in Figure 5.9. In this example, the ontological entity type may have the name DIAGNOSES and an ontological data type TREE, which is well equipped with built-in operators such as INTREE for manipulating values provided in the existing ontology's class hierarchy. Conceptually, to the users' view, ontological schema holds ontology terms whose meaning is specified by ontological data in the system-defined table within the RDBMS.

Thus, a portion of a diagnosis ontology as shown in Figure 5.1(b) (Page 122) can be modelled using a TREE mesodata type to represent a structure of this ontology. By using a TREE mesodata type definition, a set of operations (such as INTREE, SIBLING and DESCENDENT) thus become applicable to the ontology. It has the source schema R (PARENT, CHILD) which has a composite key or attributes Parent and Child as a primary key to describe the structure of the diagnosis ontology. Nodes further down the tree are more specific. Any node in a tree with both a child and a parent can be a *concept* corresponding to the *term* (or *class*) used in the ontology hierarchy. Also, connections between nodes in the tree can be represented by the semantic relationship *is-a* in the hierarchical structure since

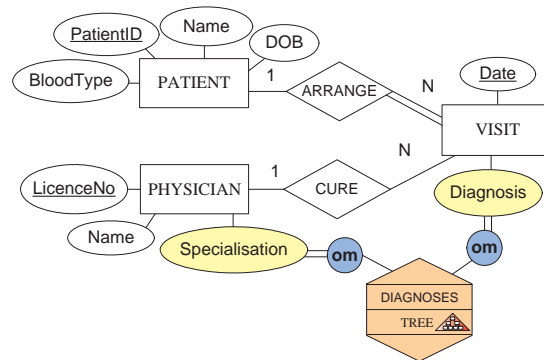


Figure 5.10: An OntoER Example.

the lower of two connected nodes is contained within the upper node. This can also provide richer querying capability that can be performed over the semantic domain. For instance, the operators such as `INTREE` are defined to operate on ontology data over a specific `TREE` ontological data type.

5.4.1.2 Ontological Mapping

The modelling of an existing ontology's class hierarchy requires mapping to convert a data value to a domain value. An ontological mapping between an ontological entity type and an attribute defines a set of mappings required to interpret a data value within a domain context.

Consistent with `MDER` extensions, `OntoER` diagrammatic notation uses the symbol of a circled `om` to represent ontological mapping as shown in Figures 5.8 and 5.10. Figure 5.10 shows how `Diagnosis` and `Specialisation` attributes can be associated with a diagnosis ontology, which is an existing ontology's class hierarchy. A portion of which is shown in Figure 5.1(b) (Page 122).

5.4.2 OntoER Constraints

The `OntoER` model includes all the constraints of the `ER/EER` models. Additionally, an integrity constraint, *total domain participation* (`TDP`), is introduced into the `OntoER` model to restrict attribute values by referencing ontologies. Consistent with the `TMDP` constraint in `MDER`, an attribute that references an ontological data type can be defined to participate either *totally* or *partially*. The `TDP` constraint specifies that every attribute value must correspond to values which exist within the ontology (domain). In other words, data stored in the

attribute must always come from existing terms in that domain. If the TDP is not specified, the value must still adhere to the base type.

In *OntoER* diagrams, the TDP is shown as a double line connecting the participating attribute to the ontological mapping as shown in Figure 5.8. For example, the double line connecting *Diagnosis* to the ontological mapping as shown in Figure 5.10 indicates TDP, meaning that for the population of *Diagnosis*, every instance must be taken from one of the values stored in the *DIAGNOSES* ontology. This constraint also imposes on the *Specialisation* attribute (Figure 5.10). The domain values for these attributes are generated from a portion of diagnosis ontology shown in Figure 5.1(b) (Page 122).

5.4.3 *OntoER* Languages

As discussed earlier (Chapter 4 and Subsection 5.3.2), Mesodata DDL (MDDL) and Mesodata DML (MDML) are extensions to SQL to enable the integration of ontologies into a relational database, removing the need for specially written application code to manage the ontological structures. Using these languages can provide a powerful mechanism to map ontology-based semantics directly into SQL.

The use of the language support for *OntoER* is shown in the transformation of an *OntoER* schema into a relational schema as discussed in Chapter 8. In particular, examples of the use of MDDL is demonstrated in the process associated with specifying schema definitions in that chapter.

5.5 The Ontological Object Role Modelling (*OntoORM*) Model

Object Role Modelling (ORM) is a conceptual modelling approach that has both verbal and graphical syntax and is based on a sound theoretical foundation. In recent years, ORM has gained much interest as a tool for modelling business rules. Like the ER model, ORM does not adequately cope with the semantics of specific domain knowledge. To address this, an extension to ORM is proposed, termed the Ontological Object Role Modelling (or *OntoORM*) that allows an existing ontology's class hierarchy, which describes a hierarchy of values, to be modelled with label types in ORM.



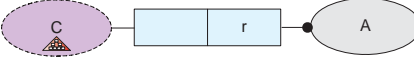
Symbol	Meaning
	Ontological label type C (where C denotes common domain structures e.g. Tree, WTree or Graph)
	Ontological label type C (to which values reference an ontology's class hierarchy)
	Ontological mandatory role constraint (ontological role r is mandatory for the population of A)

Figure 5.11: An extension of the ORM Symbols .

5.5.1 OntoORM Data Structures

Consistent with MDORM, all of the OntoORM concepts and symbols remain available together with a new concept of ontological label types for hierarchies. The ontological label types and ontological mandatory role constraints in OntoORM are similar to ontological value types and mesodata mandatory role constraints in MDORM. Figure 5.11 shows symbols for the major components in the OntoORM model.

Building upon the basic ORM model, the common domain structures are applied to a label type (or a value type), termed an *ontological label type*. An ontological label type is a label type whose values are associated with ontologies. This is represented by the name of the common domain structure that is enclosed by a dashed ellipse (refer to Figure 5.11). The triangle symbol enclosing a hierarchy for label types denotes the values referenced within an (existing) ontology's class hierarchy.

Each ontological label type is annotated as a named ellipse, with dashed lines. A name of an ontological label type is denoted by common domain structures (e.g. a tree, graph, weighted tree or list). Figure 5.12 shows a TREE ontological label type with the triangle symbol enclosing a hierarchy associated with a role played by a Disease entity type. Such an association between an entity type and an ontological label type is termed an *ontological reference type*. The ontological reference type shown in Figure 5.12 refers to a Disease where its code's values are structured as a TREE and which is to be found within a hierarchy of values of an existing ontology's class hierarchy.

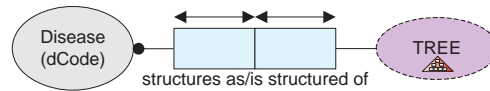


Figure 5.12: A **Disease** where the disease’s codes are associated with a **TREE** ontological label type and an ontological mandatory role constraint.

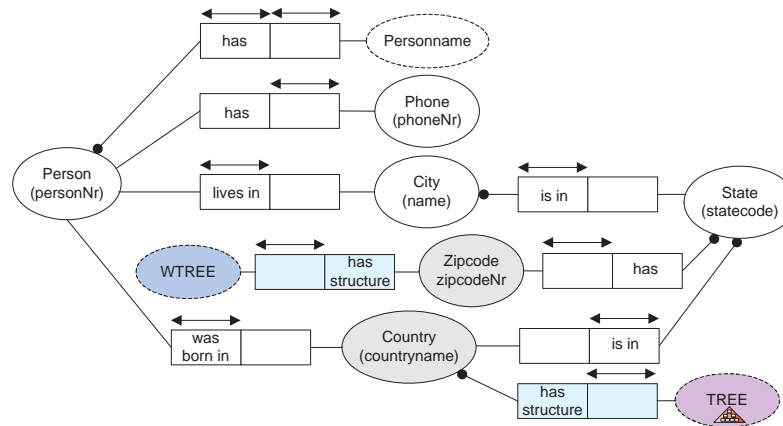


Figure 5.13: A simple OntoORM diagram.

5.5.2 OntoORM Integrity Constraints

Similar to the TDP constraint discussed in **OntoER**, the **OntoORM** model includes all the constraints of the ER/EER models. Additionally, **OntoORM** requires a constraint to indicate that the values of the object type must be taken from one of the values recorded in the ontology — the *ontological mandatory role (OMR) constraint*. This constraint is denoted as a black dot on the object type connecting to the roles of ontological reference types which are relationships between entities and ontological values (examples in Figures 5.11 and 5.12). Figure 5.12 shows the ontological mandatory role constraint declaring that every instance in the population of the role’s object type involving ontological values must play that role. This constraint is similar to a conventional ORM’s mandatory role constraint. The difference is that it imposes on all instances of the object type associated with the ontological reference types rather than just those involving the conventional ORM fact types or reference types.

Figure 5.13 shows a simple **OntoORM** model. Object types are displayed as named ellipses, with solid lines for object (entity) types and dashed lines for label (value) types. Label types that are labelled with common domain structures are ontological label types. Associations are named sequences of roles, where each role appears as a box connected to the object type playing it. Associations

that indicate the reference scheme (1:1 naming convention) may be abbreviated in parentheses (e.g. `personNr`) next to the name of the entity type. A black dot indicates that the role is mandatory (e.g. each Person has a Name). Arrow-tipped bars over one or more roles are uniqueness constraints, indicating that each instance populating that role sequence is unique. This example includes three uniqueness constraints for binary associations: N:1 (for example, each person lives in at most one city, and each city is lived in by many persons); 1:N (each person may have many phone numbers, but each phone number belongs to at most one person); and 1:1 (each person has at most one name, and each name identifies at most one person). A dashed ellipse **TREE** indicates an ontological label type. The black dot indicates that a role (between entities and ontological values) is mandatory (e.g. each Country has a **TREE** structure) meaning that the ontological mandatory role constraint is imposed on an object type's population associated with ontological reference types. For the population of country name, each instance must be taken from one of the values stored in the ontology.

5.5.3 OntoORM Languages

Consistent with OntoER, OntoORM uses the Mesodata DDL (MDDL) and Mesodata DML (MDML) (de Vries, 2006; de Vries et al., 2004), extensions to SQL, to provide richer ontology-based semantic querying capability.

The use of the language support for OntoORM is demonstrated in the transformation of an OntoORM schema into a relational schema as discussed in Chapter 8. In particular, examples of the use of mesodata data definition commands is demonstrated in the process associated with specifying schema definitions in that chapter.

5.6 The Ontological Unified Modelling Language (OntoUML) Class Diagram Model

The Unified Modelling Language (UML) is becoming widely used as a common language for creating models of object-oriented software and its acceptance as the Object Management Group (OMG) standard is helping it gain wide support in industry (Halpin, 1999b; OMG, 2005). Several types of diagrams e.g. diagrams for (a) use cases, (b) static structures (class and object diagrams), (c) behaviour (communication, formerly known as collaboration, state-chart, activity

and sequence diagrams), and (d) implementation (component and deployment diagrams) (Halpin, 1999b) are provided by UML to assist developers of object-oriented programming. For data modelling purposes, UML² uses class diagrams to show the important abstractions in the system and their interrelationships.

The main constructs of UML are *classes*, *attributes*, *associations* and *multiplicity constraints*. Classes have properties in the form of attributes and provide abstract services in the form of operations. In addition, classes can be related to each other through associations. Typically, classes in UML are depicted as named rectangles with three sections: the top section for the name of the class, the middle section for the attributes of the class and the bottom section for the operations (methods) of the class. For the purpose of this discussion, the section on operations is ignored. Whilst UML includes multiplicity constraints on the association roles, it does not have a standard notation for attribute uniqueness constraints. This may reflect the nature of object-oriented implementation where objects can be uniquely identified by system generated object identifiers called *oids*.

The underlying concept of UML may seem to be similar to that of ER diagrams. If the method/operation property of a class is ignored, it is reasonable to suggest that object modelling is similar, in concept, to data modelling (Shah and Slaughter, 2003). A *class*, an *association* between classes and a *multiplicity* of attributes in UML corresponds to an *entity type*, *relationship type* and *cardinality* in the ER model, respectively. Detailed discussions on UML may be found in Booch et al. (2005).

One of the fundamental differences between UML (as well as the ER model for that matter) and ORM is that whenever an attribute is used in UML, ORM uses a relationship instead. As is the case for the ER model and ORM, modelling an ontology in UML has not been well investigated. In this respect an extension to UML is proposed, termed the Ontological Unified Modelling Language (or *OntoUML*) Class Diagram model. This model will allow ontologies to be included in the conceptual schema by defining *ontological class types* and *ontological type constraints*.

²This chapter will use the abbreviation UML to mean UML class diagrams

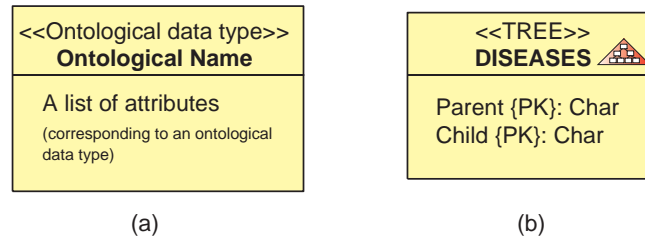


Figure 5.14: A representation of ontological class types (a) general notation (b) example of a DISEASES ontological class type.

5.6.1 OntoUML Data Structures

The OntoUML class diagram defines an ontology (i.e. domains of an attribute) independently of the usual class, termed *ontological class types*. These ontological class types are typically depicted as rectangles with two sections: the top section for the ontological name and ontological data type of the class, the bottom section for the attributes of the class as shown in Figure 5.14(a).

Figure 5.14(b) shows an ontological class type DISEASES with a TREE ontological data type that refers to an existing ontology’s class hierarchy, as represented by the addition of the triangle symbol enclosing a hierarchy. A list of attributes comprising Parent and Child corresponds to a TREE ontological data type. The symbol {PK} indicates a primary identification.

5.6.2 OntoUML Integrity Constraints

From a more general perspective, it is also necessary to allow users to express integrity constraints at the conceptual schema level (Parent et al., 2006a). For modelling applications dealing with ontologies, constraints in a domain knowledge should be addressed. Similar to the TDP and OMR constraints discussed in OntoER and OntoORM models, respectively, the OntoUML class diagram introduces the *ontological type (OT) constraint* to enable attribute domain values to be validated against the ontologies. Ontological type constraints are used to restrict possible domain values of the attributes in that these values must be taken from one of the values recorded in the ontology.

UML allows standard and user-defined constraints to be added in braces. In OntoUML class diagrams, the notation {OT} is used after the attribute name to signify the ontological type constraints, as shown in Figure 5.15. In UML, the more complex constraints can be specified informally in an attached note. In the

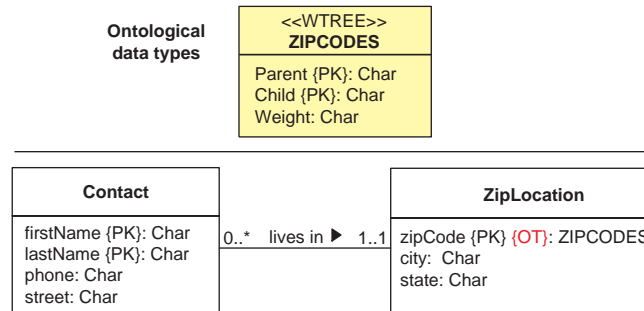


Figure 5.15: A simple OntoUML class diagram.

absence of a standard UML notation, it is at the discretion of the modeller to decide whether such constraints are specified and it is perhaps not surprising that for many of the UML models that one encounters in practice, these constraints are simply omitted (Halpin, 1999b).

Figure 5.15 shows a simple OntoUML class diagram of the geographical location whose zip code’s domain refers to an ontology. The ontological class types define the domain of zip codes independently of the ZipLocation class. The ZipLocation class has the additional attributes of the city and state where it is located. In UML the domain of any attribute may optionally be displayed after it, suffixed by a colon (Halpin, 1999b). In OntoUML class diagrams, the ontological domain for the zipCode attribute are identified by appending the ontological name. For example, ZIPCODES could be appended after zipCode to provide the ontological domain of an attribute.

Relationships in UML, which are referred to as *associations*, are drawn as lines between classes. Lines can be enhanced with relationship names and multiplicities. In Figure 5.15 *lives in* is a relationship name, with the black arrow indicating the direction in which the relationship is read. Multiplicities³ are the same as cardinalities in the ER/EER model. The 0..* (or *) multiplicity indicates ‘zero or more’. The 1..1 multiplicity for ZipLocation denotes total participation and a single value in the association. This association between ZipLocation and Contact can be described as ‘*Each contact lives in one and only one ZipLocation*’ and ‘*Each ZipLocation is home to zero or more Contacts*’.

The ontological type constraints should be specified in the schema. This concern is consistent with the context recommended by Parent et al. (2006a) that integrity constraints should be expressed at the conceptual schema level. In

³A star (*) represents the many side of a 1:N or M:N association. The number 1 indicates the one side of a total 1:1 or 1:N association. The notation 0..1 denotes partial participation in the association.

OntoUML class diagrams, the {OT} constraint applied to the zipCode attribute indicates that the values of the attribute domain must be taken from one of the values stored in the ontologies in the same way as the TDP and OMR constraints in OntoER and OntoORM model, respectively.

5.6.3 OntoUML Languages

Consistent with OntoER and OntoORM, OntoUML uses the Mesodata DDL (MDDL) and Mesodata DML (MDML) (de Vries, 2006; de Vries et al., 2004), which are extensions to SQL, to provide a powerful mechanism to map ontology-based semantics directly into SQL.

The use of the language support for OntoUML is shown in the transformation of an OntoUML schema into a relational schema as discussed in Chapter 8. In particular, examples of the use of mesodata data definition commands are demonstrated in the process associated with specifying schema definitions in that chapter.

5.7 Example of OntoER, OntoORM and OntoUML Class Diagram Schemata

Consider a MEDICAL database schema that keeps track of patients, their diagnosed diseases, drug treatments and their allocated physicians. Figures 5.16, 5.18 and 5.19 show the ontological schema for such a database based on the ER, ORM and UML class diagram, respectively.

During the design phase of this database, discussions between a user and the analyst/modeller often contains examples of the queries that an organisation wishes to be able to execute over their data. Consider the following queries, none of which is particularly unusual:

List all patients who are diagnosed with immune deficiency conditions,

List all physicians who are specialised in immune disorder,

List all physicians who study at a medical school in Australia,

List all physicians who live close to Flinders Medical Centre, and

List all patients who live close to their physicians.

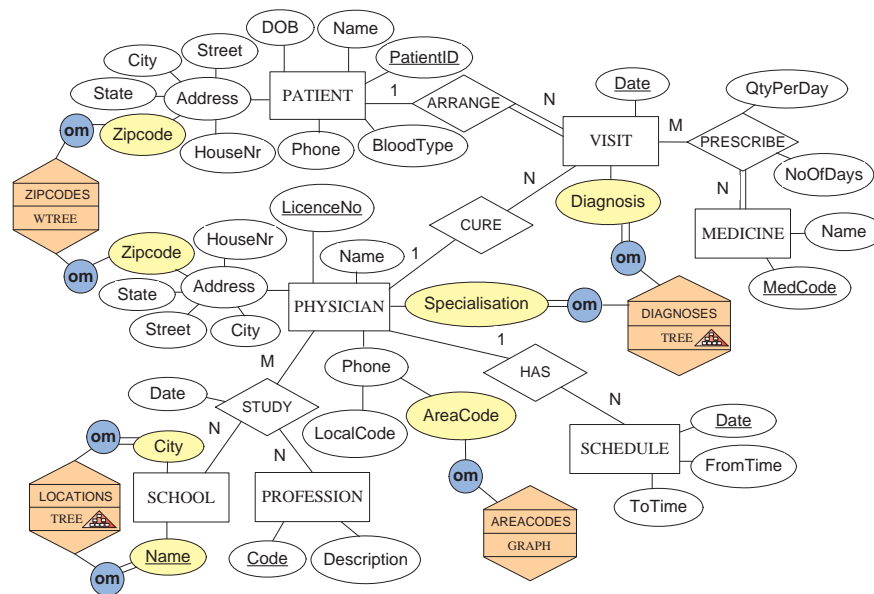


Figure 5.16: OntoER schema for an example MEDICAL database.

A conventional database application could express the queries using an equality operation as shown, for example, in the following way:

```

SELECT      PatientID
FROM        visit
WHERE       Diagnosis = 'Immune deficiency conditions';

SELECT      Name
FROM        physician
WHERE       Specialisation = 'Immune disorder';

```

According to the data shown in Figure 5.1(a) (Page 122), none of the above queries will return any results. The first would not identify patients diagnosed with 'AIDS', 'Diabetes' or 'SCID' since none of those terms identically match the term 'Immune deficiency conditions'. Likewise, the second query would not identify physicians specialised in 'AIDS', 'Diabetes' or 'SCID' since none of those terms identically match the term 'Immune disorder'. These queries produce false results as they fail to associate the term 'Immune deficiency conditions' with other more specific types identified by the terms 'AIDS', 'Diabetes', or 'SCID', as shown in the diagnosis ontology that is graphically illustrated in Figure 5.1(b) (Page 122).

Similarly, the domain knowledge providing the location of a medical school is required to answer the third query. Without a semantic support, like the diagnosis ontology (Figure 5.1(b)) or location ontology (Figure 5.17), it is difficult for the DBMS query processor to solve such vocabulary ambiguities. In this case, such

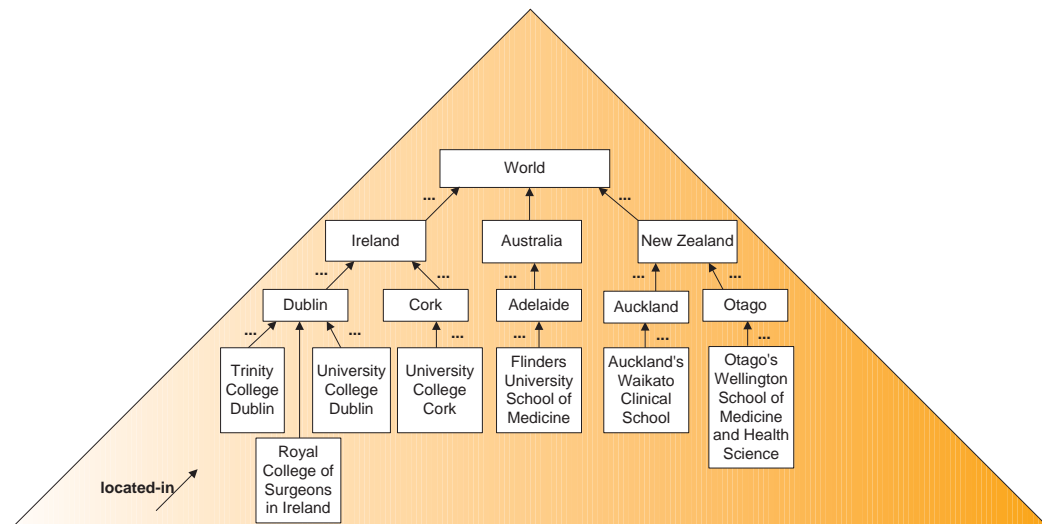


Figure 5.17: A portion of the medical school location ontology.

ontologies provide additional meanings for the database values related to a certain attribute. These meanings are expressed through the relationships between the corresponding concepts (Necip and Freytag, 2003).

Clearly, the domain knowledge required to answer such queries is not present in the relational table. A conventional database system does not take into account the semantics pertaining to a specific domain. In order to provide semantically correct answers, the DBMS must have access to definitions that describe the meaning of referenced terms or concepts.

Similarly, the point of interest of the fourth and fifth queries lies in the understanding of data semantics rather than simple transactional or analytical data processing. In processing these queries, the different aspects of address/location such as *proximity* or *adjacency* are of particular relevance. Thus, the DBMS must also be able to identify those locations with zip codes that are considered geographically close to that of the location of primary consideration.

Each of these queries requires knowledge of the domain. For the example in Figure 5.16, the attributes Diagnosis, Specialisation, Name, City, AreaCode and Zipcode are identified as requiring knowledge of the domain through their reference to the associated ontologies. In the particular case of the Diagnosis and Specialisation attributes, these are associated with the DIAGNOSES ontology of an existing ontology's class hierarchy (shown in Figure 5.1(b), Page 122). The double line from Diagnosis and Specialisation to the circled **om** indicates a *total* participation, meaning that every attribute value of Diagnosis and Specialisation must be recorded in the DIAGNOSES ontology.

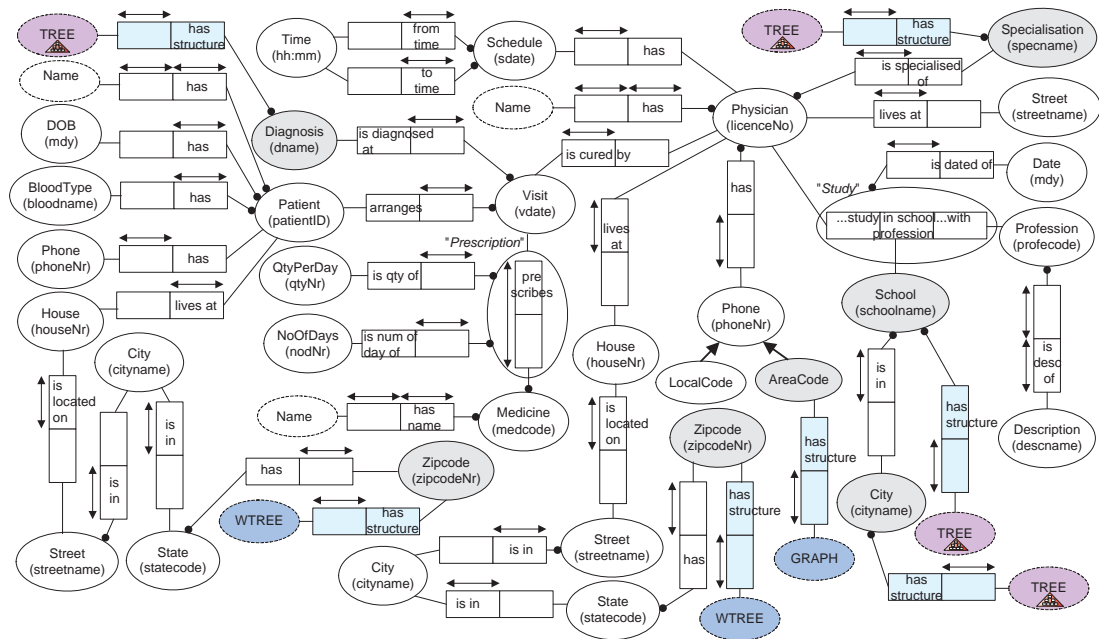


Figure 5.18: OntoORM schema for an example MEDICAL database.

In most cases, the values that constitute a domain are implied through its data type. Similarly, the values that constitute an ontological domain are specified by ontological data types using the common domain structures provided by the mesodata types. Applying this framework to the given example, the Name and City attributes refer to the LOCATIONS ontology of an existing ontology’s class hierarchy (shown in Figure 5.17). In this process, the TDP constraint is also applied to both attributes in the same way as the Diagnosis and Specialisation attributes.

In Figure 5.16, the Zipcode attribute of PATIENT and PHYSICIAN entity types refers to the ZIPCODES ontology, the domain structures of which are generated according to a weighted tree mesodata type. The single line from the Zipcode attribute to the circled om indicates a *partial* participation, meaning that some or part of the Zipcode attribute values adhere to the base data type. Likewise, the AreaCode attribute refers to the AREACODES ontology, the domain structures of which are generated according to a graph mesodata type. Like the Zipcode attribute, the TDP is not mandatory restricted, meaning that the AreaCode attribute values adhere to the base data type.

By accommodating ontologies into conceptual modelling, it is no longer difficult to express queries that manipulate both the data and their meaning. In response to the previous queries example, such queries can now be expressed as follows:

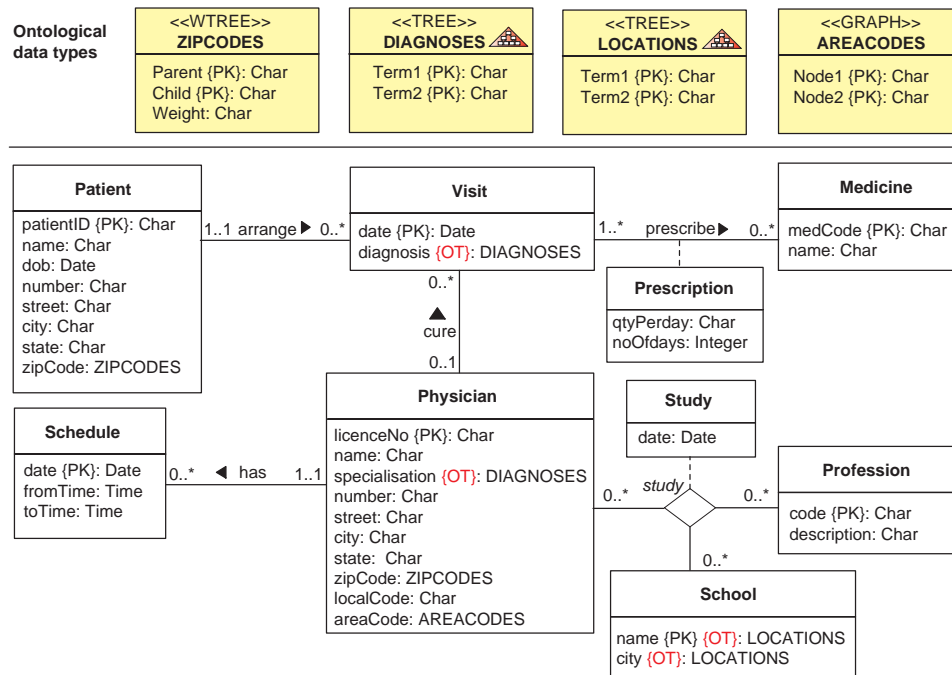


Figure 5.19: OntoUML class diagram for an example MEDICAL database.

```

SELECT      PatientID
FROM        visit
WHERE      Diagnosis INTREE 'Immune deficiency conditions';

SELECT      Name
FROM        physician
WHERE      Specialisation INTREE 'Immune disorder';

SELECT      LicenceNo
FROM        study
WHERE      Name INTREE 'Australia';

SELECT      Name
FROM        physician
WHERE      Zipcode CLOSETO 'Flinders Medical Centre';

SELECT      PatientID, Name
FROM        patient p, visit v, physician ph
WHERE      p.PatientID = v.PatientID
AND        v.LicenceNo = ph.LicenceNo
AND        p.Zipcode CLOSETO ph.Zipcode;

```

Incorporating ontology-based semantics into conceptual modelling greatly enhances modelling of the semantic domain of database applications. Applications that have to work with domain-specific knowledge such as Healthcare applications, BioInformatics and Geographical Information Systems can take great advantage of this facility.

5.8 Summary

Bringing together the concepts of common domain structures provided by meso-data and existing ontology's class hierarchies allows for the benefit of both practices to be merged, providing further research opportunities in incorporating ontologies into conceptual modelling techniques. The ER model, ORM and UML have a well-accepted conceptual model that can be used for capturing user's requirements. However, to establish a better communication between designers and users to understand the semantic domain, it is important that ontologies are explicitly represented in conceptual modelling.

In this chapter, a mechanism for incorporating ontologies in conceptual modelling techniques has been presented — the Ontological Entity-Relationship (OntoER) model, Ontological Object Role Modelling (OntoORM) and Ontological Unified Modelling Language (OntoUML) Class Diagrams. These aim to support a rich class of the ontology-based semantics in conceptual modelling that can be adopted through the common domain structures based on mesodata concepts. The new proposed conceptual framework facilitates the reuse of existing ontology's class hierarchy through the reusable **TREE** mesodata type that enriches the semantic expressiveness of data modelling using simple enhancements to standard SQL.

As well as promoting the incorporation of the backbone ontologies that are represented by the hierarchical organisation of concepts through the inheritance (**is-a**) relationship in conceptual data model, it is expected that other kinds of common semantics used in information modelling like *synonym* and *related-to* can also be included using the mesodata concept. The work presented here describes an attribute's domain within the context of a single ontology. This concept could be extended to address the case when the domain of an attribute references concepts from multiple ontologies. An additional suggestion for future research is in the use of (and combinations of) other kinds of various relationships (e.g. the spatial-temporal) or the semantics of a particular application associated with the ontology, to specify semantics of the relationships within the domain knowledge.

Chapter 6

Polymorphic Relationships in Entity-Relationship Modelling

This thesis chapter serves to provide some of the answers to support Objectives 1 and 2 of the thesis as stated in Chapter 1 (particularly Section 1.5.1) and as indicated in Figure 1.3. The focus of this chapter is in exploring the conceptual modelling extension to incorporate the concept of links as polymorphic relationships and how this concept can be used to increase the expressiveness of conceptual modelling approaches. Despite the flexibility offered by the existing ER/EER models there remains a number of conceptual modelling problems that have not been adequately resolved. Many of these relate to the relatively restricted (and static) modelling of relationships and cardinality constraints. This chapter explores the utility of *links* as *overloaded polymorphic relationships* and shows how some of the outstanding conceptual modelling issues can be handled in an intuitive manner if a revised view of links is adopted.

The structure of this chapter is as follows. Section 6.1 provides an introduction to this topic. Existing relationships as well as cardinalities on typical ER modelling relationships are examined in Section 6.2. Next, problems and motivating examples are addressed in Section 6.3. In Section 6.4, the term *link* is introduced to represent overloaded polymorphic relationships. Three additional issues, cardinality, default value and precedence, are raised in Section 6.5 in the context of modelling overloaded polymorphic relationships, followed by a summary in Section 6.6.

6.1 Introduction

Much of the research that deals with the understanding of the real world and representing it in a conceptual model uses some form of the entity-relationship model as a means of representation (Storey, 2005). Most conceptual modelling methodologies are concerned with entities and relationships between entities. Relationships between entities are fundamental to the representation and support of semantics. Until now, the focus has been on simple hierarchical structures of relationships such as *is-a/role-of*, *instance-of/member-of* and *is-part-of*. However, real world relationships between entities are much more complex and it may not be possible to represent them by using the basic relationships of the ER model. Relationships across domains may not necessarily be hierarchical in nature and their evaluation may require complex information requests involving user-defined functions and fuzzy (or approximate) matching of objects (Sheth et al., 2003). For example, consider the relationship ‘earthquake causes tsunami’. The location and time of the tsunami are largely indeterminant, requiring a fuzzy (or approximate) calculation based on the initial earthquake event. Thus, the temporal and spatial proximity between two events are the main aspects of this relationship (Sheth et al., 2003).

While research on the nature of relationships has received some attention in the literature, many database design practices restrict their use to simple, binary relationships to represent associations between entities (Ullrich, Purao and Storey, 2000). One of the most difficult problems, and one of the most common errors in database design, is the misrepresentation of relationships or incorrect decisions on whether to use ternary (or other higher-degree relationships) or multiple binary relationships to handle complex modelling (Song and Jones, 1993). As the choice between an n -ary relationship or multiple binary relationships is often not clear, the appropriateness of the conceptual design depends heavily on the semantics of the application (Elmasri and Navathe, 2007). Proposed extensions that help the designer to identify the correct degree of a relationship and to represent a relationship properly include an analysis of binary relationships within a ternary relationship in ER modelling (Song and Jones, 1993), a generalised framework for analysing relationships during conceptual modelling of real-world applications (Dey, Storey and Barron, 1999), and a comprehensive ontology for classifying the semantics of relationships (Purao and Storey, 2005).

In addition, other types of relationship have been proposed to extend the expressiveness of the ER model. For example, the superclass/subclass (or *is-a*)

relationship was introduced as part of the EER model and higher-order relationships were introduced to form the HERM model (Thalheim, 2000). In HERM, the concept of a higher-order relationship allows relationship types to be defined with components that are also relationship types. In other words, this model permits relationships among relationships. NIAM/ORM (Halpin and Morgan, 2008; Nijssen and Halpin, 1989) also allows these kinds of relationships. The representation of the semantics in relationships has been extended using object-oriented concepts, such as aggregation or relationship types that are a component of another relationship type (Fahrner and Vossen, 1995; Rochfeld and Negros, 1992; Storey, 1991). There remains, however, a number of outstanding conceptual modelling problems and this thesis argues that these can be addressed by allowing an overloaded form of polymorphic relationship.

As discussed by Bachman (1996), polymorphism is another major feature of object-oriented design. Polymorphism may manifest itself in several forms in object-oriented systems. One of these forms is that which is called a *complex relationship*. Complex relationships support the idea that an entity may be associated with a set of entities whose exact nature is not fully known. All that is known is that they have a certain behaviour pattern through which they are related to the first entity. In addition, certain types of relationship may bind the instance of one entity type to instances of two or more second entity types. To deal with this situation, polymorphism has the ability to define additional constraints that apply to the various instances that are involved.

Polymorphism comes in two forms — *ad-hoc* and *universal* (Cardelli and Wegner, 1985). Overloading is an example of the former, in which the number and type of the parameters supplied determines the behaviour. In the case of overloaded functions, the number and type determine the specific function invoked while in the case of a relationship, it determines the semantics of the relationship in terms of the participating entities. While polymorphism has previously been explored in terms of changes in type for ORM (Halpin and Proper, 1995), this is the first instance of research dealing with changes in the number of participating instances.

This chapter introduces the concept of a *link* as a specialised form of polymorphic (overloaded) relationship. This chapter highlights one of the deficiencies of conceptual modelling in dealing with the situation where the specific information for any relationship is indeterminate and shows how this can be overcome through the use of overloaded relationships by retaining the same attached entities but allowing for variations on the number of instances an entity can play a role in

a relationship. Two distinct aspects of polymorphic relationship modelling are variable numbers of entity instances participating in a relationship and the identical semantics of each entity instance participating in a relationship. The formal definition of link types is presented, along with cardinality constraints that can be imposed on links. Finally, the two issues of default values and precedence are discussed.

6.2 Review of Relationship Types in Conceptual Modelling

Relationships are an important part of conceptual modelling since they represent associations between entities. In addition to serving as a connection between entities or between a pair of classes, a relationship carries inherent semantics in the form of constraints and other functionalities. These include inheritance and specialised query capabilities that allow a given organisation to be modelled more precisely (Halper, Liu, Geller and Perl, 2003; Storey, 1993; Woods, 1988). This section reviews the expressiveness of relationships including the cardinalities and participations of relationships that are used to describe some characteristics of the relationship.

6.2.1 Types of Relationship

Modelling constructs are an important component of conceptual models and are a strong determinant of the expressive power of the model. Most conceptual modelling methodologies are constructed with entities and relationships. Relationships are thus one of the fundamental constructs in conceptual modelling. A typical relationship is usually defined as an association between two or more entities. The number of connections between these entities and the relationship determine the degree, order and dimension of the relationship (Rochfeld and Negros, 1992). Cardinalities indicate the associative capacity of each entity involved in the relationship.

Capturing some of the semantics of the real world are clearly represented through the use of relationships. Halper et al. (2003) discussed that a semantic relationship is a data modelling construct that connects between entities and has inherent constraints and other functionalities that precisely reflect the characteristics of the specific relationship in an application domain. Examples

of semantic relationships include **is-a**, **part-whole**, **role-of**, **ownership** and **materialisation**. Such relationships are important in the construction of data models for advanced applications.

Various types of relationship (or association) are used in modelling the real world. Three common types of relationship that are defined in the literature (Dahchour, Pirotte and Zimányi, 2005; Kerschberg and Weishar, 2000; Potter and Kerschberg, 1988) are summarised as follows:

1. **Classification.** A classification relates a class with a set of objects sharing the same properties via **is-instance-of** or **is-of** relationships. An object must be an instance of at least one class. Classifications provide a means of grouping specific object instances together which can be considered to be an object type (Kerschberg and Weishar, 2000). For example, **James** is an instance of class **Person**.
2. **Generalisation.** A generalisation¹ relates superclasses to their specialisations² called *subclasses*. In other words, object types are abstracted into a higher level object type via an **is-a** relationship. For example, **Vehicle** is a generalisation (or superclass) of **Car**, or **Car** is a subclass of **Vehicle**. Subclasses inherit all properties (attributes, methods, roles and integrity constraints) from their superclasses. Subclasses may have defined new properties.
3. **Aggregation.** An aggregation represents a relationship, in which one entity represents a larger entity (whole or composite), consisting of smaller entities (parts). This kind of relationship is also known as **is-part-of**, **has-a** or **part-whole**. For example, **Car** is an aggregation of **Body**, **Engine** and **Wheel**.

These generic relationships are powerful *abstraction* mechanisms that are used in conceptual modelling (Batini et al., 1992). Data abstractions are well-accepted in conceptual modelling and have been applied to all of these listed types of relationship. Generalisation and aggregation were later adopted by the object-oriented models.

¹Generalisation is the process of minimising the difference between entities by identifying their common features (Connolly and Begg, 2004).

²Specialisation is the process of maximising the differences between members of an entity by identifying their distinguishing features (Connolly and Begg, 2004).

As discussed by Connolly and Begg (2004), the two main purposes of data modelling are to assist in the understanding of the meaning (semantics) of the data and to facilitate communication about the information requirements. The additional concept of aggregation should only be used when the organisation data is too complex to easily represent using only the basic relationships of the ER model.

Other types of relationship such as `membership` and `temporal` are also important to data modelling (Kerschberg and Weishar, 2000; Potter and Kerschberg, 1988) as discussed below:

Membership. A membership is an abstraction mechanism that especially supports the `is-member-of` relationship between entity (object) types.

Temporal. A temporal relationship is used to model specific tasks that are related to the relative distances between events or entities in time or space.

6.2.2 Constraints on Relationship Types

Relationship types usually have certain restrictions that limit the possible combinations of entities and relationship types that may participate in the corresponding relationship set (Elmasri and Navathe, 2007). For binary relationships in the ER model, two main types of constraints are defined as *cardinality* and *participation* as described below:

Cardinality Constraints. A cardinality constraint specifies the *maximum* number of relationship instances that an entity can participate in. In Elmasri and Navathe (2007), this is called a cardinality ratio or maximum cardinality constraint. Often, only two values, ‘one’ and ‘many’, are considered for the maximum cardinality constraint. The cardinality constraints of a binary relationship are generally referred to as a one-to-one (1:1), one-to-many (1:N) and many-to-many (M:N). For example, in Figure 6.1 the `manages` relationship is 1:1 with a maximum value of 1 on both sides of the relationship.

Participation Constraints. A participation constraint specifies the *minimum* number of relationship instances that an entity can participate in. In Elmasri and Navathe (2007), this is also called existence dependency or a minimum cardinality constraint. At least two values, ‘zero’ or ‘one’ are

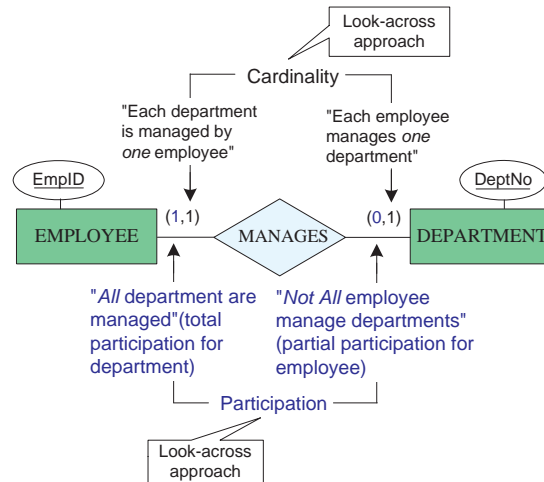


Figure 6.1: Cardinality and participation constraints for the *employee manages department* (1:1) relationship.

considered for the minimum cardinality constraint. The participation of entities in a relationship appears as the minimum values for the relationship. If the minimum cardinality = 0, then the participation of an entity in a relationship is called a *partial* (optional) participation. If the minimum cardinality = 1, then the participation of an entity in a relationship is called a *total* (mandatory) participation.

Note that maximum cardinality, as always, must be greater than or equal to the minimum cardinality to ensure that nonsensical cardinalities, such as (2,1), are avoided for the relationship (Storey, 1993). It is important to note that the participation for a given entity in a relationship is represented by the minimum value on the *opposite* side of the relationship (Connolly and Begg, 2004). For example, in Figure 6.1, the partial participation for the **employee** entity in the **manages** relationship is shown as a minimum value of 0 beside the **department** entity and the total participation for the **department** entity in the **manages** relationship is shown as a minimum value of 1 beside the **employee** entity. The participation constraint in Figure 6.1 can also be denoted using a double line and a single line to represent the respective total and partial participation as shown in Figure 6.2.

As discussed by Galindo et al. (2006), the (min,max) notation can be used to express participation and cardinality constraints since if min = 0 then the relationship is dealing with a *partial* participation, and if min > 0, the relationship is dealing with a *total* participation. Alternatively, if max = 1, the relationship

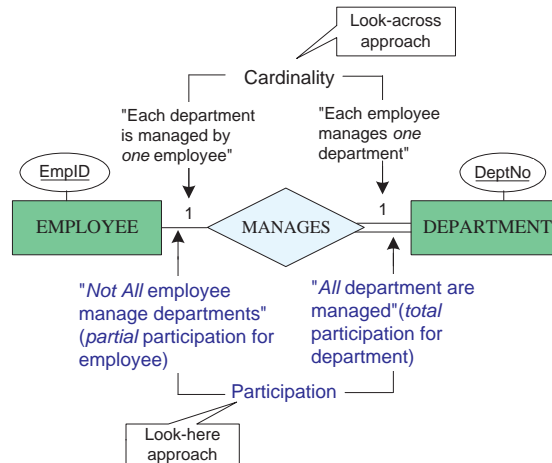


Figure 6.2: Another representation of cardinality and participation constraints of Figure 6.1.

will be 1:1 or 1:N (on the side of 1), and if $\max > 1$ (or $\max = N$), the relationship will be M:N or 1:N (on the side of N).

The literature definition of *cardinality* constraints are varied. In Chen's original ER model (Chen, 1976), the cardinality constraint can be defined as *look-across* constraints based on how many entities can be seen through a relationship type of a certain entity (Thalheim, 2000). Alternatively, the interpretation of the participation constraint can be defined as *look-here* (Hartmann, 2003). This has led to two different notations of expressing the cardinality and participation constraints in relationships. In a further work by Ferg (1991), these two notations are referred to as the **Chen** approach (Chen, 1976)³ and **Merise** approach (Rochfeld and Tardieu, 1983)⁴. In Figure 6.2, the participation and cardinality can be considered as *look-here* and *look-across* approaches, respectively whereas in Figure 6.1 both constraints can be considered as *look-across* approaches.

Both approaches represent the same semantics in binary relationships but the way they are expressed is different. It is possible that the use of these two approaches can result in the conceptual schema being misunderstood. In the case of *n*-ary ($n \geq 3$) relationship types, the definitions of cardinality and participation constraints are more complex, not well understood and have a slightly different

³The Chen approach defined the cardinality constraint as a *look-across* constraint that considers only the maximum cardinality and that for binary relationships it must be 1:1, 1:N or M:N (Ferg, 1991).

⁴The Merise approach defined the cardinality constraint using a *look-here* interpretation of the participation. This cardinality notation is placed near the constrained entity while in the Chen approach, the cardinality notation is placed at the opposite end (Ferg, 1991).

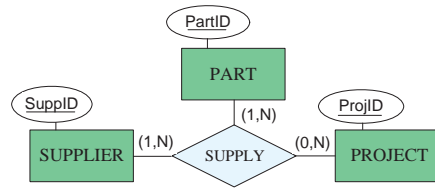


Figure 6.3: Constraints of the ternary **SUPPLY** relationship.

meaning to that for binary relationships. The use of both the **Merise** and the **Chen** approaches for n -ary relationship types to simultaneously represent the semantic constraints is effectively unworkable as such constraints cannot be replaced or expressed by each other (Ferg, 1991; Hartmann, 2003; McAllister, 1998). For this reason, most ER modelling textbooks suggest to use only one of the possible approaches.

Generally, when the **Chen** approach is used, the constraint (the number of possible instances) of an entity type in an n -ary relationship is determined by fixing values/instances of the other $(n-1)$ components. For example, the constraint for a ternary relationship represents the number of entity instances/occurrences of a particular entity in a relationship where the values for the two other entities are fixed. Consider the ternary **supply** relationship between **supplier**, **part** and **project** as shown in Figure 6.3. To determine the constraint of the **project** entity, the two values of **supplier** and **part** are first fixed, then the number of possible occurrences (minimum, maximum) of the **project** entity are counted. The same procedure is applied to **supplier** (with pairs of **project** and **part** instances) and **part** (with pairs of **supplier** and **project** instances). In Figure 6.3, it can be seen that the participation constraint of **project** is zero, that is, there are instances of the **supply** relationship that associate instances of **supplier** and **part** entities, but with no instance of **project**. This leads to difficulties in identifying the relationship instances where information is indeterminate.

The polymorphic relationships presented later in this chapter show another option to handle the situation where the specific information for any relationship is indeterminate.

6.3 Motivation

The current modelling paradigms, including the ER/EER model, do not adequately address a number of difficult conceptual modelling issues. Consider the

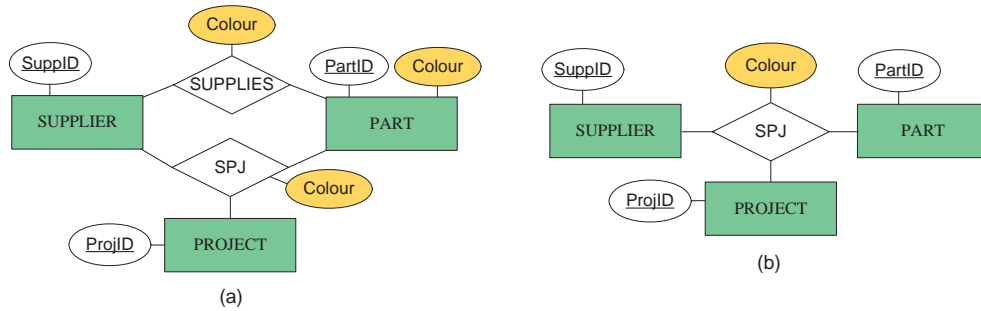


Figure 6.4: Two design diagrams for Example 1 (a) duplication of the Colour attribute and (b) an SPJ relationship requiring *dummy* instances.

following example:

[Example 1.] *The ABC Company manufactures three types of widget — **wid**ayes, which are always blue (irrespective of the manufacturer), **wid**bees, which the ABC Company paints blue (but which other manufacturers produce in other colours), and **wid**seas that are by default black but which can be painted according to the project on which they are used. The XYZ Company manufactures **wid**ayes, green **wid**bees and has recently made a test batch of red **wid**dees that are as yet unused on any project.*

In order to capture this, two diagrams can be designed:

1. a diagram where the colour attribute is repeated at three levels, as shown in Figure 6.4(a), and
2. a diagram shown in Figure 6.4(b) where some of the semantic information is lost and which uses *dummy* projects and suppliers.

The problem here is that one or more of the identifying attributes for the Supplier-Part-Project (SPJ) relationship may be missing thus requiring either the alternative, redundant values (such as for the former case above) or key value substitution (as nulls conflict with entity integrity constraints) to overcome the deficiency (the latter case). In both of these cases, the querying and updating of the resulting database is made more complicated by the complexity of the relationship. Similar problems apply to binary relationships. Consider the following example:

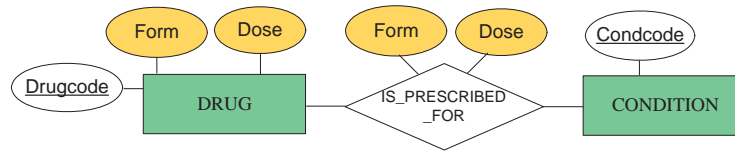


Figure 6.5: Duplication of Form and Dose attributes.

[Example 2.] *A drug has a standard form (tablet, capsule, etc.) and dosage. For a relatively small number of conditions, either or both of the form and dose may be varied.*

Once again, the Form and Dose attributes need to be repeated as shown in Figure 6.5. Note that the repetition of the attributes is not due to their taking different roles different roles but rather whether the associated condition is known. In most cases, a condition will reflect the relevant values for a standard dose and form. Thus operationally, three non-mutually-exclusive scenarios can occur:

- The standard form and dose are duplicated for most conditions.
- Those conditions taking the standard form and dose are given a value of **as standard** for form and dose.
- Those conditions taking the standard form and dose are given null values.

Unless adequately controlled, any system may potentially use all of these design solutions and thus inheriting these same problems.

Finally, consider the following example:

[Example 3.] *Details of meetings between groups of staff and students are required to be stored. Meetings can be held between two or more people.*

In this case, meetings can be modelled as shown in Figure 6.6. In particular note:

- the creation of a supertype **person** together with a relationship between two **persons** as shown in Figure 6.6(a). Apart from the additional modelling overhead, this example demonstrates the problems that may arise from the potential exponential explosion of instances as the number of **persons** participating in the meeting increases.

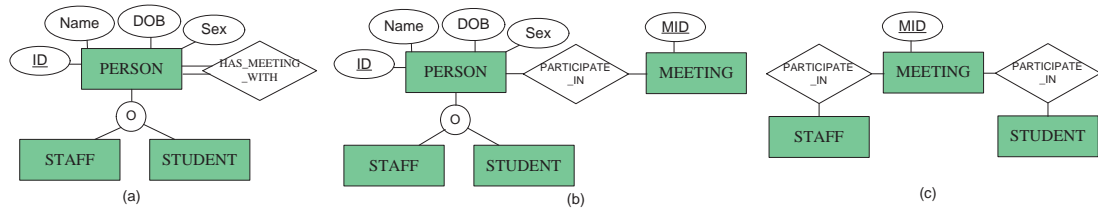


Figure 6.6: Three options for the modelling of meetings.

- the development of an entity type `meeting` requiring the use of a system-generated key. Once again, this may involve the creation of a supertype person (Figure 6.6(b)) or the creation of two separate relationships — *student participated in* and *staff member participated in* (Figure 6.6(c)).

In addition to the static modelling problems described above, Chen (2006) pointed out that the static nature of current models also makes the modelling of dynamic situations difficult. This affects the modelling of behaviours and perspectives.

6.4 Links as Overloaded Polymorphic Relationships

Object-oriented programming languages extensively use overloading⁵ to activate a method to operate differently according to the type and number of parameters passed to it. For example consider a `line` method. Using two parameters (from and to) a single default black line is produced e.g. `public Line (double x, double y)`. When three parameters are used, a line colour can be selected e.g. `public Line (double x, double y, char colour)`. Effects can vary according to the types of parameters passed, such as where one is a point, another a direction and a third a distance, e.g. `public Line (double x, double theta, double distance)`. In all these cases the same method is used, and the same outcome can be achieved (i.e. a line), but the enabling code differs. That is, the method has been ‘overloaded’.

⁵Overloading is a special case of the more general concept of **polymorphism**, from the Greek meaning ‘having many forms’ (Connolly and Begg, 2004).

6.4.1 Link Type Structure: Formal Definition

A variety of types of relationship have been discussed and different types of relationship may exist in the same diagram (Chen, 1977). This thesis suggests that a user using a polymorphic relationship, or *link*, can vary the number of instances of an entity participating in an instance of a relationship. This section provides a more formal definition of a link type as an overloaded polymorphic relationship. The structural constraints of a link type are given in Section 6.5.

The ER model is based on a strong mathematical foundation. Like the relational model it uses mathematical relations to express the relationships between entities. A relationship in the ER model is defined as an ordered tuple of entities. In other words, a cartesian product of *entities* is a *relationship*, while in the relational model a cartesian product of data *domains* is a *relation* (Chen, 2002). A relationship type R can be defined as a subset of the cartesian product $E_1 \times E_2 \times \dots \times E_n$ (Elmasri and Navathe, 2007; Chen, 2002) as follows:

$$\begin{aligned} R &= \{r_1, r_2, \dots, r_n\} \\ r_i &= [e_{i_1}, e_{i_2}, \dots, e_{i_n}] \mid e_{i_1} \in E_1, \dots, e_{i_n} \in E_n \end{aligned}$$

Similarly, a link type L is a set of associations among n entity types E_1, E_2, \dots, E_n . The following notations are used for a formal definition of a link:

$$\begin{aligned} e &: \text{entity} \\ E; e \in E &: \text{entity type (or set)} \\ l &: \text{link} \\ L; l \in L &: \text{link type (or set)} \end{aligned}$$

For links, a link type L can similarly be defined as a mathematical relation, however, this allows each e_i in L to exist zero or more times. Note that this retains the need for E_i not to be distinct in order to maintain the semantics of roles.

$$\begin{aligned} L &= \{l_1, l_2, \dots, l_n\} \\ l_i &= [(e_1)^{\kappa_1}, (e_2)^{\kappa_2}, \dots, (e_n)^{\kappa_n}] \mid (e_1)^{\kappa_1} \in E_1, (e_2)^{\kappa_2} \in E_2, \dots, (e_n)^{\kappa_n} \in E_n \end{aligned}$$

Given an instance $e \in E_i$, the set of all instances of E_i that participate in L

is defined as $L_{(E_i,e)} \subset L$. The cardinalities of E_i with respect to L are given by

$$\kappa_i = \alpha |L_{(E_i,e)}|$$

where α indicates that the term α may be repeated according to the cardinality κ specified for e_i and $|L_{(E_i,e)}|$ means the number of instances in the set $\{L_{(E_i,e)}\}$. Importantly, if $\forall i : \kappa_i = 1$ then a link degrades gracefully to a relationship.

It should also be noted that each instance of E_i can participate in L zero or more times, which means $\{e | L_{(E_i,e)} = \emptyset\}$ or $\{e | L_{(E_i,e)} \neq \emptyset\}$, respectively. However, a missing link instance still retains the degree of a relationship. Note also that where $\kappa_i > 1$ then the role played by all instances e_i must be the same (that is, a link cannot be used to substitute for an employer-employee relationship).

6.4.2 Link Type Representation

This work utilises a more restrictive form of overloading, in the form of a link type⁶. Importantly, any given link imparts the same semantic connection regardless of the type and number of participating entities. Consider the SPJ relationship from the prior example as shown in Figure 6.4(b). This can be expressed as a link type as shown in Figure 6.7(a). The SPJ link provides an association between the three entity types for two purposes. Firstly, to record that a semantic association exists, and secondly to provide a value for Colour in the context of that association. For the SPJ link, the semantics of *supplier provides part for project* is interpreted such that if there is no project instance then the **project** for which the **supplier** provides **part** does not matter. Similarly, if neither a **supplier** nor **project** instance is provided then SPJ is simply interpreted as *part has default colour*.

While this polymorphic link greatly simplifies the visual representation, its main advantage is in its overloading of the SPJ association. In Example 1, **widseas** supplied by the **ABC Company** are by default **black** but may be painted according to the project. In this case, an instance can be added to link only to **supplier** (**ABC Company**) and **part** (**widsea**) with a **colour** of **black**, while further instances, linked to all three entities, can provide the colour specific to an individual project. In Example 3, meetings between staff and students can consist

⁶Since both polymorphic and non-polymorphic relationships are likely to exist in any given schema, to avoid confusion between the two concepts the shorter term *link* has been adopted to refer only to the former. This also varies the symbol used in ER diagrams to be a pentagon.

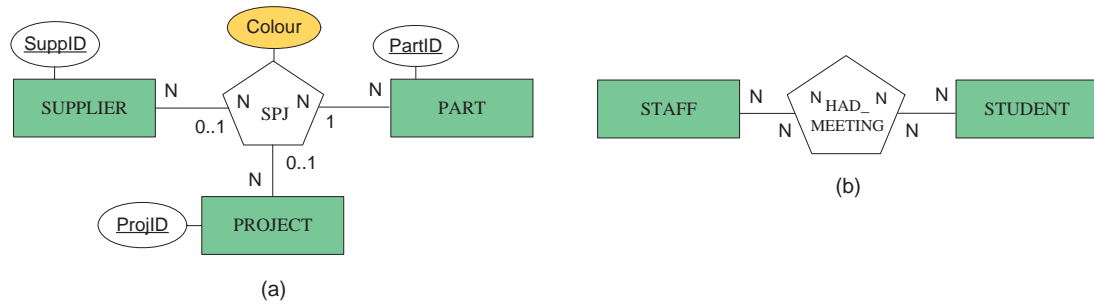


Figure 6.7: Examples 1 and 3 redrawn as links.

of a variable number of each. Figure 6.7(b) shows the `had_meeting` link for which a variable number of instances of each of `staff` and `student` that can participate. Cardinalities are discussed in more detail in Section 6.5.

In ER/EER modelling, the recording of cardinalities indicates the maximum number of instances of a relationship that a single instance of an entity can participate in. Since an instance of a relationship must be associated with *exactly one* instance of each linked entity, this also effectively implies the maximum number of times that an entity can be associated with any other entity. In many cases this is sufficient. However, there are instances, such as those discussed in Section 6.3, where there may be a maximum cardinality between any two entity types that is lower than the cardinality of a specific entity with another entity type.

In Figure 6.7(a), for example, an instance of a `supplier` may participate in many `SPJ` links and an `SPJ` link may include zero or one `supplier`, whereas an `SPJ` link must include exactly one `part`. Participation constraints can be considered as a form of cardinality and can be used for links in the same way as they are for relationships⁷. In developing links, this thesis acknowledges that some degree of semantic ambiguity may result as discussed by Wand et al. (1999). However, this thesis argues that the benefits of such an extension outweigh the additional consideration a designer may have to expend.

⁷Without the specification of a total participation constraint (TPC) the common cardinality of *one-to-many* implicitly means *zero or one-to-zero to many*. Adding a TPC essentially changes the lower limit on the number of relationships an entity can/must participate in from *zero* to *one*. Thus, the common cardinalities are effectively *zero-to-one*, *zero-to-many*, *exactly one*, *one-to-many* and *many-to-many*. Refer to the discussion in Section 6.5.

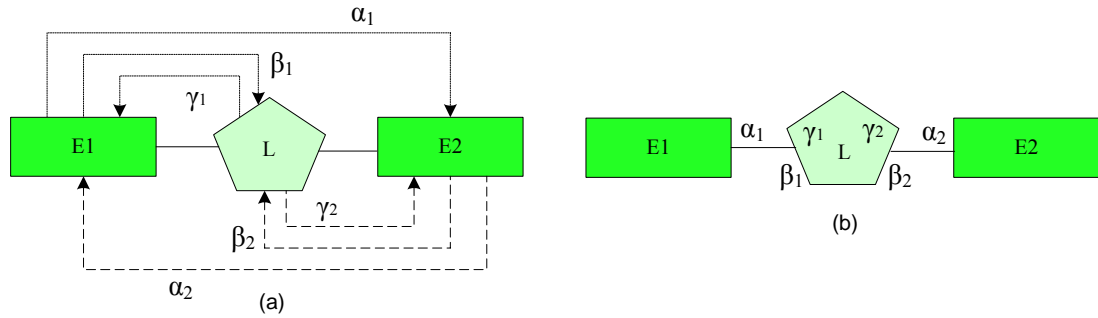


Figure 6.8: A link type (a) three types of cardinality in a binary link (b) a suggested diagrammatic format. The α cardinality position ensures backward compatibility.

6.5 Cardinality, Default Value and Precedence for Link Types

The overloading of relationships gives rise to the three additional issues: (1) an expansion of the ideas of cardinality for a link type; (2) the concept of default values to overcome the missing information in the entity integrity constraint; and (3) the concept of precedence to handle multiple conflicting values.

6.5.1 Cardinality Constraints on Link Types

Since link types can vary the number of instances of an entity that can participate in an instance of a link type, the concept of cardinality and participation must be re-examined. In general, each relationship instance r_i in R is an association of entities, where the association includes exactly one entity from each participating entity type. For links, there are three types of cardinality as shown in Figure 6.8.

The cardinalities designated with the suffix ‘1’ indicate that they either relate to the number of instances of **E1** that can match to a single instance of **L** or **E2**, or the number of instances of **L** that can be related to **E1**. This can occur for:

α (entity-to-entity) cardinalities. These correspond to the number of instances of one entity that can be linked to a given instance of another entity through (any number of instances of) **L**. This type corresponds to the cardinalities available under the current ER/EER modelling technique. Note that α cardinalities only make sense in binary relationships (which is

why they are often omitted in ternary and higher relationships). In Figure 6.7(b) it would correspond to the statement *Each staff member may only meet with each student once*.

β (**entity-to-link**) **cardinalities**. These correspond to the number of instances of an entity that can participate in a given instance of L. In Figure 6.7(b) it would correspond to the statement *Each meeting may only have one staff member present*.

γ (**link-to-entity**) **cardinalities**. These correspond to the number of instances of L that a given instance of an entity may participate in. For example, in Figure 6.7(b) it would correspond to the statement *Each staff member may only meet with one student at a time*.

Including the concept of participation as a cardinality constraint, the common cardinality ratios for binary relationship types are *zero-to-one*, *zero-to-many*, *exactly one*, *one-to-many* and *many-to-many*. These cardinalities can also be applied to binary link types.

6.5.2 Default Values for Link Types

Solutions to the problem of missing information in the design of databases has been widely addressed and the most commonly adopted technique to model missing data is null values (Codd, 1986; de Tré, de Caluwe and Prade, 2004). However, the absence of a specific value can be resolved in the following two ways:

Null Value. This signifies the absence of information that occurs when the value is unknown, inapplicable or one of the many other semantics of a null value⁸.

⁸Much of the research on the semantics of null values in relational databases dates to the 1970s and 1980s (Codd, 1970; Lacroix and Pirotte, 1976; Maier, 1983; Zaniolo, 1984; Roth, Korth and Silberschatz, 1989; Imieliński and Lipski, 1984; Vassiliou, 1979). The two definitions of nulls as given by Codd (1970) are missing and applicable, and missing and inapplicable. Zaniolo (1984) later proposed a third definition as, essentially, a lack of knowledge about the attribute's applicability, or *no information*.

Subsequently, various logical approaches were proposed to handle null values. For example, the commonly-used three value logic includes **true**, **false** (often by virtue of a value's absence — q.v. the closed-world assumption (Reiter, 1978)), and a **maybe** value that indicates that the results may be true (Yue, 1991; Codd, 1979). A four value logic has also been proposed that includes an additional truth value representing the result of evaluated expressions which have inapplicable values (Codd, 1986; Gessert, 1990). Approaches to accommodating null values in practical systems include the work of Motro (1988) who uses the ideas of conceptual closeness to *fill the vacancies* represented by a null value and Roth et al. (1989) who suggests the inclusion of nulls in NF² databases. Null values have also been studied in relation to schema evolution and integration (Kim and Seo, 1991; Roddick, 1995) and distributed databases (Chan and Roddick, 2006).

This value can be applied to non-key attributes only.

Default Value. This signifies the existence of information when a specific attribute value is not necessary for the relationship being described. Thus, the use of a default value in a key attribute does not imply a null value but rather an *if not otherwise specified* value.

The concept of default values to provide an alternative approach in some situations has been discussed (Date and Darwen, 1992). Default values have been suggested to store properties related to the existence of data such as *exists but not at the moment*, *exists but not known*, *exists but under change*, and usually only one default value is allowed (Thalheim, 2000).

To provide more clarity, the marker δ is used to indicate that the values provided hold *unless more specific information is available*. It also indicates a unique attribute value if it is used in a key attribute. Thus, the closed world assumption (Reiter, 1978) continues to hold and no recourse to 3-valued or 4-valued logic is required. Note also that the use of a default value does not violate the idea of primary key uniqueness — at most one tuple can have a specified key value regardless of whether it contains default values or not.

One the advantages provided by polymorphism is the ability to naturally provide for default values. In Example 1 (Page 167), the default colour for **widseas** can be specified as a binary link to **part** and **supplier** while a **widsea** coloured **green** by the **ABC Company** for the **Foreshore** project can be specified as a ternary link to **part**, **supplier** and **project**. Using the δ notation, potential combinations from this scenario can lead to the following information:

[Example 4.] *Sample instances in the SPJ link type*

<u>Partid</u>	<u>Suppid</u>	<u>Projid</u>	<u>Colour</u>
Widaye	δ	δ	blue
Widaye	ABC Company	δ	δ
Widaye	XYZ Company	δ	δ
Widbee	ABC Company	δ	blue
Widbee	δ	Regius	white
Widbee	XYZ Company	δ	green
Widsea	ABC Company	Foreshore	green
Widsea	ABC Company	δ	black
Widdee	XYZ Company	δ	red

Any query on this information requesting the colour of **Widseas** on the Civic project will return the default value of **black**.

6.5.3 Precedence

Given the nature of default values and the need to provide a degree of certainty over what values are taken, a system of precedence is needed to ascertain the values taken in specific circumstances. This will also help to clarify what happens when multiple conflicting values are specified. Such multiple conflicting values occurs, for example, in Example 4 when enquiring about the colour of **Widbees** supplied by the **ABC Company** for the **Regius** project.

This thesis thus adopts the following rules when determining the value of an attribute.

1. Where multiple instances fulfil a relational query, all are returned. For example,

```
SELECT  colour FROM SPJ
        WHERE  partid = "Widbee";
```

returns both **blue**, **white** and **green**.

2. Tuples comprised entirely of default value markers are not returned. For example,

```
SELECT  colour FROM SPJ
        WHERE  partid = "Widaye";
```

returns **blue**.

3. Where a query does not return a result, then the results of the highest degree, more general query are returned. For example,

```
SELECT  colour FROM SPJ
        WHERE  partid = "Widsea"
        AND    suppid = "ABC Company"
        AND    projid = "Civic";
```

returns **black**.

To provide clarification on the results, the following precedence rules can be applied:

- (a) Determine the query degree. The *query degree* refers to the number of entity key values specified in a query Q over a link type L. For example, the query above has a *query degree* of 3.
- (b) Apply the query degree to achieve the results. The results are the union of queries when query terms are omitted such that the *query degree* is successively reduced until the result is non-null or the *query degree* = 1. For example, the query above is rewritten as:

```

SELECT   colour FROM SPJ
        WHERE   partid = "Widsea"
        AND     suppid = "ABC Company"

UNION

SELECT   colour FROM SPJ
        WHERE   partid = "Widsea"
        AND     projid = "Civic"

UNION

SELECT   colour FROM SPJ
        AND     suppid = "ABC Company"
        AND     projid = "Civic";

```

If this query does not return any results then the *query degree* is reduced further until a result is achieved.

6.6 Summary

A number of areas are presented for further investigation. In particular, the concept of links are presented in this thesis as a modelling tool only. To make these operational, a method of mapping them to the relational model is required in the same way that basic relationships are mapped. Generally, the links can be handled in the same way as relationships but not always.

It is possible that further extensions may be more focused on query languages, such as SQL, that may be particularly useful in supporting link processing. A minor extension has been discussed in Subsection 6.5.3.

As discussed earlier in Subsection 6.4.1, links can degrade gracefully to relationships. It is possible that this notion could be applied to the situation where schema is ill-formed, but where there is a need to store information rapidly such as the case for rapidly evolving events. It may be possible that links can provide solutions to some of the problems in active conceptual modelling as identified by Chen (2006).

Chapter 7

Data Modelling in Rapidly Changing Complex Environments

This chapter proposes a rapid conceptual modelling framework that builds upon the underlying technology components such as a knowledge base, hypotheses and analysis routines, using the previously presented concepts of mesodata, ontologies (Chapters 4 and 5) as a general ontology storage component, and the previously presented concept of links as polymorphic relationships (Chapter 6) as part of a common conceptual schema component. These components are used to form an architecture for a novel conceptual modelling approach — *Low Instance-to-Entity Ratio* (LItER) modelling. LItER modelling introduces an overarching architecture that incorporates hypothesis, knowledge base and ontology support together with a common conceptual schema. This allows data to be stored immediately and for a more refined conceptual schema to be developed later. The LItER model attempts to provide a platform and modelling technique to handle rapidly changing phenomena.

The structure of this chapter is as follows. Section 7.1 provides an introduction to this topic. In Section 7.2, some outstanding conceptual modelling issues relating to systems and data issues are presented. Section 7.3 discusses the late binding of the conceptual model. Section 7.4 proposes the LItER model with an examination of its schema, architecture and characteristics. Section 7.5 presents a summary of the model.

7.1 Introduction

There is a growing recognition that data has to be integrated rapidly where investigation of the data is urgent. This demands data modelling techniques that facilitate the rapid exploration of data which accommodates automated data analysis techniques. As discussed by Roddick et al. (2008), a number of issues with current conceptual modelling methodologies that affect the ability of information systems to deal with rapid data acquisition and assimilation include:

- The storage of data and the retrieval of information must take priority over the full definition of a schema describing that data.
- The number of instances for each entity is relatively low resulting in the data definition process taking a disproportionate amount of effort.
- The underlying structural changes are subject to information loss as a result of changes to the schema's information capacity.
- The structure of the information is only partially known or for which there are multiple, perhaps contradictory, competing hypotheses as to the underlying structure.

As discussed by Roddick, Ceglar and de Vries (2007), dealing with sudden events require systems capable of rapidly tracking and explaining the phenomenon for a number of reasons:

- to eliminate, or at least limit, any immediate damage caused by the event;
- to explain how an event occurred or was allowed to occur, including accommodating alternative hypotheses as to why the event happened;
- to assist in any subsequent investigations, including the generation of inferences regarding the people or other agents that may have been involved;
- to expose weaknesses in measures designed to prevent such events; and
- to prevent such events happening again.

The impact of such scenario on information systems can vary widely but given that such events are largely unexpected, the rapid development of information

systems capable of answering questions is clearly important. Unfortunately, current conceptual modelling techniques are not capable of handling some of the vagaries of these situations (Chen, 2006).

Roddick et al. (2007) further discusses that in the case of unexpected events, the data necessary to assist in dealing with these events fall into two categories:

- *context-specific* data that could not reasonably have been foreseen, and
- *referential* or *global* data (such as ontologies, classifications, taxonomies, etc) that can be compiled in advance and used as needed.

What is therefore required is a conceptual structure within which contextual data can be loaded and waiting but which can also rapidly accommodate any other data that might be deemed necessary.

In addition, Howard (2008) discusses that the importance of a common conceptual data model is increasingly being recognised in providing a data model that spans organisations' applications and data sources. It defines data relationships that span multiple data repositories. To build such infrastructure is not simple and for this reason common conceptual data models are not widely available. Howard (2008) has suggested three main challenges to using a common conceptual model as follows:

- establishing a consensus on what defines a common conceptual model;
- creating definitions of all those mappings between applications and the common conceptual data model; and
- the management of ongoing changes; both the common conceptual model and application-specific models will evolve throughout the entire development and maintenance lifecycle to reflect changes in business requirements.

From the above discussions, the major questions to be addressed in this chapter include:

- What is the best method for accommodating multiple, diverse and complex datasets within a single model that can provide support to multiple users with different requirements?
- Is the concept of common conceptual data models viable?

- How are related technologies (such as data mining) and the concepts previously presented in this thesis (such as ontologies) able to be accommodated in a conceptual modelling framework?
- Can conceptual modelling techniques allow rapid and simultaneous storage of data and data modelling?

This chapter outlines a new modelling approach, **LtER** modelling (Roddick et al., 2008), that allows for rapid data modelling. This model accommodates both context-specific and referential data and, as the model possesses a common conceptual schema, the data can be recorded in a (temporal) database environment with all of the advantages that such a database offers, including security, auditing and decision justification.

7.2 Significant Issues in Driving Rapid Conceptual Modelling Techniques

Rapidly changing information technology systems and user needs are by far the most prominent of forces that drive the focus of conceptual modelling techniques, so they are better able to handle complex, multi-channelled data in a sometimes chaotic mix of user needs and application environments. This section outlines some outstanding issues that affect rapid conceptual modelling techniques.

7.2.1 Systems Issues

As discussed by Roddick et al. (2008), some types of information systems are not well served by using these common information system development techniques. These include:

- systems where the immediate storage of data takes priority over data organisation, with the development of a conceptual schema becoming a secondary issue. For some systems, a mechanism to collect and store data is required more rapidly than the database design phase will permit. This includes systems designed rapidly in response to an immediate need (Chen, 2006).

- systems for which the number of entity types is large in comparison with the number of instances stored. For these systems, the overhead of conceptual modelling can be high and can lead to short-cuts such as the aggregation of inappropriate entity types.
- systems that undergo substantial structural change. While schema conversion and schema's information capacity changes do not always result in a loss of information¹, systems that regularly undergo structural change often lose information. Such systems include those used for hypothesis creation such as scientific databases (Shoshani and Wong, 1985), criminology systems (Chen, Zeng, Atabakhsh, Wyzga and Schroeder, 2003) and *ad hoc* models established to track evolving phenomena.
- situations where the structure of the information is only partially known or where there are multiple, sometimes competing (although equally valid) models of the same data. While XML can handle semi-structured schema in which instances may possess varying structure, the overall schema is still largely formalised. However, this approach deals with systems where the existence of different entity types and the attributes they possess are largely unknown or where there is no agreement on the structure. Such systems include those that aim to handle empirical evidence in which the overall structure may be changed as ideas are developed and the evidence may still be in the process of being discovered. Where conflict between data and schema arises in these types of systems, no assumptions can be made as to whether the data or the schema are at fault.

All of these aspects are exhibited, to a greater or lesser extent, by any system set up to handle sudden events and/or rapidly changing systems. While the traditional forms of conceptual modelling may eventually handle these types of systems through trial and effort, the overhead and side effects of doing so are often excessively high. In practice, the conceptual modelling step (and its associated benefits provided by the use of a DBMS) are often bypassed because of this overhead, resulting in systems lacking the functionality offered by databases.

¹As discussed by Roddick and de Vries (2006), the limits for *practical* schema versioning in a database \mathcal{D} are such that $S_1 \stackrel{p}{\equiv} S_2$ iff $I'(\mathcal{D}|S_1) \rightarrow I'(\mathcal{D}|S_2)$ is bijective where $I'(\mathcal{D}|S_n)$ is the set of all instances of S_n inferable from \mathcal{D} given the constraints of S_n .

7.2.2 Data Issues

In addition to the types of system issues outlined above, Roddick et al. (2008) has identified data issues that are common in the modelling and implementation of even conventional systems. Some of these data issues are also apparent in rapid deployment systems and some are affected by ambiguity in the application world.

Consider the following motivating example of the *widgets* manufacturing system, that was previously introduced in Chapter 6, which is based around the part-subpart and the supplier-part-project structures.

*The ABC Company manufactures three types of widget — **widayes**, which are always blue (irrespective of the manufacturer), **widbees**, which the ABC Company paints blue (but which other manufacturers produce in other colours), and **widseas** that are by default black but which can be painted according to the project on which they are used. The XYZ Company manufactures **widayes**, green **widbees** and has recently made a test batch of red **widdees** that are as yet unused. **Widayes** are not only sold by themselves but are also used to make **widseas**. **Widayes** are also known as **ayewids**.*

Some of the problems illustrated in this example that may lead to further data modelling difficulties include:

- Any collections of objects must be treated in the same manner as the objects themselves, often transitively, sometimes recursively. For example, if a batch of **widayes** are found to be defective then there may also be some **widseas** that also need to be recalled. This particularly occurs where groups are referred to in place of individuals (either through metonyms, holonyms or hypernyms).
- Attribute values are often introduced into the system in ways that are not directly comparable despite conforming to the domain's type definition. For example, **widayes** may be described as *blue*, *dark blue*, *x3333cc*, *royal blue*, *PMS286* and so on. Despite being relatively common, synonyms, such as **widayes** and **ayewids** in the example, are not well accommodated. While data coercion is sometimes possible, this is not always a solution as the provenance and integrity of the original data may need to be maintained.

Recent research into the incorporation of mesodata into conceptual modelling (La-Ongsri et al., 2008) discussed in Chapter 4, the introduction

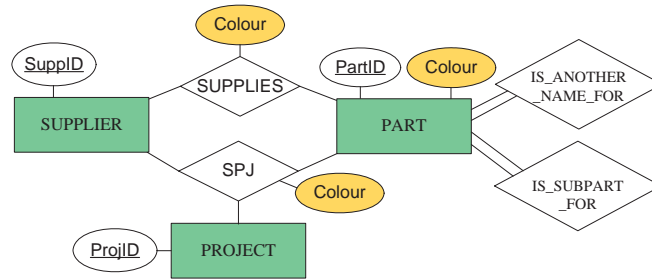


Figure 7.1: An example schema for a parts example (from Roddick et al. (2008)).

of ontologies into conceptual models for database design (Chapter 5) and earlier research into the support for mesodata (de Vries et al., 2004; de Vries and Roddick, 2007) may assist here.

- Relations formed from n -ary or binary many-to-many relationships in many conventional modelling techniques (such as ER/EER) must, for reasons of entity integrity, have a stored instance for all its associated entities. In some cases this is either not possible or not desirable and the common practice is to either deform the model to suit a small fraction of the instances or, more commonly, to create *dummy* or default codes to circumvent this constraint.

Consider the above widgets example. As *widayes* only come in blue there may be no requirement to record the supplier or the project but for *widbees* and *widseas* the project must be recorded if a specific colour is needed. In the case of red *widdees*, the location of where the colour is to be stored is not as obvious. Classical modelling would require colour to be recorded in two (or more) places in the model, for example, in a schema such as that depicted in Figure 7.1.

7.3 Late Binding the Conceptual Modelling

The latest trends in conceptual modelling are to rapidly build applications to deal with dynamic and evolving world environments. To sustain this level of development, the following two ideas can assist with the support of this approach.

7.3.1 Deferred Schema Deployment

Rapid application development relies on the rapid collection, collation and assimilation of information into a database. In many cases, the data has loose

associations and patterns and a specific schema is not required in advance. The build options are flexible — the schema can be specified at a later stage or omitted altogether. The nature of this approach suggests that schemata should be in the form of common conceptual schemata that are easy to change or that can evolve so as to be consistent with the data.

Based on this interpretation, Stonebraker and Hellerstein (2005) infer that whilst there can be some structure to the data, data instances can vary in the fields that are present and how they are represented. Specific schemata cannot be created to handle all the possible combinations of entities, relationships and occurrences. Typically data comes in a *semi-structured* form, such as free text, which can be parsed to find information of interest, and then segmented into appropriate attributes for each schema.

In contrast, when using traditional conceptual data modelling techniques, the data must be rigidly structured and must conform to a schema. The schema is first specified before any data can be stored in the database. The database is always consistent with the pre-existing schema as the DBMS rejects any records that are not consistent with the schema (Stonebraker and Hellerstein, 2005). This modelling technique can be called a *schema first* methodology.

7.3.2 Common Conceptual Schemata

Common conceptual schemata are consensually agreed structures that are used within interoperable environments to unify their disparate component databases, with varying native conceptual schemata (MacFarlane, McCann and Liddell, 1996). They define all the data relationships and meanings that exist between different data items and then map existing applications and data definitions to the common data model. As discussed by Howard (2008), a common conceptual schema effectively provides a bridge between the different meanings associated with each of the applications. These structures can thus facilitate data interoperability between applications.

As discussed by Roddick et al. (2008), the situation where the storage of data precedes the conceptual model creation requires that a different position be adopted in that a generic or common conceptual model must exist for the initial data storage in the absence of the more specialised model. However, having established a common conceptual model, specialisations to that model can be developed incrementally through the testing and imposition of constraints.

For example, consider a scenario in which a University's student and faculty data is stored in a common storage structure (ignore for the moment the details of that storage structure). In the absence of any specialised schema, reference to entities and attributes must be phrased in terms of the structures provided by the common data model.

Over time, constraints could be tested and added, providing more specialisation and eventually providing a level of structure consistent with a conventional conceptual schema. These may be tested, for example, through the discovery of induced dependencies (Roddick, Craske and Richards, 1996). For example, consider a constraint in a relationship between a doctoral student and a supervisor. An occurrence where a doctoral student has exactly one supervisor is theoretically possible, and if it is considered sensible, the constraint could be added. Labels could also be added so that conventional query languages, such as SQL, can function.

The advantages to this approach as discussed by Roddick et al. (2008) include:

- The deployment of multiple, perhaps conflicting, structures can be selectively delayed such as where the overall plan involves transitioning between structures or the staged development of hierarchies of schema.
- Multiple modelling paradigms can be used. For example, an EER model can be superimposed to provide a schema view while graph-oriented structures could be tested between data elements.
- The use of a common conceptual schema provides the ability to construct general utilities as well as facilitating schema integration.
- A common conceptual model lends itself to data mining as an *a priori* defined structure that will not mask hidden associations.

The intent of a common schema is to provide the flexibility to store all the necessary data, whilst retaining sufficient simplicity that avoids all the overheads associated with creating a full conceptual model. This chapter argues that the proposed LtER modelling method provides a common structural model capable of accommodating data derived from a variety of situations but which remains sufficiently structured to retain the semantics of the data.

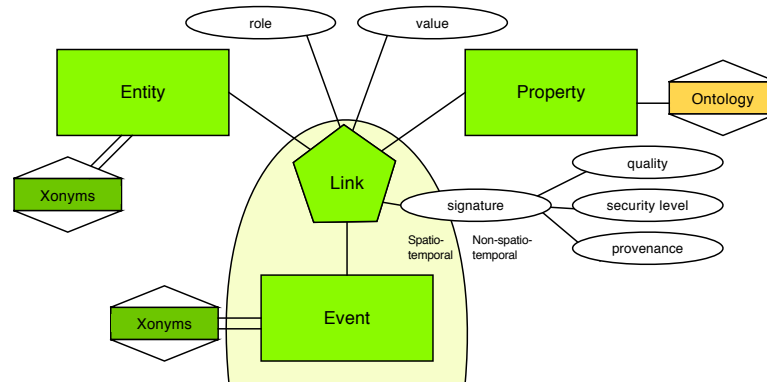


Figure 7.2: LtER schema (from Roddick et al. (2008)).

7.4 LtER Modelling

The LtER approach developed by Roddick et al. (2008) resulted from a practical industry need to generate systems rapidly where the full nature of the system is not known in advance but where some initial preparation (for example, the collection of mesodata and/or ontologies) can be undertaken. Such areas include national security, natural disaster and large-scale incidents where the finer details are unlikely to be known in advance. The LtER approach uses a common schema, the LtER Schema, as discussed in Section 7.4.1 and shown in Figure 7.2, embedded within an overarching architecture as discussed in Section 7.4.2 and shown in Figure 7.3.

7.4.1 The LtER Schema

The schema consists of three primary meta-entity types, Entities, Properties and Events, together with a ternary Link (a form of polymorphic (overloaded) relationship type). Events and Links are temporally referenced. Specifically, the model consists of the following components:

Entities. These represent objects in the model. This includes not only elementary objects such as those that might be represented by strong and weak entities in an EER model, but also components and aggregations of entities. If data is obtained from multiple sources, the same entity may also be recorded more than once (linked by a synonym link). The only attribute directly recorded is the entity’s identifier (which might be user or system supplied). All other attributes are recorded through reference to a property.

Properties. These allow the description of properties that can be associated with either an entity or an event. A property may be associated with an ontology which can be used by the constraint manager/hypothesis checker or by the query language as appropriate.

Events. These allow the recording of spatio-temporal elements. Once again, the only attribute directly recorded is the event's identifier.

Links. Links allow entities, properties and events to be combined. LtER uses an overloaded form of polymorphism (as discussed in Chapter 6) that specifies that links can occur between all or any of **Entities**, **Events** or **Properties**. Additionally, more than one of each can participate in a link with the sole requirement being that a link must include at least two identifying instances. Thus, a link can allow:

- one or more **entities** to be associated with a describing **property**, optionally with a *value* of that property. Note that the value may also be provided as a simple formula that may include a variable (such as > 25 or $P(\text{Age}).E(\text{Christopher}) + 2$). This allows facts such as ... *is over 25* or ... *is 2 years older than Christopher* to be recorded².
- one or more **events** to be associated with a describing property, optionally with a *value* for that property.
- a link between an **entity** and an **event** such that an **entity's role** in the event can be recorded, and
- a relationship between two entities.

Links have a number of optional predefined attributes as follows:

Value. A value provides a qualification for the relationship being described. For example, Entity **Mary** linked to Property **Nationality** might have the value **Australian**.

Role. A role provides a qualification for the relationship being described. For example, Entity **Luke** linked to Event **Phone_Call** might have the role **Caller**.

²Specifically, this allows first order formulae over constants or variables participating in the link plus those accessible through graph traversal. For example, $P(\text{Age}).E(\text{Christopher})+2$ references the value associated with the link between the entity with entity identifier *Christopher* and the Property *Age*, while $\max(P.(Age).E(*).P(\text{InDept}[\text{Consultant}].\text{Sales}))$ returns the maximum *Age* of all entities with a link to Property *InDept* with a value of *Sales* and a role of *Consultant*.

Quality. A quality provides a measure of confidence (such as a probability) to the link.

Security level. A security level provides a mechanism for restricting access to data.

Provenance. A provenance provides a mechanism to record the owner or source of the data.

Ontologies and Xonyms. These are an integral part of the model.

Ontologies. Within the context of this model, full ontologies are defined as complex domain structures associated with properties. Such ontologies are similar to the mesodata concept presented in Chapter 4 including its use for modelling ontologies as discussed in Chapter 5 (also refer to its precursor in de Vries et al. (2004); de Vries (2006)).

Xonyms. These are a variety of common binary references that are widely used to allow common linkages between objects to be recorded more simply. For example, a **Person** can be found to *be the same as* another. A **Meeting** *is a form of* **Communication** and so on. In these cases *synonym* and *hypernym* references would be created respectively.

Other **Xonyms** include acronyms, holonyms (and meronyms), hypernyms (and hyponyms), metonyms, pseudonyms and synonyms. The alternative, merging **Entities**, would result in a loss of both information and provenance, particularly if the reason for the merge was later discredited.

Extending query languages and data mining routines to interpret the semantics of ontologies and Xonyms and building these into the model should not be a difficult task. It is possible that some systems may not require any changes for this to occur, such as in the case of association mining routines which may be given their input data with all synonyms resolved.

7.4.2 The LtER Architecture

While the LtER model is independently useful, an overarching architecture has been developed that maximises the benefits of the model (as shown in Figure 7.3). Some of the important points are discussed below.

Analysis Routines. Four sets of routine are made available to the user:

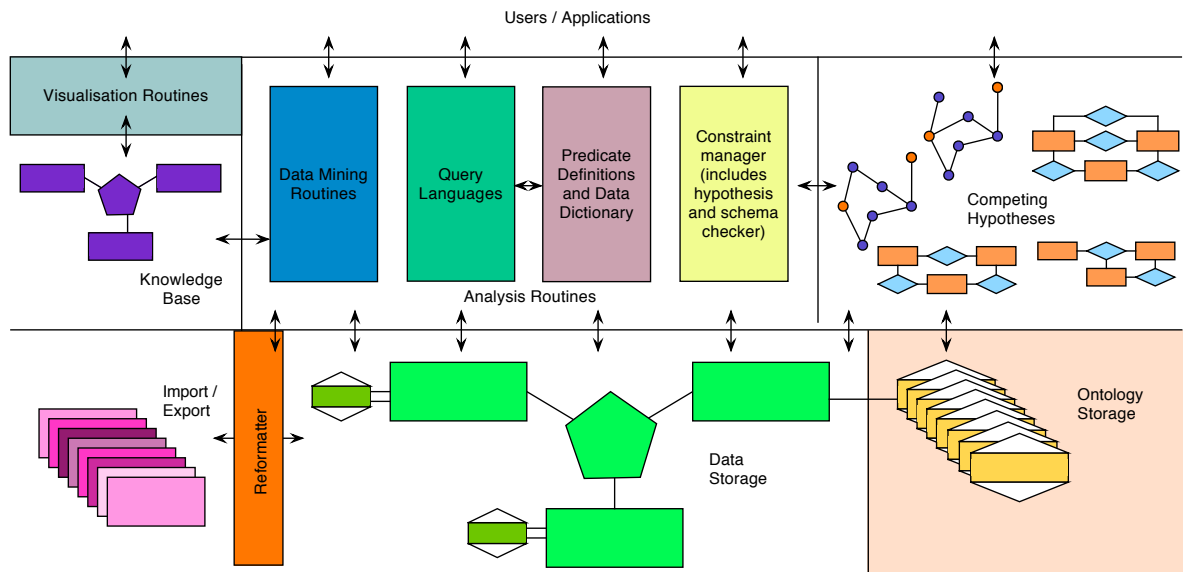


Figure 7.3: LtER architecture (from Roddick et al. (2008)).

Predicate Definition and Data Dictionary. These provide a resource to allow easy reference to data items.

Query Languages. It is possible to provide a form of SQL which resolves the terms provided by reference to the Predicate Definitions and Data Dictionary. For example, the query:

```
SELECT  EmpNAME
FROM    EMPLOYEE
WHERE   EmpAGE < 25;
```

may, depending on how the data was organised, be resolved by the following operations:

```
EMPLOYEE ::= {E(*) . P(WorkFor)}
EmpNAME  ::= P.(HasName) . EMPLOYEE
P.HasAge ::= V.(WasBorn) . E(*) - TODAY()
EmpAGE   ::= P.(HasAge) . EMPLOYEE
```

The first statement creates a set of instances of *Entity*. The second returns the value for the link to the *Property HasName*. The third creates a virtual property of *HasAge* which exists for all instances of *Entity* with a link to an *Event* of type *WasBorn*. The last returns the desired values for all instances of *EMPLOYEE*.

Constraint Manager/Hypothesis Checker. This allows the creation

of structures that are either used to constrain the data or as a putative hypotheses that can be checked against the data.

Data Mining Routines. One of the drawbacks of many data mining systems is the lack of reusability caused, in part, by changes in the manner in which data is stored. LtER accommodates data mining routines by utilising a common schema. The techniques of graph mining (Chakrabarti and Faloutsos, 2006) and association mining (Ceglar and Roddick, 2006) have been found to be particularly useful. For most purposes, these routines are fairly generic and independent of the data. For example, consider a graph mining routine that seeks to characterise modes of communication between actors. If any other graph uses the same LtER schema, then the routine can also access entities within these other graphs if they are linked to the principal entity through some event. This method of linking across different databases could, for example, be used to identify actors that are all affected by a transmission of infection.

A Knowledge Base. In most cases the knowledge base uses the same LtER architecture. Importantly, visualisation routines can operate over the knowledge base and this can take advantage of multiple runs and different forms of data mining.

A General Ontology Storage Area. This can be populated independently of the data storage providing the benefit that it can be prepared prior to the loading of data. To date, the use of ontologies has been restricted to complex structures of attribute values as discussed within the context of the mesodata concept (Chapters 4 and 5).

The LtER architecture can be realised through technology integration (e.g. data mining, knowledge based technologies, hypotheses, probabilistic reasoning and temporal auditing, etc.) and by combining related concepts (e.g. mesodata, ontologies, polymorphic relationships, etc.) and modelling techniques. Initial ideas on the use of the LtER architecture suggested that there may be utility in adopting a common meta-schema that may be able to be translated to a conventional model. This process may require certain assumptions to be applied and various aspects such as those described in Subsection 7.2.1 would need to be addressed.

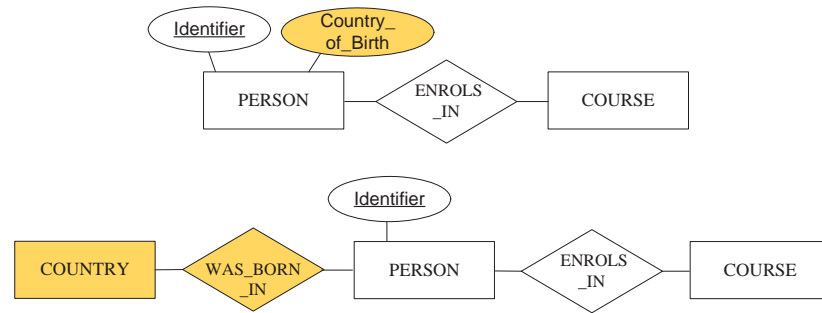


Figure 7.4: Two design choices (from Roddick et al. (2008)).

7.4.3 The LtER Characteristics

The LtER model as discussed by Roddick et al. (2008) has a number of important modelling characteristics as follows:

1. As objects in a system can play many different roles, entities are not directly associated with a specific type. Instead their types are recorded by virtue of the properties (and potentially the property-value pairs) that they can hold. Thus, objects that are owners may be identified through the property *is an owner* (cf. the category concept of Elmasri et al. (1985)). For instance, Australians can be identified through the property-value pair of *having nationality* with value *Australian*. Significantly, this allows the creation of heterogeneous sets — being *Australian* is, of course, a property that could be assigned to more than just people.
2. As the data stored in such a system is often used for hypothesis creation, the model must also allow for temporal auditing and probabilistic reasoning. Furthermore, such systems often obtain data from various sources and therefore not only must the provenance of the data be recorded but also, as far as practicable, the format and content of the data must be maintained. Thus, a data matching capability becomes an essential component of the system.
3. Relationships, and their cardinalities, are induced rather than explicitly stored. In many cases, the choice of whether a property of an object should be an attribute or a relationship to an entity type representing the concept is largely determinant on the data available (for further discussion on semantic ambiguity, refer to Wand et al. (1999)).

Consider the example in Figure 7.4 that shows how the schema can vary depending on whether an attribute or a relationship type and entity type are used. In this determination process, the LtER model mirrors the bottom-up approach used by ORM.

4. Unlike the EER Model, relationships (referred to here as links to avoid confusion) are polymorphic, and can vary a number of Entities to provide the key for a link type.

For example, the property *has colour* may be specified with a *part number* and the *project* and/or with just a *part number*. In LtER, this polymorphic use of relationships (Chapter 6) is allowed subject to the absence of any constraint forbidding it.

5. Models can be gradually refined with the successive addition of constraints. This resultant collection of constraints can be used to either validate data (either before or after DBMS commit), validate the schema or to determine contradictory information. Furthermore, constraints can take the form of hypotheses, in which ideas can be tested so as to ascertain the extent of missing information.

Note that the systems for which the LtER model is most suited do not necessarily have low volumes of data. What distinguishes this particular model is that it can handle a diverse range of information, perhaps with some data coming from large databases combined with the structure of other information being more or less specific to one or a few entities only.

7.5 Summary

The LtER model and architecture as discussed by Roddick et al. (2008) is being developed as a result of genuine industry requirements and has been applied in both defence and health industry environments. Interestingly, as has been noticed in some data mining research (Spencer, 2001), a cascade effect has been observed in that there appears to be a positive correlation between improvements to overall connectivity between data objects and the amount of data that has been added into the system.

There has been considerable discussion as to the role of Reiter's closed world assumption (Reiter, 1978) and whether it always remains valid in the context of

some systems. How these can accommodate negative information is a subject of ongoing investigation.

Despite a number of inadequacies, such as its restriction that ontologies must conform to the concepts of mesodata domains, the **LitER** model continues to generate substantial interest. Specifically, its ability to quickly bring together data from a variety of data sources and to open up new areas for subsequent investigation and research have been of great interest.

Chapter 8

From Conceptual Design to Logical Design for the Relational Data Model

This chapter focuses on how to design a relational database schema based on a proposed conceptual schema. This chapter presents the procedures to create a relational schema from a Mesodata Entity-Relationship (MDER), a Mesodata Object Role Modelling (MDORM), an Ontological Entity-Relationship (OntoER), an Ontological Object Role Modelling (OntoORM) and an Ontological Unified Modelling Language (OntoUML) schema. This discussion relates the constructs of the MDER and MDORM models presented in Chapter 4 and the constructs of the OntoER, OntoORM and OntoUML class diagram models, presented in Chapter 5.

The structure of this chapter is organised as follows. Section 8.1 provides an overview to this topic. Next, a brief introduction to the relational data model is discussed in Section 8.2. Sections 8.3 and 8.4 present an additional step of a transformation algorithm to convert the mesodata constructs in the MDER and MDORM schemata into a relational schema. Next, Sections 8.5, 8.6 and 8.7 continue to present the mapping algorithm by describing how to map ontological constructs of the OntoER, OntoORM and OntoUML class diagram schemata into a relational schema, followed by a summary in Section 8.8.

8.1 Introduction

During the process of database design, the schema may undergo transformation from one model to another. For example, the schema of a database application may be initially described using the ER model in the form of the ER diagram. It may then be mapped into the relational data model which uses structured query language (SQL) to define the schema (Navathe, 1992).

Since commercial DBMSs rely upon the relational model and do not support either of the ER models or any other conceptual models, several algorithms that translate an ER schema into a relational schema have been investigated in the literature. The theory, concepts and processes used to transform a conceptual schema into a relational schema are described in most database textbooks (e.g. for the ER/EER model (Batini et al., 1992; Connolly and Begg, 2004; Elmasri and Navathe, 2007; Kim and Seo, 1991; Teorey et al., 2006), for the NIAM/ORM (Halpin and Morgan, 2008; Halpin, 2001; Nijssen and Halpin, 1989), for the HERM model (Thalheim, 2000), for the MADS (Parent et al., 2006*a*) and for the fuzzyEER model (Galindo et al., 2006)). In addition, most publications focus on the mapping from a traditional ER schema to a relational schema (Chen, 1976; Dumpala and Arora, 1981; Par e, 2002; Teorey et al., 1986) whilst few deal with the mapping from temporally extended ER schema to relational schema (refer to Theodoulidis et al. (1991*a*) and Gregersen, Mark and Jansen (1998)). These approaches mostly use common principles which are based on the same basic methodology.

With the increasing use of ontologies in various applications and the need to provide tools to support ontologies, there is now a significant and urgent focus on improving the query reformulation task in DBMSs. To achieve this, the mapping of information between the ontology and the underlying database must exist (Necip and Freytag, 2005). To assist with this, this chapter also provides a mapping of ontological constructs discussed in Chapter 5.

It is unlikely that a complete mapping of all constructs and constraints in the source schema can be represented directly in terms of structures and constraints of the chosen logical model and thus any information-reducing transformation needs to be manipulated and realised through application programs.

Typically a high-level (conceptual) schema must be mapped onto a logical schema in order for the database to be populated and queried. As the relational data model is simple and easy to use and SQL has become the dominant relational

language used in the commercial environment, this thesis thus focuses on mapping to a relational schema using SQL.

8.2 The Relational Data Model

Relational databases have been successfully used for several decades for storing information in many application domains. Nowadays, the relational data model is the most often used approach for storing persistent information and is likely to remain so in the foreseeable future (Malinowski and Zimanyi, 2008).

The relational data model was first proposed by Codd (1970) as a mathematical basis for analysis and modelling of data. The model provides: (a) data independence by elevating the model higher, away from the physical implementation details; (b) a formal methodology for addressing redundancy; (c) efficacy of database structures; and (d) greater powers in terms of set-at-a-time operations on the model (Navathe, 1992).

Whilst database management systems (DBMSs) have been configured for the various implemented data models (hierarchical, network, relational, object-oriented, etc.), the relational data model proposed by Codd (1970) has been the most widely used since it has a solid theoretical foundation and is based on a simple data structure, i.e. a *relation*. In the mid 1970s, the development of conceptual modelling approaches, also called semantic data models, was presented. These models are used in the preliminary phase of the database design process to analyse the real world to create a formal description of the user requirements. The result is a conceptual schema which is independent of the DBMS. Where the models are not directly supported by a commercial DBMS, they can still provide value as schema design tools. This allows schemata to be first designed using a high level conceptual model and then translated into one of the classical data models, for example, relational, for implementation.

As discussed by Navathe (1992), the relational data model was a landmark development as it provided a mathematical foundation to the discipline of data modelling based on the notions of sets and relations. Due to its simplicity of modelling, it gained a wide popularity among business application developers. A number of well-known commercial DBMS products (e.g. DB2, Oracle, MS SQL Server, Informix and Ingres) provide access to the relational model for a wide variety of users.

The diagram shows a table representing a relation. The table has 7 columns: PropertyNo, Street, Suburb, Zipcode, Type, Room, and Rent. There are 3 rows of data. Labels on the left side point to the table: 'Relation Name' points to 'PROPERTY_FOR_RENT', 'Attributes' points to the column headers, and 'Tuples' points to the data rows. A label 'Domain value assigned to attribute Rent' points to the value '300' in the first row. A vertical label 'Relation' is on the right side of the table.

PropertyNo	Street	Suburb	Zipcode	Type	Room	Rent
PB6	16 Kelvin Rd	Bedford Park	5042	House	3	300
PB12	2 Nalark St	Bellevue Height	5050	Townhouse	3	260
PK14	4 Somie Dr	Kensington	5068	Flat	2	220

Figure 8.1: An example of a relation PROPERTY_FOR_RENT.

The representation of the relational model corresponding to the components of a data model is described as follows:

Data structures. The relational data model organised data in the form of *relations*. These relations consist of tuples of information defined over set attributes. Relations, tuples and attributes are formal terminologies that are used to label what are informally known as tables, rows and columns, respectively. The attributes, in turn, are defined over a set of atomic domains of values. The specification of a domain is commonly determined from the data types associated with the data values that span the domain (Elmasri and Navathe, 2007). Typically relational implementations provide a few basic domains, such as integer, real, float, date and string. Figure 8.1 shows a sample of a data structure for a relational model for an example of a relation PROPERTY_FOR_RENT.

Integrity Constraints. There are two types of constraints that fall under the schema-based constraints category for the relational data model. The first, called the *entity integrity constraint*, ensures that no records are duplicated and that no primary key value can be NULL. It guarantees uniqueness of keys. This constraint is more formally defined and referred to as a *primary key*. A primary key attribute of a relation is depicted as an underlined attribute. The second, called the *referential integrity constraint*, ensures that whenever an attribute in one relation derives values from a key of another relation, those values must be consistent. This constraint is more formally defined and referred to as a *foreign key*. For example, in Figure 8.2, the attribute OwnerNo of PROPERTY_FOR_RENT gives the owner number to which each property belongs and can assume two potential states. Either the value in each PROPERTY_FOR_RENT will match the OwnerNo value (primary key) of some tuple in the PRIVATE_OWNER relation, or it will have a value of NULL if the property for rent does not belong to a private owner or will be assigned to a private owner later. In Figure 8.2 this referential integrity

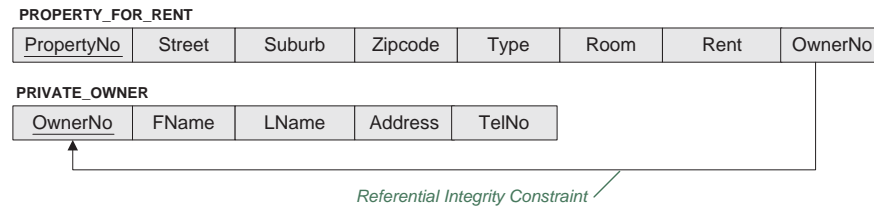


Figure 8.2: A referential integrity constraint displayed on a sample relational schema.

constraint is depicted by an arrow connecting `OwnerNo`, a foreign key of the `PROPERTY_FOR_RENT` relation, to a primary key of the `PRIVATE_OWNER` relation.

Languages. The relational model has an associated algebra which includes operations of selection, projection, join as well as the set operations of union, intersection, difference, cartesian product and so on. The set-at-a-time nature of these operations makes them very powerful since entire tables become arguments of operators.

Relational query languages such as the Structured Query Language (SQL) and Query By Example (QBE) are *declarative*, emphasising what has to be done rather than how to do it. With fourth-generation language (4GL) queries, a single statement can be used to perform operations on whole tables, or set of rows, at once and thus 4GLs such as SQL are *set-oriented* rather than record oriented (Halpin, 2001). SQL is a commonly used high level non-procedural (declarative) language used for creating, manipulating and retrieving data from RDBMSs. Due to its very common usage, it has become a *de facto* standard for the data processing industry. SQL is composed of two main sub-languages, a Data Definition Language (DDL) and a Data Manipulation Language (DML). The DDL is used to define the schema of a database. The DML is used to modify (i.e. add, update and delete) and to query data in a database. The set of SQL DDL commands defining the relational schema of Figure 8.2 is as follows:

```
CREATE TABLE private_owner (
  OwnerNo CHAR(10) NOT NULL,
  FName CHAR(20),
  LName CHAR(25),
  Address CHAR(40),
  TelNo CHAR(20),
  PRIMARY KEY (OwnerNo);
```

```

CREATE TABLE property_for_rent (
  PropertyNo CHAR(10) NOT NULL,
  Street CHAR(30),
  Suburb CHAR(30),
  Zipcode CHAR(4),
  Type CHAR(20),
  Room NUMERIC,
  PRIMARY KEY (PropertyNo),
  FOREIGN KEY (OwnerNo) REFERENCES private_owner(OwnerNo));

```

As discussed by Navathe (1992), the main argument against the relational data model is its *flatness* of structure, through which it loses the valuable information contained in the relationships or links among the data. It therefore clearly lacks the features for expressiveness and semantic richness which explains why the semantic data models are a preferred tool. This argument may be supported by the fact that the relational data model is a *logical* model targeted toward particular implementation platforms while the conceptual data model aims at expressing concepts as closely as possible to the user's perspective (Malinowski and Zimanyi, 2008).

8.3 Transformation from the MDER Schema to a Relational Schema

The transformation of the MDER schema follows the same patterns as with other ER and EER schema, but differs in that the additional mesodata entity types must also be transformed. In common with all other ER and EER components, mesodata constructs are also characterised with relations and constraints. The transformation of MDER components is an extrapolation of the ER schema transformation, with the addition that mesodata must also be considered and processed in a manner consistent with the other components. In other words, mesodata constructs must be processed by well-defined algorithms to generate a relational schema.

The seven steps of the usual mappings for ER-to-relational and the additional eighth and ninth steps of the EER-to-relational mapping algorithm are maintained (Elmasri and Navathe, 2007) with an additional tenth step to handle mesodata. This tenth step is described below and uses the MDER conceptual schema diagram for the INVENTORY database example to illustrate the mapping procedure. The INVENTORY MDER schema is shown again in Figure 8.3.

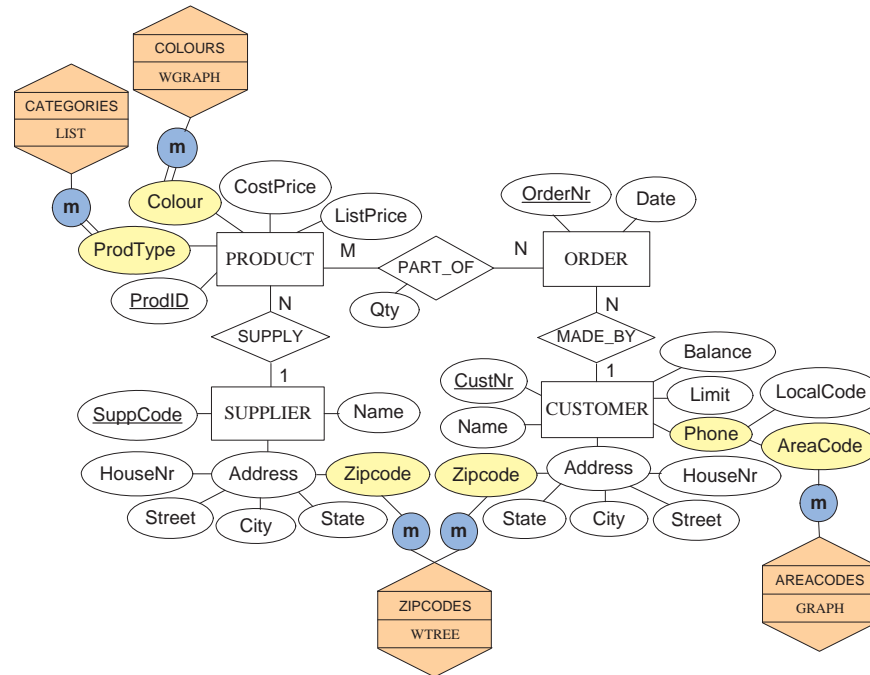


Figure 8.3: MDER schema for an example INVENTORY database.

Step 10: Mapping of Mesodata Entity Types. For the mapping of each mesodata entity type in the MDER schema, undertake the following three sub-steps. The first creates the source relational schemata, the second the domain definition and the third the attribute specification in order to reference the mesodata domain.

Substep 10A: *For each mesodata entity type M attached to attribute A , create a source relational schema R that corresponds to the mesodata type of M . The primary key of R is dependent on the mesodata type of M .*

Applying this Substep to the INVENTORY MDER schema example would create, for example, the COLGRAPH source relational schema that corresponds to the WGRAPH mesodata type which comprises the required attributes $\{\text{Value}, \text{LinkedItem}, \text{Weight}\}$ and has a composite key $(\text{Value}, \text{LinkedItem})$ as a primary key for the COLGRAPH schema.

The corresponding source relational schemata for each mesodata entity types in Figure 8.3 are shown in Figure 8.4.

Note that any end-user can both view and modify these source relational schemata. The full result of mapping the MDER schema Figure 8.3 into a relational schema is given in Appendix C.

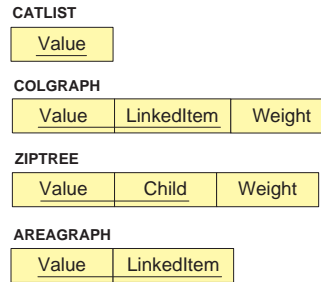


Figure 8.4: Mapping the mesodata entity types.

Substep 10B: *Create the mesodata domain definition MD over the corresponding source relational schema R. Use the mesodata type of M, the base type of A and the name of R in the AS, OF and OVER clause of the CREATE DOMAIN statement, respectively.*

Applying this Substep to the INVENTORY MDER schema example, the CATEGORIES, COLOURS, ZIPCODES and AREACODES mesodata domains are created over their corresponding source relational schemata created in Substep 10A. For example:

```

CREATE DOMAIN CATEGORIES
AS list
OF CHAR(50)
OVER catlist;

CREATE DOMAIN COLOURS
AS wgraph
OF CHAR(15)
OVER colgraph;

CREATE DOMAIN ZIPCODES
AS wtree
OF CHAR(15)
OVER ziptree;

CREATE DOMAIN AREACODES
AS graph
OF NUM(2)
OVER areagraph;

```

Substep 10C: *For each attribute A associated with a mesodata entity type M, alter the domain of A to be MD.*

Applying this Substep to the INVENTORY MDER schema example, the domains of these ProdType, Colour, AreaCode and Zipcode attributes will be modified as CATEGORIES, COLOURS, AREACODES and ZIPCODES, respectively. As a further example, the SQL for a definition of a product table that has ProdType and

Colour attributes is modified as follows:

```
CREATE TABLE product (
  ProdID CHAR(6) NOT NULL,
  ProdType CATEGORIES CLOSEDa,
  Colour COLOURS CLOSED,
  ListPrice NUMERIC,
  CostPrice NUMERIC,
  SuppCode CHAR(5),
  PRIMARY KEY (ProdId),
  FOREIGN KEY (SuppCode) REFERENCES supplier(SuppCode));
```

^aThe SQL Clause CLOSED indicates that there is a TMDP constraint on the attribute (refer to Chapter 4).

The full SQL schema definition for the example INVENTORY database is given in Appendix D.

8.4 Transformation from the MDORM Schema to a Relational Schema

One of the attractions of ORM is its ability to be easily mapped to the relational model. Since each fact type represents an elementary fact, the relations mapped from ORM are all in fifth normal form (5NF) (Leung and Nijssen, 1987, 1988; Puntheeranurak and Chittayasothorn, 2002). Consequently, there is no resultant redundancy in the schemata. Compared to the ER model, the transformation from an ER schema guarantees only a first normal form (1NF) relational database schema, and thus the relational schemata may need further normalisation. This can occur as the relationship types in the ER model are only those between the entity types, while the relationships between the attributes (functional dependencies) that can lead to redundancies, are not taken into consideration.

This thesis has extended the Optimal Normal Form (ONF) algorithm (Leung and Nijssen, 1987; Nijssen and Halpin, 1989) to incorporate the mesodata constructs. The three steps of the ONF grouping algorithm (Nijssen and Halpin, 1989) are maintained with an additional fourth step to handle mesodata. This fourth step is described below and uses the MDORM conceptual schema diagram for the INVENTORY database example to illustrate the mapping procedure. The INVENTORY MDORM schema is shown again in Figure 8.5.

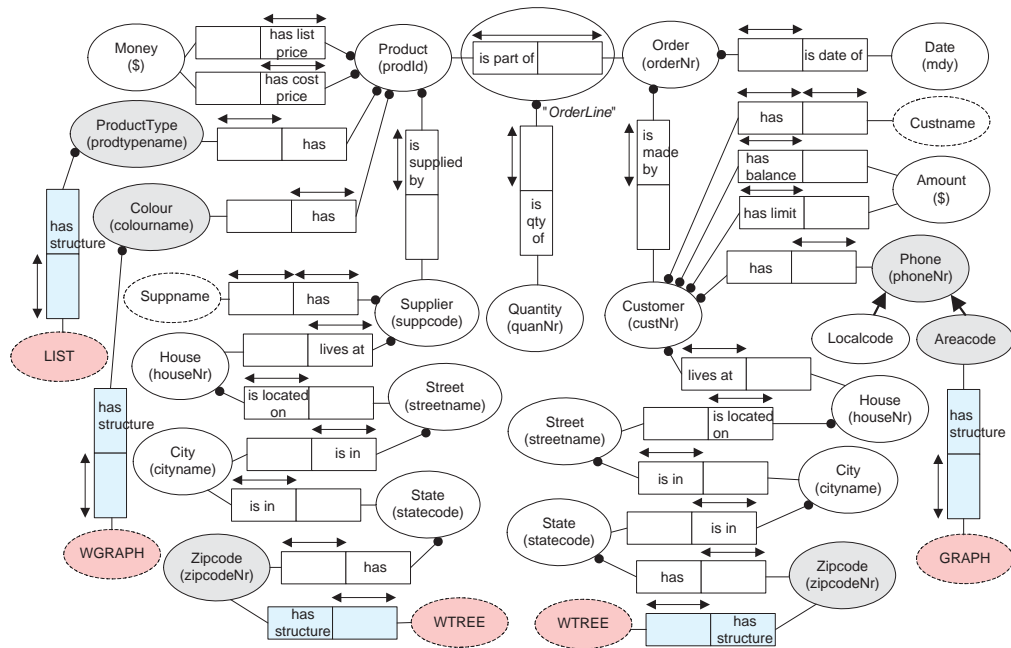


Figure 8.5: MDORM schema for an example INVENTORY database.

Step 4: Mapping of Mesodata Value Types. For each mesodata value type in the MDORM schema, there are three substeps. The first creates the source relational schemata for the specific structure of mesodata type, the second substep the mesodata domain definition and the third substep the reference to the mesodata domains.

Substep 4A: For each mesodata value type MT , create a source relational schema R that corresponds to the mesodata type of MT . The primary key of R is dependent on the mesodata type of MT .

Applying this Substep to the INVENTORY MDORM schema example, for the WGRAPH value type, the COLGRAPH source relational schema is created corresponding to the WGRAPH as COLGRAPH (Value, LinkedItem, Weight). The primary key for this schema is a composite key (Value and LinkedItem).

The source relational schemata for the mesodata value types are identical to those created with MDER. Note that any end-user can both view and modify these relational schemata.

Substep 4B: Create the mesodata domain definition MD over the corresponding source relational schema R . Use the mesodata value type MT , the base data type of the associating entity type's identifier E and the name of R in the AS, OF and OVER clause of the CREATE DOMAIN statement, respectively.

Applying this Substep to the INVENTORY MDORM schema example, the CATEGORIES, COLOURS, AREACODES and ZIPCODES mesodata domains are created in the same way as with MDER. For example, in the particular case of the WGRAPH mesodata value type, the COLOURS mesodata domain is created by using the WGRAPH in the AS clause, CHAR as the base data type of colourname, an identifier of the Colour entity type, in the OF clause and colgraph as the source relational schema (created in Substep 4A) in the OVER clause. The COLOURS mesodata domain is created as:

```
CREATE DOMAIN COLOURS
AS          wgraph
OF          CHAR(15)
OVER       colgraph;
```

Substep 4C: *For each entity type E associated with mesodata value types MT, alter the domain of an identifier of E to be MD.*

Applying this Substep to the INVENTORY MDORM schema example, the domain of these entity types' identifiers — prodtypename, colourname, zipcodeNr and phoneNr are modified to become CATEGORIES, COLOURS, ZIPCODES and AREACODES mesodata domains, respectively. For example, consider a zipcode schema resulting from the mapping of MDORM schema, the domain of zipcodeNr is altered from CHAR(4) to become the ZIPCODES mesodata domains as shown in the following SQL:

```
CREATE TABLE zipcode (
  zipcodeNr  ZIPCODES NOT NULL,
  statecode  CHAR(3),
  PRIMARY KEY (zipcodeNr);
```

Note that as the mesodata mandatory role (MMR) constraint is not imposed on the Zipcode entity type (refer to Figure 8.5). Thus, the SQL clause CLOSED does not apply to zipcodeNr (cf. Substep 10C of Section 8.3).

8.5 Transformation from the OntoER Schema to a Relational Schema

In common with all other ER and EER components, ontologies are also characterised with relations and constraints. The transformation of OntoER components

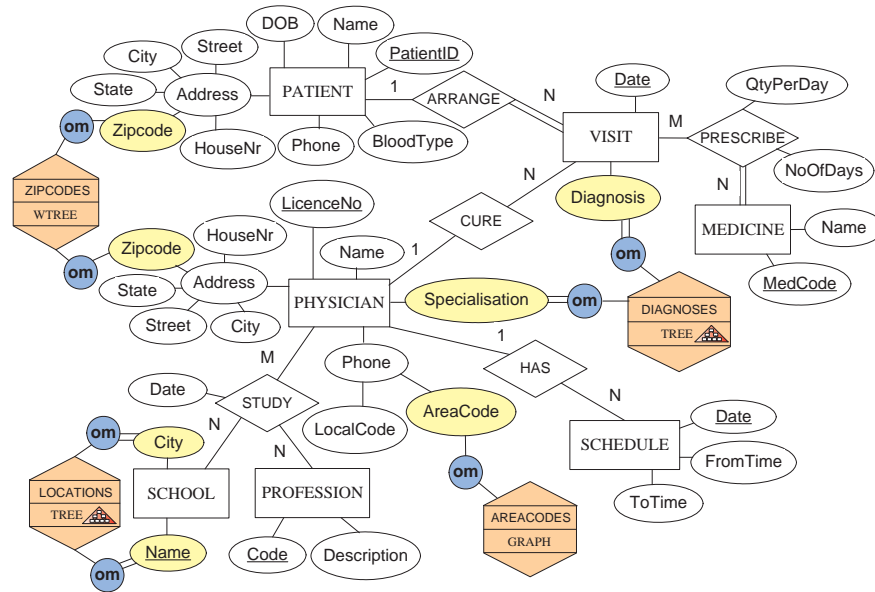


Figure 8.6: OntoER schema for an example MEDICAL database.

is an extrapolation of the ER schema transformation, with the addition that ontologies must also be considered and processed in a manner consistent with the other components. In other words, ontology components must be processed by well-defined algorithms to generate a relational schema.

The seven steps of the usual mappings for ER-to-relational and the additional eighth and ninth steps of the EER-to-relational mapping algorithm are maintained (Elmasri and Navathe, 2007) with an additional tenth step to handle ontologies. This tenth step is described below and uses the OntoER conceptual schema diagram for the MEDICAL database example to illustrate the mapping procedure. The MEDICAL OntoER is shown again in Figure 8.6.

Step 10: Mapping of Ontological Entity Types. For each ontological entity type in the OntoER schema, undertake the following four substeps. The first creates the source relational schemata, the second creates the domain definition, the third creates the attribute specification in order to reference the ontological domains, and the fourth handles the total domain participation (TDP) constraints.

Substep 10A: For each ontological entity type O attached to attribute A , create a source relational schema R that corresponds to the ontological data type of O . The primary key of R is dependent on the ontological data type of O .

Applying this Substep to the MEDICAL OntoER schema example would create

the `DIAGTREE` source relational schema corresponding to the `TREE` ontological data type containing the hierarchy of diagnosis ontology comprising the required attributes `{Term1, Term2}` representing the domain knowledge that defines the relationship between two terms. A primary key of this schema is a composite key `(Term1, Term2)`. Similarly, the `ZIPWTREE` source relational schema is created that corresponds to the `WTREE` ontological data type, formally comprising a ternary form of the required attribute `{Parent, Child, Weight}`. A primary key of this schema is a composite key `(Parent, Child)`. The corresponding source relational schema for the ontological entity types in Figure 8.6 is shown in Figure 8.7.

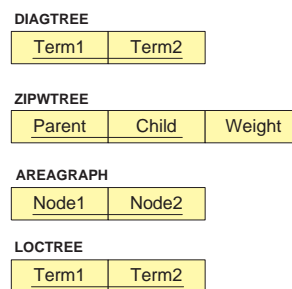


Figure 8.7: Corresponding source relational schemata for the ontological entity types.

Note that any end-user can both view and modify these source relational schemata. The full result of mapping the `OntoER` schema in Figure 8.6 is given in Appendix C.

Substep 10B: *Create the ontological domain definition `D` over the corresponding source relational schema `R`. Use the ontological data type of `O`, the base data type of `A` and the name of `R` in the `AS`, `OF` and `OVER` clause of the `CREATE DOMAIN` statement, respectively.*

Applying this Substep to the `MEDICAL OntoER` schema example, the `DIAGNOSES`, `ZIPCODES`, `LOCATIONS` and `AREACODES` ontological domains are created over their corresponding source relational schemata (created in Substep 10A) as follows:

```
CREATE DOMAIN  DIAGNOSES
AS            tree
OF           CHAR(50)
OVER        diagtree;

CREATE DOMAIN  ZIPCODES
AS            wtree
OF           CHAR(15)
OVER        zipwtree;
```

```

CREATE DOMAIN LOCATIONS
AS          tree
OF          CHAR(50)
OVER       loctree;

CREATE DOMAIN AREACODES
AS          graph
OF          CHAR(15)
OVER       areagraph;

```

Substep 10C: *For each attribute A connecting to an ontological entity type O, alter the domain of A to be the ontological domain definition D.*

Applying this Substep to the MEDICAL OntoER schema example, there are seven attributes — `Specialisation`, `Diagnosis`, `City`, `Name`, `AreaCode` and two `Zipcode` connecting to the ontological entity type. The domains of these attributes need to be modified in the same manner as the `Specialisation`, `Zipcode` and `Areacode` attributes as illustrated in the SQL data definition of `physician` table as shown below:

```

CREATE TABLE physician (
  LicenceNo CHAR(6) NOT NULL,
  Name      CHAR(40),
  Specialisation  DIAGNOSES,
  Number    CHAR(6),
  Street    CHAR(15),
  City      CHAR(15),
  State     CHAR(15),
  Zipcode   ZIPCODES,
  LocalCode CHAR(10),
  AreaCode  AREACODES,
  PRIMARY KEY (LicenceNo),
  :

```

In this case, the domains of `Specialisation`, `Zipcode` and `AreaCode` attributes are modified to be the `DIAGNOSES`, `ZIPCODES` and `AREACODES` ontological domains, respectively. The full SQL schema definition in which the domain of these seven attributes have been altered is included in Appendix D.

Substep 10D: *If attribute A of entity type E has total domain participation (TDP) into O then make the attribute A a foreign key into a corresponding source relational schema R.*

Applying this Substep to the MEDICAL OntoER schema example, `Specialisation`, `Diagnosis`, `City` and `Name` attributes possess the TDP constraint. Thus, the `Specialisation` attribute of `PHYSICIAN` entity type becomes a foreign key

into a source relational schema `DIAGTREE`. Similarly, the `Diagnosis` attribute of `VISIT` entity type becomes a foreign key into a source relational schema `DIAGTREE`, and so on. The TDP constraint for `Specialisation` can be illustrated in SQL data definition for `physician` table using `FOREIGN KEY` clause as shown below:

```
CREATE TABLE physician (
  LicenceNo CHAR(6) NOT NULL,
  Name CHAR(40),
  Specialisation DIAGNOSES,
  Number CHAR(6),
  Street CHAR(15),
  City CHAR(15),
  State CHAR(15),
  Zipcode ZIPCODES,
  LocalCode CHAR(10),
  AreaCode AREACODES,
  PRIMARY KEY (LicenceNo),
  FOREIGN KEY (Specialisation) REFERENCES diagtree(Term1),
  :
  :
```

The definition of the TDP constraint that is imposed on `Diagnosis`, `City` and `Name` attributes is applied in the same manner as for the `Specialisation` attribute. These foreign key references can be represented as the dashed line arrows linking between the related attributes as shown in the full schema mapping in Appendix C.

8.6 Transformation from the OntoORM schema to a Relational Schema

This thesis has extended the Optimal Normal Form (ONF) algorithm (Leung and Nijssen, 1987; Nijssen and Halpin, 1989) to incorporate ontological constructs. The three steps of the ONF grouping algorithm (Nijssen and Halpin, 1989) are maintained with an additional fourth step to handle ontologies. This fourth step is described below and uses the OntoORM conceptual schema diagram for the `MEDICAL` database example to illustrate the mapping procedure. The `MEDICAL` OntoORM schema is shown again in Figure 8.8.

Step 4: Mapping of Ontological Label Types. For each ontological label type in the OntoORM schema, there are four substeps. The first substep creates the source relational schemata for the selected common domain structure, the second substep creates the ontology domain definition, the third substep creates

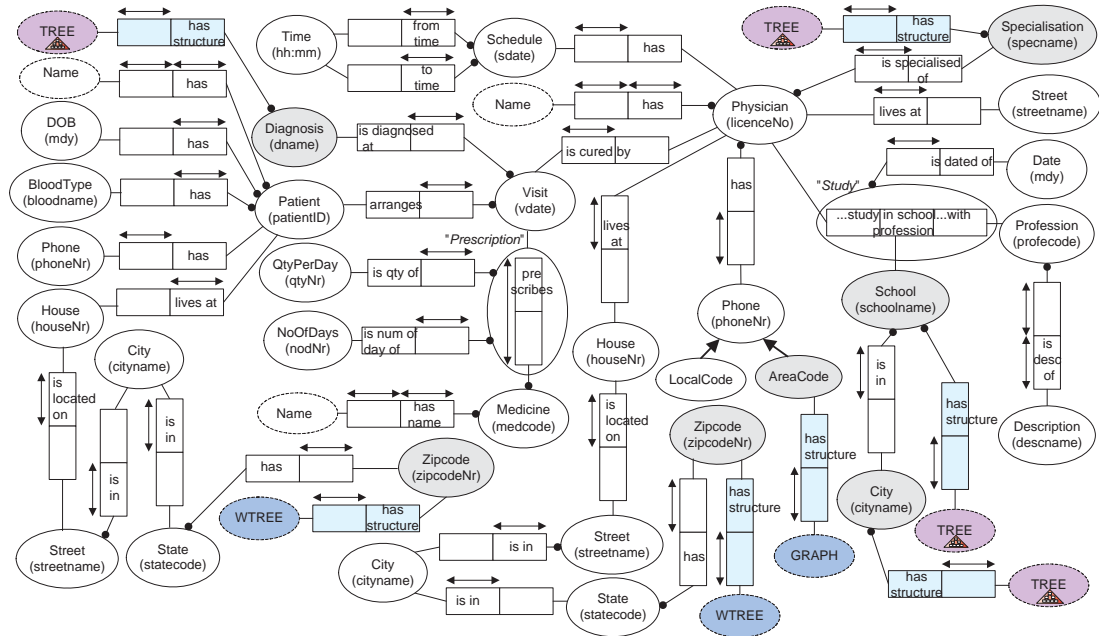


Figure 8.8: OntoORM schema for an example MEDICAL database.

the reference to the ontology domains, and the fourth substep handles the ontological mandatory role (OMR) constraints.

Substep 4A: For each ontological label type C , create a source relational schema R that corresponds to the common domain structure of C . The primary key of R is dependent on the common domain structure of C .

Applying this Substep to the MEDICAL OntoORM schema example would create the DIAGTREE source relational schema corresponding to the common domain structure TREE as DIAGTREE (Term1, Term2). The primary key for this schema is a composite key which is shown as the underlined Term1 and Term2. The corresponding source relational schemata of the ontological label types in Figure 8.8 are identical to those for OntoER.

Note that any end-user can both view and modify these source relational schemata.

Substep 4B: Create the ontological domain definition OD over the corresponding source relational schema R . Use the common domain structure of C , the base data type of the associating entity type's identifier E and the name of R in the AS, OF and OVER clause of the CREATE DOMAIN statement, respectively.

Applying this Substep to the MEDICAL OntoORM schema example, the

DIAGNOSES, ZIPCODES, AREACODES and LOCATIONS ontological domains are created over their corresponding source relational schemata (created in Substep 4A) as was the case with *OntoER*. Consider the case of the DIAGNOSES ontological domain. Its create statement uses the common domain structure *TREE* in the *AS* clause, *CHAR* as the base data type of *dname*, an identifier of the Diagnosis entity type in the *OF* clause, and *diagtree* as the source relational schema in the *OVER* clause. The DIAGNOSES ontological domain is created as:

```
CREATE DOMAIN  DIAGNOSES
              AS      tree
              OF      CHAR(50)
              OVER    diagtree;
```

The creation of the ZIPCODES, AREACODES and LOCATIONS ontological domains can be applied in the same way as with the DIAGNOSES ontological domains. Note that ontological domains with the same hierarchy of values can be reused. For example, the ontological domain DIAGNOSES which has values based on an existing ontology's class hierarchy *diagnosis* (Figure 5.1(b), Page 122), can be also used for the Specialisation entity type.

Substep 4C: *For each entity type E associated with ontological label types C, alter the domain of an identifier of E to be OD.*

Applying this Substep to the MEDICAL *OntoORM* schema example, these entity types — *Specialisation*, *Diagnosis*, *City*, *School*, *Areacode* and two *Zipcodes* are associated with ontological label types and therefore the domains of each of these entity types' identifiers will be modified to become ontological domains. For the case of creating a *visit* schema the SQL data definition alters the domain of *Dname* (an identifier of *Diagnosis* entity type) to become the DIAGNOSES ontological domain as follows:

```
CREATE TABLE  visit      (
                  Vdate    DATE NOT NULL,
                  Dname    DIAGNOSES,
                  PatientID CHAR(6),
                  LicenceNo CHAR(6),
                  PRIMARY KEY (DATE),
                  :
                  :
```

Substep 4D: *If entity type E is constrained by an ontological mandatory role (OMR) then make an identifier of E a foreign key into a corresponding source relational schema R.*

Applying this Substep to the MEDICAL OntoORM schema example, the Specialisation, Diagnosis, City and Name entity types are constrained by OMR and thus each of these entity types' identifiers becomes a foreign key into a corresponding source relational schema (created in Substep 4A). Consider the case of an identifier (dname) of Diagnosis entity type which becomes a foreign key into a DIAGTREE relational schema. The handling of OMR for Diagnosis entity type in OntoORM can be illustrated in the SQL data definition for the visit table using the FOREIGN KEY clause as shown below:

```
CREATE TABLE visit (
  Vdate DATE NOT NULL,
  Dname DIAGNOSES,
  PatientID CHAR(6),
  LicenceNo CHAR(6),
  PRIMARY KEY (DATE),
  FOREIGN KEY (PatientID) REFERENCES patient(PatientID),
  FOREIGN KEY (LicenceNo) REFERENCES physician(LicenceNo),
  FOREIGN KEY (Dname) REFERENCES diagtree(Term1));
```

This constraint can also be represented as the dashed line arrows linking to the referring attributes as shown in Figure 8.9.

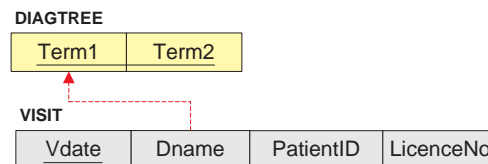


Figure 8.9: Mapping an OMR constraint in OntoORM.

8.7 Transformation from the OntoUML Schema to a Relational Schema

One of the best features of relational databases is that they are well established, supported by a very well-known, and proven, underlying mathematical theory. In addition, the structures of a relational database are simple. The transformation of the OntoUML model to a relational schema aims to retain the simplicity of the relational constructs.

The transformation of the UML class diagram into the relational schema has been discussed in Muller (1999) and Shah and Slaughter (2003). The steps of such

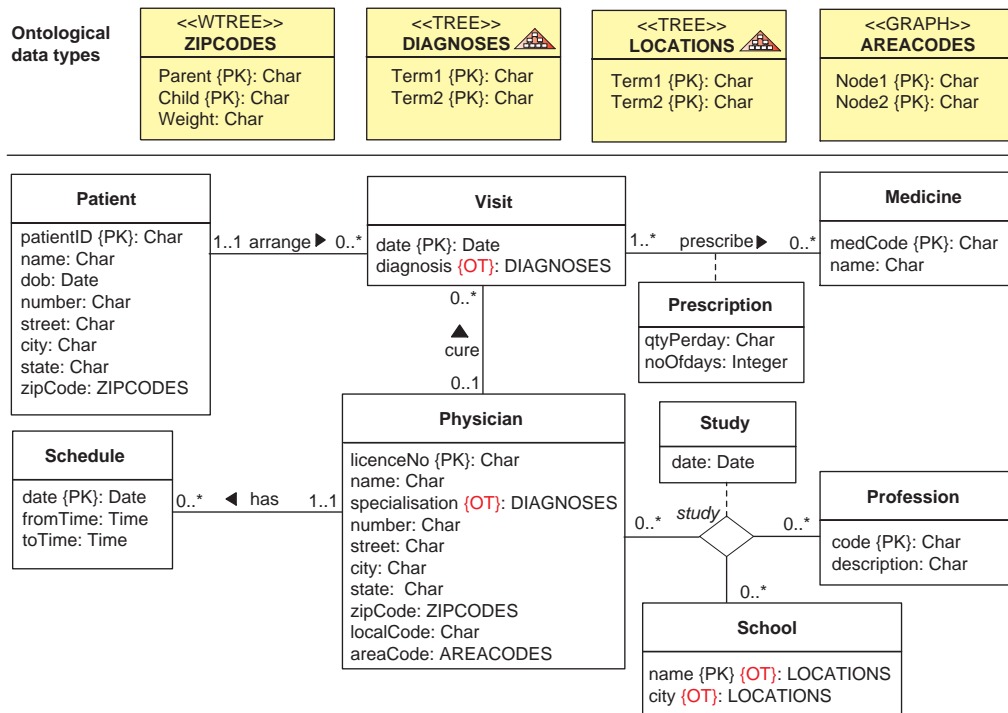


Figure 8.10: OntoUML schema for an example MEDICAL database.

mapping are not mentioned explicitly as with the ER/EER-to-relational or the ORM-to-relational mapping. However, a further process to handle the mapping of ontological constructs in OntoUML class diagrams can be included. This extra process is described below and uses the OntoUML conceptual schema diagram for the MEDICAL database example to illustrate the mapping procedure. The schema of the MEDICAL OntoUML class diagram is shown again in Figure 8.10.

Mapping of ontological class types. For each ontological class type in the OntoUML schema, undertake the following four steps. The first creates the source relational schemata, the second creates the domain definition, the third creates the attribute specification in order to reference the ontological domains, and the fourth handles the OT constraints.

Step 1: For each ontological class types O , create a source relational schema R that consists of a list of attributes of O . The attribute denoting a {PK} symbol becomes the primary key of R .

Applying this Substep to the MEDICAL OntoUML schema example, the source relational schema **DIAGTREE** (Term1, Term2) is created for the **DIAGNOSES** ontological class type. The primary key of **DIAGTREE** is a composite key (Term1,

Term2). The corresponding source relational schemata for all of the ontological class types in Figure 8.10 are the same as those with **OntoER** (refer to Figure 8.7).

Note that any end-user can both view and modify these source relational schemata.

Step 2: *Create the ontological domain definition D over the corresponding source relational schema R. Use the ontological data type of O, the base data type of the associated attributes of the usual class and the name of R in the AS, OF and OVER clause of the CREATE DOMAIN statement, respectively.*

Applying this Substep to the **MEDICAL OntoUML** schema example, the **DIAGNOSES**, **ZIPCODES**, **AREACODES** and **LOCATIONS** ontological domains are created over their corresponding source relational schemata (created in Step 1) as was the case with **OntoER**. Consider the case of the **LOCATIONS** ontological domain (refer to Figure 8.10). Its create statement uses the **tree** ontological data type in the **AS** clause, **CHAR** as the base data type of Name and/or City attributes of the usual class in the **OF** clause and **loctree** as the source relational schema in the **OVER** clause as shown below:

```
CREATE DOMAIN LOCATIONS
AS tree
OF CHAR(50)
OVER loctree;
```

Step 3: *For each attribute A of the usual class referring to the ontological class type, modify the domain of A to be D.*

Applying this Substep to the **MEDICAL OntoUML** schema example, the attributes of Specialisation, Diagnosis, City, Name, AreaCode and Zipcode of the usual classes, all refer to the ontological class type. Thus, the domains of these attributes will be modified to be the corresponding ontological domains (created in Step 2).

Consider the case of the creation of a schema of the **Physician** usual class. The SQL data definition alters the domain of Specialisation, Zipcode and AreaCode attributes to be the **DIAGNOSES**, **ZIPCODES** and **AREACODES** ontological domains, respectively as was the case for **OntoER** (Section 8.5, Substep 10C).

Step 4: *If attribute A of the usual class is constrained by {OT}, then make the attribute A a foreign key into a corresponding source relational schema R.*

Applying this Substep to the MEDICAL OntoUML schema example, the Specialisation, Diagnosis, City and Name attributes of the usual classes all possess the {OT} constraint. Therefore, these attributes become a foreign key into a corresponding source relational schema (created in Step 1). The handling of {OT} is applied in OntoUML class diagram in the same way as in OntoER. Consider the case of the Name and City attributes of the School usual class. These attributes become a foreign key into a LOCTREE relational schema. The handling of the {OT} constraint for Name and City can be illustrated in a SQL data definition for school table using FOREIGN KEY clause as shown below:

```
CREATE TABLE school (
  Name LOCATIONS NOT NULL ,
  City LOCATIONS,
  PRIMARY KEY (Name),
  FOREIGN KEY (Name) REFERENCES loctree(Term1),
  FOREIGN KEY (City) REFERENCES loctree(Term1));
```

This constraint can be represented as the dashed line arrows linking to the referenced attributes as illustrated in the full schema mapping as was the case for OntoER in Appendix C.

As many database designers and users have familiarity with the modelling techniques of the ER model, they prefer viewing database schemata as ER diagrams. In this case, an optional mapping may be desirable to first map a UML class diagram to an ER/ERR diagram (a translation of a UML class diagram to an ER diagram can be found in Ou (1998)) and then apply the transformation algorithm provided by the ER/EER model (and the OntoER model) to produce the relational schema.

8.8 Summary

The transformation process and its steps presented in this thesis follow a pragmatic approach. It facilitates a practical working system that can capture the semantics of complex attribute domains, i.e. the mesodata system, leading to the delivery of a target schema that supports implementation. This chapter continues to assert the pragmatism of the modelling techniques presented in this thesis.

A summary of the transformation process to create a definition of a complex domain of an attribute based on the mesodata or ontology concepts is given as follows.

1. *Define the source relational schema for describing the structure of mesodata types (or ontological data type/common domain structures) selected.* With reference to the example in Figure 4.6 (Page 110), the source relational schema `suburbrel` that corresponds to the structure of the `WGRAPH` mesodata type is defined by:

```
CREATE TABLE suburbrel (
  Suburb1 CHAR(25) NOT NULL,
  Suburb2 CHAR(25) NOT NULL,
  Distance NUMERIC,
  PRIMARY KEY (Suburb1, Suburb2);
```

2. *Define the mesodata domains (or ontological domains).* With reference to the example in Figure 4.6 (Page 110), the `LOCATIONS` mesodata domain is defined by:

```
CREATE DOMAIN LOCATIONS
  AS wgraph
  OF CHAR(25)
  OVER suburbrel;
```

3. *Define the attribute specification to refer to the mesodata domains (or ontological domains).* With reference to the example in Figure 4.6 (Page 110), the `Suburb` attribute that references to the `LOCATIONS` mesodata domain is defined by:

```
CREATE TABLE person (
  EmpID char(4) NOT NULL,
  Name char(40),
  Dept char(25),
  Suburb LOCATIONS,
  Salary NUMERIC),
  PRIMARY KEY (EmpID);
```

4. *Handle the integrity constraints that are imposed on the models.* The SQL Clause `CLOSED` is introduced to handle constraints in `MDER` and `MDORM` models and the `FOREIGN KEY` statements are introduced to handle constraints in the `OntoER`, `OntoORM` and `OntoUML` models. The handling of the constraints that are imposed on such models is discussed in the mapping procedures in this chapter and is summarised as follows:

- To handle the total mesodata domain participation (TMDP) constraints in the `MDER` model (or the mesodata mandatory role (MMR)

constraints in the MDORM model), the SQL Clause `CLOSED` can be used as shown below:

```
CREATE TABLE person (
  EmpID char(4) NOT NULL,
  Name char(40),
  Dept char(25),
  Suburb LOCATIONS CLOSED,
  Salary NUMERIC,
  PRIMARY KEY (EmpID);
```

- To handle the total domain participation (TDP) constraints in the OntoER model (or the ontological mandatory role (OMR) constraints in the OntoORM model or the ontological type (OT) constraints in the OntoUML model), the `FOREIGN KEY` statements can be used as shown below:

```
CREATE TABLE person (
  EmpID char(4) NOT NULL,
  Name char(40),
  Dept char(25),
  Suburb LOCATIONS,
  Salary NUMERIC,
  PRIMARY KEY (EmpID),
  FOREIGN KEY (Suburb) REFERENCES suburbrel(Suburb1));
```

Producing a logical schema from a conceptual schema essentially involves moving away from a formalism designed to represent perceptions of the real world to a different formalism designed to provide for a data representation that best supports efficient techniques for data management, in particular while retrieving the stored data (Parent et al., 2006a).

As there is substantial difference between the expressive power of conceptual and logical data models, the translation process in moving from conceptual to logical design implies a semantic loss (Malinowski and Zimanyi, 2008; Parent et al., 2006a). This often occurs for the reason that the target data model either has inadequate or non-existent concepts to match those of the source schema. The usual response is to accept a degraded level of service and to request application developers to embed code to support the full translation requirements. These application programs may use mechanisms such as triggers or stored procedures, or embed integrity constraints into the schema declarations to provide for the missing service (Parent et al., 2006a).

Chapter 9

Conclusions and Future Research

The overall purpose of this thesis is to promote better practices in conceptual modelling and to integrate more semantics from real world application knowledge into database schemata. This research is focused on exploring the potential uses of sophisticated domain semantics to extend conceptual models and to tackle some of the modelling issues of semantics that are not easily accommodated in traditional conceptual models. The thesis also presents mapping algorithms to transform the constructs of the proposed models to the constructs of the widely implemented relational model including a novel approach to conceptual modelling.

This chapter discusses conclusions about the work presented in this thesis. A summary is presented at the end of each of the preceding chapters, however in this chapter, the focus is on three topics, namely: (1) the principle research contributions of this thesis (Section 9.1); (2) discussions on future research in the subject area (Section 9.2); and (3) final conclusions (Section 9.3).

9.1 Research Contributions

The focus of this section is on summarising the main points of the research contributions through answering the following research questions that were initially posed in the introduction chapter.

9.1.1 A Survey on ER Modelling Extensions

What existing extensions to the ER model have been previously described in the body of research literature?

The diverse range of the extended ER models that have been developed and researched reflects a growing interest in the use of this model to provide a greater level of understanding and organisation within database modelling. This thesis examines all the proposed ER extensions as compiled by the author and groups each proposal according to their expressiveness of enhancing the ER model. In presenting each model, this thesis describes the features of each according to the key considerations of the conceptual data model, namely; data structures, integrity constraints and languages. A comparison of the features of each model is then presented, using the Classification of ER Modelling Extension (CERME) framework. Through this exploration of the various aspects of ER extensions, a consolidated overview of ER extensions has been compiled that will serve as a valuable reference point for future research.

9.1.2 Accommodating Mesodata into Conceptual Modelling Methodologies

How can mesodata be modelled in conceptual modelling methodologies?

This question is answered in this thesis by investigating how mesodata (de Vries et al., 2004) can be incorporated in conceptual modelling methodologies to allow more advanced semantics to be associated with the domains of an attribute. The new extended conceptual models, MDER and MDORM, proposed in this thesis support advanced semantics for attribute domains in database design techniques. The models are simple and intuitive. Querying domain level data is as easy as querying relational data.

In addition to providing richer semantics, both MDER and MDORM can provide solutions in the area of information and schema changes. This can be achieved through one of the key properties of mesodata of being able to support schema changes, in particular, attribute changes whilst preserving semantic correctness and information content.

9.1.3 Incorporating Ontology-based Semantics in Conceptual Modelling

How can ontologies be modelled in conceptual modelling methodologies?

One of the main contributions of this thesis is to show how ontologies can be integrated into conceptual modelling methodologies thus generating an extended conceptual framework that can be used for capturing the domain knowledge into conceptual data models. As proposed in Chapter 5, The combination of the concepts of *common domain structures* and an *existing ontology's class hierarchy* creates a much richer level of semantics within conceptual models.

This method of integrating ontologies has been tailored to suit three conceptual modelling techniques, namely, the ER model, ORM and UML class diagrams. By incorporating the semantic relationships of ontologies in the domain knowledge, more meaningful information about attribute domains can be explicitly represented in conceptual schemata. The aptly named *OntoER*, *OntoORM* and *OntoUML* class diagram models proposed in this thesis are capable of supporting sophisticated domain semantics that can be applied in a simple and intuitive manner. The key features of these new models are:

- support for high-level conceptual data models with expandability for domain data that have a set of operations applicable for their utilisation;
- minimal syntax changes required to support the extended modelling constructs that protects the simplicity of the models;
- simplicity of use even for more complex relationships associated with the domain knowledge; and
- utility of the desirable features of mesodata and ontologies that allows the domain of an attribute to change over time without requiring changes to the schema.

9.1.4 Polymorphic Relationships in ER Modelling

How can the relatively restricted and static modelling of relationships be modified to handle a broad spectrum of situations, or is there a more feasible approach to model overloaded relationships?

Basic relationships may not be flexible enough to solve the overloaded relationships applicable to a challenging application. This thesis presents polymorphic relationships to denote the *links* of overloading associations among entities in ER modelling. These links that represent polymorphic (overloaded) relationships retain the same semantics of each entity instance participating in a link but provides a varying flexibility on how many instances of an entity that can play a role

in a link. These intuitive rules govern the defaulting of values where there is no specific information to determine the initial state value of an entity.

The idea of polymorphic relationships in ER modelling is an adaption of the idea of overloading in object-oriented programming and contributes to a unique and significant extension of the semantics of such relationships in data modelling.

9.1.5 Rapid Conceptual Modelling

How can these presented modelling constructs be related to a (re)design of conceptual modelling approaches to facilitate the rapid exploration of data?

Rapid conceptual modelling is a new modelling approach intended to define a common schema that allows data to be either stored immediately or retained for the later refinement and development of the conceptual schema. In traditional conceptual modelling, a specified schema needs to be developed first before any data can be stored. This new rapid conceptual modelling concept differs from traditional modelling aspects in that the data can be stored immediately within a framework of a common conceptual schema even in the absence of a more specialised schema. This technique is compatible with existing conceptual modelling methodologies and expands the ability of traditional modelling to manage the modelling of data in rapidly changing complex environments.

This approach supports the rapid development of complex applications which can be realised through an integration of the work presented in this thesis (the support for mesodata and ontologies and a form of polymorphic relationship) together with other related technologies (such as data mining, knowledge based technologies, hypotheses, probabilistic reasoning, temporal auditing, and so on). A rapid conceptual modelling methodology, the LtER model (Roddick et al., 2008), can be advocated as an initial perspective on the active conceptual modelling paradigm (Chen, 2006; Chen et al., 1999, 2008) that aims towards data management of the future.

9.1.6 Data Model Mapping

How can the presented modelling constructs be mapped into the relational database schema?

Mappings play a major role in guiding and managing the overall efficiency of information systems, and is crucial for validating the conceptual model (Esculier and Friesen, 1995). The MDER, MDORM, OntoER, OntoORM and OntoUML models support direct translation into classical database models (relational models) that serve as the implementation platform, whilst preserving normal forms.

9.2 Future Research Directions

The research directions discussed below indicate potential extensions of conceptual modelling. Using the same premises as presented in this thesis, it would be relatively easy to extend this work to include more conceptual perspectives on wider or additional modelling dimensions, e.g. multidimensional modelling in data warehousing. Other suggestions predict that active conceptual modelling will be the next frontier of research.

9.2.1 Extending This Work

With increasing demand for the support of more complex applications, the capturing of more semantics is highly important. Exploring how additional semantics can be gathered and represented in conceptual models is a further possible research direction. The work presented in this thesis indicates other potential areas for future research as detailed below:

- 1. Accommodating ontologies into data warehousing methodologies.**

There are interesting possibilities for future research on the topic of accommodating ontologies (including mesodata) into data warehousing methodologies. Two aspects of this topic could be useful. The first investigative challenge is in determining how to apply ontologies to multidimensional modelling based on the *star* and *snowflake* schemata. The second challenge is in investigating a method of mapping ontological constructs of multidimensional models into the constructs of ROLAP (Relational On-Line Analytical Processing) architecture. This mapping would focus on identifying transformation algorithms to create a ROLAP schema from an ontological multidimensional schema. This would rely upon schema transformation procedures that are commonly used for mapping between different levels. This suggested research could be pursued along the lines of the semantic

expressions and a data model mapping as proposed in this thesis (refer to Chapters 4, 5 and 8).

- 2. Extending attributes' domains.** As discussed in Chapter 5, another possible direction of research is to extend the use of ontologies (including mesodata) to describe an attribute's domain. The work presented in this thesis describes an attribute's domain within the context of a single ontology. This could be extended to address the case where the domain of an attribute references multiple ontologies.

Additionally, the inclusion of different relationship types such as spatial-temporal relationships for attribute domains can be undertaken using an approach that is consistent with a generic style of modelling formalism (such as common domain structures) as discussed in Chapter 5. These directions would add further merit to the reusability opportunities of ontologies and show how other kinds of relationship extensions could be suitably represented to enrich the semantic expressiveness of conceptual data models.

- 3. Mappings.** Since most DBMSs used in industry are relational, the developed models should ideally include appropriate mappings for the particular implementation platform. Thus, the further investigation of the processes to transform between the conceptual and logical levels that can be applied to *polymorphic relationships* and the LtER model is a worthwhile area of research. The outcome should deliver a well-defined mapping algorithm to transform *links* as overloaded modelling constructs into a relational schema. In addition, the method of mapping the LtER model to the accepted conceptual modelling methods is an interesting challenge that should be considered.

- 4. Using ontologies as a solution for semantic conflict.** Another direction of research that could be useful in conceptual modelling is in using ontologies as a solution for resolving semantic conflict. Conceptual modelling should provide a sound semantic coherence. Semantic conflict in heterogeneous systems is caused by the redefinition of the particular state of an entity in such a way that it may no longer hold any of the expected range of values of the real world. This prevents information integration from accomplishing semantic coherence (Han and Qing-zhong, 2004). As a consequence, one could consider ontologies to help solve semantic problems as another challenging direction of research.

9.2.2 New Areas to Explore: Advances in Conceptual Modelling

Traditional conceptual modelling concentrates on modelling static views (referred to as a snapshot by the temporal database community) resulting in a fixed representation of the real world domain (Chen, 2006; Chen et al., 2008). The *dynamics* and *changes* of evolving scenarios seem particularly promising for defining and driving advanced conceptual modelling.

The key areas for future advances in conceptual modelling are in database technologies and modelling techniques, as discussed below.

- 1. Advanced database technologies.** Database technologies have expanded their application significantly ranging from classical applications (e.g. payroll, inventory), multimedia and temporal data (e.g. maps, satellite pictures, radar signals) to more complex applications, verging on AI (Esculier and Friesen, 1995). However, most of this database technology is based on a static data model, recording the most recent information only as a single snapshot in time. Thus, the notions of temporal and spatial features, time-varying and cause-effect relationships between events/phenomena, and fuzzy data are not fully supported by current DBMSs. Unfortunately commercial reality focuses Research and Development energies on the performance and implementation considerations of database technologies at the expense of significant pioneering research into developing new practical models. Consequently, data modellers are unable to deal with the evolving and changing world state, with the effect that rapid environmental changes are not adequately handled.

The solution lies in more knowledgeable and adaptive data management systems, capable of managing a more comprehensive description of the world (Spaccapietra et al., 2008) or new types of databases along the lines of information and software technologies that support the Internet. Advanced database technologies and DBMS can be realised through technology integration (e.g. AI, software engineering, information/knowledge management, cognitive science, neuroscience, etc.), which will stimulate the next major advance in the development of conceptual modelling (Chen et al., 2008).

In parallel, the development of new information management technologies and DBMS, which have a new meaning in the context of the Internet, also

calls for new modelling techniques (Chen et al., 2008).

- 2. Advanced modelling techniques.** With the advent of the Internet, conceptual modelling is shifting to the proposed active paradigm (Chen et al., 2008). *Active conceptual modelling* (Chen, 2006; Chen et al., 1999, 2008) is a new framework aimed at describing all aspects of a domain, its activities, and changes under different perspectives using multi-perspective knowledge and human recognition. Conventional conceptual modelling for database design is a simple case of such modelling (Chen et al., 2008).

A challenge for the next generation of conceptual modelling is the development of new methods of conceptualisation and different ways of viewing reality as a single dynamic model. This will integrate the following aspects into a theoretical framework of conceptual models (Chen et al., 2008):

- time, space and perspective dimensions (e.g. temporal and spatial entities, time-varying relationships and temporal events);
- management of continuous change and learning; and
- behaviours of evolving systems (including model evolution, patterns, interpretation and uncertainty).

This approach will assist in the understanding of relationships within changing environments, potentially raising the level of awareness of society in understanding the current world state, enabling it to be better prepared to deal with future uncertainties. This active model will have significant impacts on future research directions, not only on the frameworks and architectures of current database and knowledge technologies but also on the current interpretations and viewpoints of data modelling.

9.3 Conclusions

This thesis presents extensions to conceptual modelling methodologies to capture more advanced semantics by incorporating **mesodata** and **ontologies** in the models and introducing **polymorphic relationships**. As no comprehensive survey on extensions to the Entity-Relationship (ER) model that covers various semantic aspects exists, this thesis presents a consolidated review of the various aspects of extensions to the ER model, termed the **CERME** framework that forms the basis of common aspects and comparative criteria which can be used to categorise

and compare all various proposals and which describes the key features of each proposal. Additionally, this thesis presents a useful, pragmatic and significant contribution to conceptual modelling by introducing the procedures to transform the constructs of the proposed conceptual models into a logical model which serves to assist in the database implementation. The transformation from MDER, MDORM, OntoER, OntoORM and OntoUML schemata to a relational schema is the first step to gain practical relevance in real world database applications. Furthermore, a novel approach to conceptual modelling, the LitER model, is presented to introduce a common conceptual schema and an overarching architecture to facilitate the modelling of data in rapidly changing complex environments.

These contributions demonstrate that the theory and work presented in this thesis is of sound and useful merit and has genuine applicability to real world situations. Not only have these new semantic extensions of conceptual modelling resolved some limitations of conceptual modelling, but it is anticipated that this innovative approach, the LitER model, can assist in providing a solid foundation for new and emerging application designs in the context of the active conceptual modelling paradigm as suggested by Chen (2006) and Chen et al. (2008).

The subject matter of conceptual modelling is inherently abstract and becomes increasingly complex when providing richer semantics or more expressiveness. For this reason, the number of the proposed components have been minimised in order to simplify the concepts discussed so as to assist in the easier understanding of this topic.

Appendix A

Publications Resulting from This Thesis

The following journal and conference articles have been published as a result of work associated with this thesis.

La-Ongsri, S., Roddick, J. F. and de Vries, D. (2008), Accommodating meso-data into conceptual modelling methodologies, *Information and Software Technology* 50(5), 424-435.

This journal paper introduces the use of mesodata concepts for conceptual modelling. It further discusses (a) how to incorporate mesodata into entity-relationship and object role modelling approaches with a focus on capturing semantics of complex domain structures associated with attributes, and (b) how to map it to relational schemata. This paper is related to Chapters 4 and 8 and has been revised and updated before inclusion.

Roddick, J. F., Ceglar, A., de Vries, D. and La-Ongsri, S. (2008), Postponing schema definition: Low Instance-to-Entity Ratio (LtER) modelling, in P. P. Chen and L. Y. Wong, eds, *Active Conceptual Modeling of Learning*, Vol. 4512 of *Lecture Notes in Computer Science*, Springer, pp. 206-216.

This paper introduces a new conceptual modelling approach that would allow the data to be recorded in a database by adopting a common conceptual schema. This new methodology can be realised through the integration of, for example, a knowledge base, data mining routines and ontology storages that will allow rapid and simultaneous storage of data and data modelling. This paper is fully covered in Chapter 7 and has been revised and updated before inclusion.

Appendix B

Sample Proposals from the CERME Framework

This appendix provides a more detailed description of some of the proposals examined in the CERME framework (Chapter 3). For the following proposals, their descriptions are presented based on the key considerations of a conceptual data model, namely *data structures*, *integrity constraints* and *languages*.

Note that the following representative models are a summary of the original proposals. For elaborated descriptions of these models, this thesis refers the reader to their original sources using the citations provided.

B.1 The Enhanced Entity-Relationship (EER) Model

The EER model (Elmasri and Navathe, 2007) enhanced the ER model by incorporating the concepts of *superclass/subclass* (or generalisation/specialisation), *superclass/subclass relationships*, *type inheritance* (including various types of *constraints* on specialisation/generalisation) and *category*. The concept of category is used to model the UNION construct into the ER model.

Data Structures. The EER model includes all the modelling constructs and concepts of the ER model. In addition, it extends entity types in the ER model with the new types of subclasses, superclasses and categories as detailed below.

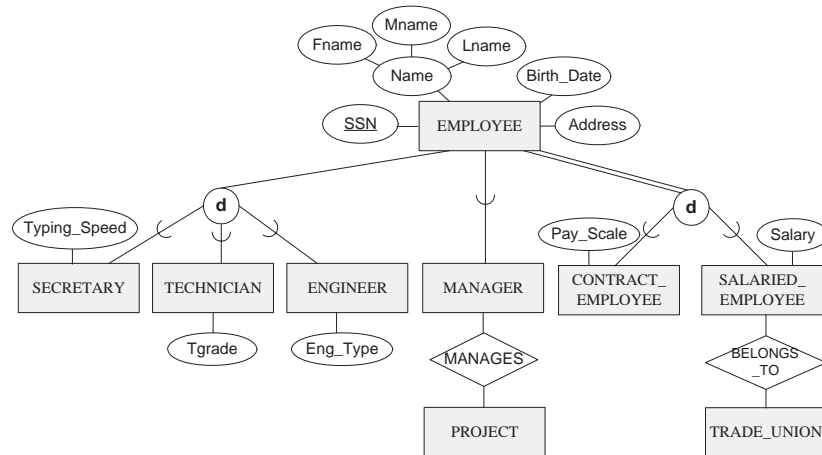


Figure B.1: Subclasses and specialisation in EER diagram (adapted from Elmasri and Navathe (2007)).

- **Subclasses.** A subclass is a specialisation of a superclass, such that each member of a subclass must be a member of the superclass. In other words, an entity in the subclass represents the same real-world entity from the superclass and can possess values for its specific attributes as well as values that are consistent with the superclass attributes. For this reason, *type inheritance* is a significant concept associated with subclasses. That is, an entity that is a member of a subclass *inherits* all the attributes of the entity as a member of the superclass as well as all the relationships in which the superclass participates. Subclasses are shown in rectangles in EER diagrams.

Specialisation is the process of defining a *set of subclasses* of an entity type; this entity type is called the *superclass* of the specialisation (Elmasri and Navathe, 2007). A specialisation through which the subclasses are connected to the superclass is represented by lines to a circle.

- **Superclasses.** A superclass is a generalisation of one or several subclasses. The term generalisation refers to the process of defining a generalised entity type from the given entity types. Superclasses are shown as rectangles in EER diagram, in a similar way to entity types.
- **Categories (or union types).** A category is a subclass of the UNION of the distinct entity types. In other words, it is a subclass with various superclasses. A category has two or more superclasses that may represent a distinct entity type while other superclass/subclass relationships always have a single superclass. A category to which the subclass is connected to the superclasses

is represented by a line to the circle with a \cup symbol together with an arc with the subset symbol connecting the circle to the category (subclass).

Integrity Constraints. The EER model includes all the constraints of the ER model. Additionally, it proposes three other constraints applicable to specialisation and category.

- **Disjointness constraints on specialisation.** A disjoint constraint specifies that the subclasses of the specialisation must be disjoint. This means that an entity can be a member of at most one of the subclasses of the specialisation. This is shown in the EER diagrams by placing a circled ‘d’, as shown in Figure B.1.

If the subclasses are not constrained to be disjoint, their sets of entities may *overlap*; that is, the same entity may be a member of more than one subclass of the specialisation. Overlapping specialisations, which are the default, are represented by a circled ‘o’.

- **Completeness constraints on specialisation.** This constraint may be total or partial. A *total specialisation* constraint specifies that *every* entity in the superclass must be a member of at least one subclass in the specialisation. This is shown in the EER diagrams by using a double line to connect the superclass to the constraint circle. Consider the example in Figure B.1. If every `EMPLOYEE` must be either a `CONTRACT_EMPLOYEE` or a `SALARIED_EMPLOYEE`, then the specialisation $\{\text{CONTRACT_EMPLOYEE}, \text{SALARIED_EMPLOYEE}\}$ represents a total participation. A single line is used to display a *partial specialisation*, which precludes an entity from belonging to any subclass. For example, if some `EMPLOYEE` entities do not belong to any of the subclasses $\{\text{SECRETARY}, \text{ENGINEER}, \text{TECHNICIAN}\}$ (as shown in Figure B.1), then that specialisation is partial.
- **Completeness constraints in category.** A category can be *total* or *partial*. A category is total if it holds the union of all entities in its superclasses, whereas if a category is partial, it holds only a subset of the union. A total category is represented by a double line connecting the category and the constraint circle, whereas partial categories are indicated by a single line.

Languages. EER provides the mapping of EER model constructs to relational constructs by extending the ER-to-relational mapping algorithm. To achieve this, EER uses SQL rather than introducing its own languages.

In summary, the EER model presents the concept of a subclass and its superclass including the rules related to attribute/relationship inheritance. A class/subclass relationship proposed in EER is often called an *is-a* relationship providing generalisation/specialisation. The constraints *total/partial* and *disjoint/overlapping* can be applied to specialisation, generalisation and category.

B.2 The Time Extended EER (TIMEER) Model

The TIMEER model (Gregersen and Jensen, 1998, 2004) extends the EER model as defined by Elmasri and Navathe (1994) to include built-in temporal support for entities, relationships, superclass/subclass and attributes. The model supports four distinct types of temporal aspects, namely *valid time*, *lifespan*, *transaction time* and *user-defined time*.

As discussed by Gregersen and Jensen (1998, 2004), new temporal constructs are presented to describe the fundamental features of the TIMEER model. The annotations added to the modelling constructs are used to indicate which temporal aspects are to be captured. These are LS indicating lifespan, VT indicating valid time, TT indicating transaction time, LT indicating lifespan and transaction time and BT indicating valid and transaction time.

Data Structures. The TIMEER extends the basic constructs in the EER model by providing support for (a) lifespans and transaction time for both regular and weak entity types, (b) valid time and transaction time for relationship types, (c) valid time and transaction time for all attribute types, and (d) time inheritance. These are described as follows:

- **Entity types supporting lifespans and transaction time.** If the lifespan or the transaction time of an entity type is to be captured, this is indicated by placing a LS (LifeSpan) or a TT (Transaction Time) in the upper right corner of the entity rectangle, respectively. If both lifespan and the transaction time are captured, a LT (Lifespan and Transaction time) is placed in the rectangle corner instead. Entity types that capture at least one temporal aspect are termed temporal entity types; otherwise, they are termed non-temporal.

Temporal aspects can be applied to both regular and weak entity types. For example, if it is required to capture both the life span and transaction

time of employee entity type, this can be modelled in the TIMEER model as shown in Figure B.2.



Figure B.2: The temporal entity type employee (from Gregersen and Jensen (1998)).

- Relationship types supporting valid time and transaction time. If a temporal aspect is captured for a relationship type, the indication is placed in the lower corner of its diamond symbol (such as VT appearing in the relationship type WORKS_FOR as shown in Figure B.3).

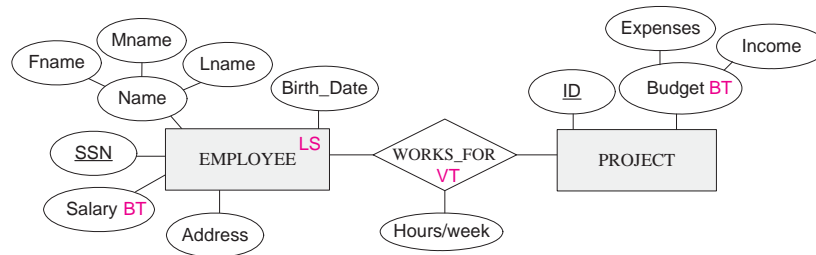


Figure B.3: The temporal relationship type WORKS_FOR (from Gregersen and Jensen (1998)).

- Attributes supporting valid time and transaction time. If a valid time of an attribute is captured, a VT is placed to the right in the attribute symbol oval. If a transaction time is captured, a TT is placed instead. If both the valid time and the transaction time is captured, a BT (BiTemporal) notation is used. If no temporal aspects of an attribute are captured, the attribute is called non-temporal.

For example, if it is required to capture the valid time and the transaction time of the Salary attribute of an employee, this can be modelled in the TIMEER model as shown in Figure B.4.

- Subclasses inherit the temporal aspects of their superclasses. This model provides support for specifying superclass/subclass relationships by extending the syntax of the EER model to include temporal attributes in the subclasses. Consider the example where lifespans for employee entities need to be captured and where secretary is defined as a subclass of the employee entity type, which also requires the lifespan to be recorded. Lifespan does not

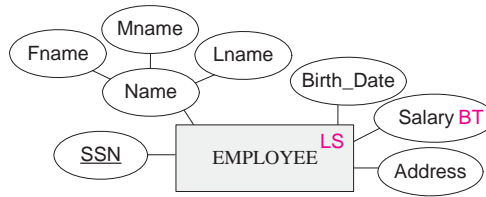


Figure B.4: The temporal (single-valued) attribute Salary (from Gregersen and Jensen (1998)).

need to be separately defined for **secretary** as both the lifespan and transaction time for **secretary** are already established as a consequence of this rule that subclasses inherit all properties, and thereby also the temporal support, of their superclasses. According to the rule of inheritance, new properties can be added, but it is not possible to delete or modify inherited properties.

Integrity Constraints. Two additional participation constraints (snapshot and lifespan) are proposed in the TIMEER model.

- **Snapshot participation constraints.** The snapshot participation constraints are a fundamental feature of the model and describe the constraint of the participation of the entities of each isolated point in time. The snapshot participation of an entity type E with respect to a relationship type R is represented by placing min and max values in parentheses by the line connecting entity type E with relationship type R as shown in Figure B.5. If $min = 0$ then the participation of the entities of E is optional. If $min \geq 1$ then the participation is total (mandatory). If $max = 1$, this means that the entities of E can not participate in more than one relationship at a time, whereas a $max = n$, with $n > 1$ means that E entities can participate in n relationships at a time.

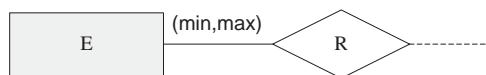


Figure B.5: Representation of snapshot participation constraints in TIMEER (from Gregersen and Jensen (1998)).

- **Lifespan participation constraints.** Lifespan participation constraints are an advanced feature of the model that describe the participation of an entity in a relationship over the entire existence time of the study. For example, it can be used to state that an employee can be assigned to any

number of projects, but only one project at a time, and at least one over the entire employment. While the snapshot participation constraint ensures that an employee participates in exactly one relationship at any point in time, the lifespan participation constraint expresses participation constraints throughout the existence time of the entities, namely over the entire employment period.

The life span participation constraint of an entity type E with respect to a relationship type R is represented by placing min and max values in square brackets by the line connecting entity type E with relationship type R as shown in Figure B.6. The lifespan participation constraint specified for the participation of an entity type with respect to a non-temporal relationship type must be the same as a specified snapshot participation constraint, and for this reason they can be omitted from the diagrams.

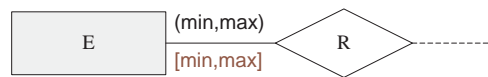


Figure B.6: Representation of lifespan participation constraints in TIMEER (from Gregersen and Jensen (1998)).

The use of both participation constraints as shown in Figure B.7, states that any employee must be assigned to at most one project at a time, but must be assigned to at least one project during the employment period. Refer to Gregersen and Jensen (2004) for a detailed discussion of combinations of snapshot and lifespan participation constraints.

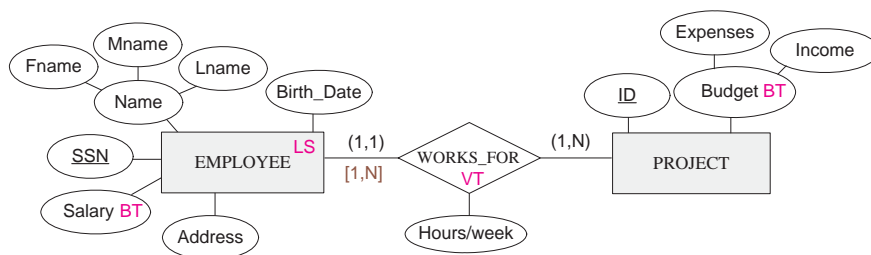


Figure B.7: The temporal relationship type WORKS_FOR with participation constraints (from Gregersen and Jensen (1998)).

Languages. The TIMEER model does not provide a query language for this temporal ER database. An alternative proposition is to convert the temporal constructs of the temporal ER model into conventional ER constructs

Table B.1: Assigning temporal aspects to ER constructs (from Tryfona and Jensen (1999)).

	Entity types	Attributes	Relationship types
Existence time	Yes	No	No
Valid time	No	Yes	Yes
Transaction time	Yes	Yes	Yes

(Zimanyi et al., 1997) and then reuse the traditional ER/EER-to-relational mapping. This solution may be attractive for minor extensions to the ER model. Another solution is to create an algorithm that can map temporal ER diagrams directly to relational schemata.

In summary, this approach introduces new temporal constructs that provide implicit temporal support and includes snapshot reducible attribute types (i.e. providing temporal single valued, temporal multi-valued, temporal composite and temporal derived attribute types) as well as snapshot reducible participation constraints. The model also supports both valid time and transaction time including lifespans.

B.3 The Spatio-Temporal ER (STER) Model

The STER model (Tryfona and Jensen, 1999) extends the original ER model to capture spatio-temporal aspects of information. This extension accommodates the combined time and space information into each basic ER construct. These new constructs are described below:

Data Structures. All basic constructs of the ER model can have spatial and/or temporal extent; however, not all types of time (i.e. existence time, valid time and transaction time) can be assigned to each construct. For example, STER defines that existence time can be attached to only entity types while transaction time can be attached to all constructs i.e. entity types, relationship types and attributes (Table B.1).

- **Temporal entity types.** Entity types can be assigned existence and transaction time. Support for existence time for an entity type in STER is indicated by placing an ‘et’ in a circle in the upper-right corner for the

entity type as shown in Figure B.8(a). The detailed representation diagram is shown in Figure B.8 (b) indicating that ‘existence time/id’ values consist of pairs of ‘existence time’ and ‘id’ values. The shorthand notation is convenient as it concisely states that the existence times of the entities should be captured in the database.

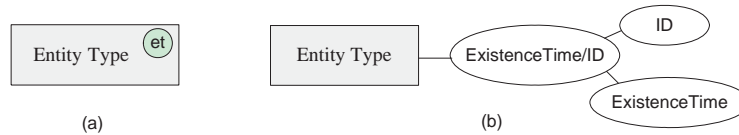


Figure B.8: Entity type capturing existence time (a) shorthand (b) detailed representation (from Tryfona and Jensen (1999)).

- **Spatial entity types.** STER entity types can be defined to capture space, i.e. to have a particular geometry attribute. STER supports three spatial geometric types (point, line and region) and their combinations. The letters, ‘s’, ‘P’, ‘L’ or ‘R’, in a circle in the lower-right corner of an entity type specify the type of spatial support. Letter ‘s’ stands for Spatial and is used to indicate a spatial entity type whose exact geometric type is unknown. Letters ‘P’, ‘L’ and ‘R’ specify geometric types Point, Line and Region, respectively. Examples of a spatial entity type is shown in Figure B.9.

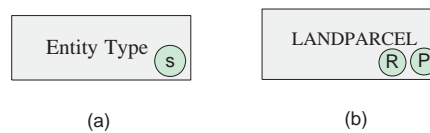


Figure B.9: Spatial entity types (a) general notation (b) capturing Point and Region (from Tryfona and Jensen (1999)).

- **Spatio-temporal entity types.** Spatial entity types with temporal dimensions are called spatio-temporal entity types. The temporal aspects (valid time, transaction time and bitemporal time (both valid and transaction time)) of spatial entity types are recorded by placing ‘svt’, ‘stt’ or ‘sbt’ in a circle in the lower-right corner of the entity type as illustrated in Figure B.10. Note that if the geometric types of the geometric figures of the entity types are known, the ‘s’ can be replaced with ‘P’, ‘L’ or ‘R’ or a combination of these as appropriate. This is placed in a circle in the lower-right corner of the entity type as illustrated in Figure B.10.

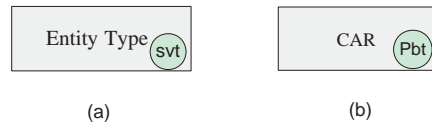


Figure B.10: A Spatio-temporal entity type with valid time support (a) general notation (b) example recording of a *car*'s position with valid and transaction time support (from Tryfona and Jensen (1999)).

- **Temporal descriptive attributes.** Values of attributes of entities denote facts about the entities and thus have both valid and transaction time aspects. A circle with a 'vt' or a 'tt' in the upper-left corner of an oval denoting an attribute indicates that valid or transaction time, respectively, is to be captured. A circle with 'bt' (bitemporal) indicates that both temporal aspects are to be captured. An attribute with valid time support is shown in Figure B.11.

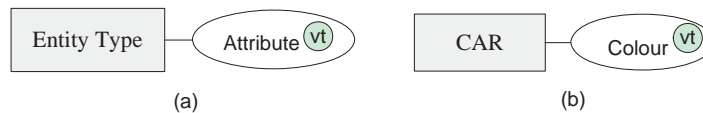


Figure B.11: An attribute with valid time support (a) general notation (b) example recording of a *car*'s colour and its valid time (from Tryfona and Jensen (1999)).

- **Spatial attributes.** Facts captured by attributes may also have associated locations in space, which are described as sets of geometric figures. To capture this spatial aspect of an attribute, a circle with an 's' is used, as shown in Figure B.12. Figure B.12 (a) depicts the general representation of a spatial attribute, while Figure B.12 (b) shows an example of a 'soil type' value of a land parcel that is associated with a set of spatial regions ('R').

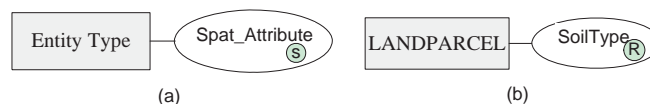


Figure B.12: A spatial attribute (a) general notation (b) example of SoilType as a spatial attribute (from Tryfona and Jensen (1999)).

- **Spatio-temporal attributes.** Spatial attributes with temporal dimensions are termed spatio-temporal attributes. The temporal aspects (valid and transaction time) of spatial attributes are recorded by placing 'svt', 'stt' or

‘sbt’ in lower-right corner of the entity type as illustrated in Figure B.13. As with entity types, if the geometric types of the geometric figures of the attributes are known, the ‘s’ can be replaced with ‘P’, ‘L’ or ‘R’ or a combination of these as appropriate.

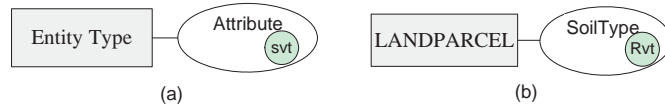


Figure B.13: A spatio-temporal attribute in STER (a) general notation (b) example of SoilType as a spatio-temporal attribute (from Tryfona and Jensen (1999)).

- **Temporal relationship types.** Temporal aspects (valid and transaction time or both) can be attached to relationship types in the same way as attributes.
- **Spatial relationship types.** Spatial relationship types in STER are associations between the geometries of the spatial entities. For example, the relationship *traverses* between cities and rivers relates the geometries of entities of these two spatial entity types.
- **Spatio-temporal relationship types.** A spatio-temporal relationship type is a spatial relationship type with time support. Figure B.14(a) shows the general representation of a spatio-temporal relationship type, while Figure B.14 (b) depicts the example of changes to the relationship *traverses* between cities and rivers as recorded over time.

Note that, the previous discussion about temporal, spatial and spatio-temporal attributes also applies to attributes of relationship types.

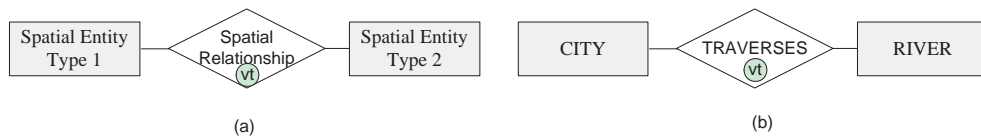


Figure B.14: A spatio-temporal relationship type (a) general notation (b) example of *traverses* as a spatio-temporal relationship type (from Tryfona and Jensen (1999)).

Integrity Constraints. There is no formal notation for constraints provided by STER other than those constraints provided by the spatial and temporal relationships such as topological and metric relationships (Friis-Christensen et al., 2001).

Languages. No query and manipulation languages are considered by STER.

In summary, STER is an extension of the basic ER model which includes spatio-temporal entities, attributes and relationships for modelling spatio-temporal information. STER offers support for the spatial data types (point, line and regions), for geometries as well as combinations thereof, and for the temporal aspects (existence time (**et**) for objects, valid time (**vt**) for attributes and relationships and transaction time (**tt**) for all constructs). Support for both valid time and transaction time is represented as bitemporal (**bt**).

B.4 The Modelling of Application Data with Spatio-temporal features (MADS)

The MADS approach (Parent et al., 2006a, 1999) is a conceptual spatio-temporal model based on the extended entity-relationship model (EER) model.

Data Structures. The modelling constructs provided by MADS extends the basic EER constructs as described below:

- **Spatial, temporal and spatio-temporal object types.** A Spatial (temporal) object type is an object type that holds spatial (temporal) information pertaining to the object itself as a whole. This is in contrast to other models where the spatial information only pertains to one of its components or characteristics, or to a link between the object and other objects. For example, a **road** object type in a cartographic database can be defined as a MADS spatial object type if it bears the information on the extent of the roads, such that it can be used to create a geographic map of the road network. An **employee** object type in a staff database can be defined as a MADS temporal object type if it bears the information over all the time periods when the person has been a regular employee of the company (as opposed to the discrete employment stages such as pre-hiring, training, sabbatical leave or outsourced).
- **Relationship types.** In addition to the basic ER relationship types, MADS defines multi-association relationship types and spatial, temporal and spatio-temporal relationship types. MADS also supports semantic enhancements to the basic relationship types in the form of aggregation, generation and

transition. For a detailed discussion of these characteristics, refer to Parent et al. (2006a, 1999).

- **Multi-association relationship types.** As discussed by Parent et al. (2006a), a multi-association relationship type is a relationship type that links, for each role, a non-empty set (or list) of instances of the linked object type. In other words, multi-association relationships link groups of objects (entities), rather than single objects. Consequently, each role in a multi-association relationship type bears two pairs of (minimum, maximum) cardinalities. The first pair is the conventional relationship type that defines for each object instance how many relationship instances it can be linked to via the role. The second (additional) pair defines for each relationship instance how many object instances it can link with this role and has a minimum value of 1. Multi-association relationship types are visually represented by a double ellipse as shown in Figure B.15.



Figure B.15: An example of a multi-association relationship type (from Parent et al. (2006a)).

- **Spatial, temporal and spatio-temporal relationship types.** These hold spatial and/or temporal information pertaining to the relationship as a whole, in the same way as object types. For example, a **crosses** relationship type may be defined to hold the spatial extent covered by the intersection of two roads. **Crosses** thus become a spatial relationship type.
- **Spatial, temporal and spatio-temporal attributes.** A Spatial (temporal) attribute is a simple attribute whose value domain belongs to one of the known spatial (temporal) data types. A spatio-temporal attribute is a time-varying spatial attribute. Each object and relationship type, whether spatio-temporal or not, can have zero, one, or more spatial, temporal and spatio-temporal attributes. For example, a **road** object type may include, in addition to its spatial extent, a spatial attribute `restAreas` holding the spatial extent of all rest areas along the road.

Integrity Constraints. MADS uses new kinds of integrity constraints that are embedded in the data model to deal with spatial and temporal data (refer to detailed discussion in Parent et al. (2006a)). MADS suggests the use of predefined spatial coverage, partition and disjointness constraints in association with spatial aggregation relationships. Thus, whenever topological constraints are applied to specific pairs of related instances, they can be conveyed by associating a topological constraint semantic to the relationship between these instances. As suggested by Parent et al. (2006a), these constraints may be expressed using a specific integrity constraint specification language, rather than being embedded as data modelling constructs. In summary, constraints in MADS are defined by topological (temporal) predicates.

Languages. The MADS model includes an algebraic data manipulating language that can be used to support querying and updating. MADS also provides a description of the translation process to facilitate an implementation into current DBMSs and GISs. Such translation processes have also been implemented in tools used with the MurMur project (Parent et al., 2006b).

In summary, MADS supports multi-associations and a larger number of semantic features of relationships (in particular, generation and transition) as well as providing some diversity in the way constraints are handled. The orthogonality principle is an important aspect in the MADS model in adding different modelling dimensions i.e. spatial and temporal characteristics. MADS also caters for multi-representation functionality in the model providing the opportunity for further extensions (Parent et al., 2006b). Some of the deficiencies of MADS are that it does not yet support transaction time, nor uncertainty.

As discussed by Khatri et al. (2006), MADS allows geospatiality to be associated with object types, attributes, relationships and aggregation. Additionally, space and time features are supported via abstract data types (ADT) while spatial ADTs provide shape and location information and temporal ADTs support timestamps. Spatial entities are associated with a spatial ADT e.g. point and line.

B.5 The DIstributed design of SpaTio-temporaL data (DISTIL)

DISTIL (Ram et al., 2001) is another recent design tool that provides a design support environment for modelling spatio-temporal databases. This approach classifies spatio-temporal conceptual design into two steps:

- capture the current reality of application using a conventional ER conceptual model, without considering the spatial aspects, and then
- annotate the schema with spatial and temporal semantics of the application.

This approach integrates the semantics related to space and time into a traditional conceptual model without adding any special constructs. DISTIL emphasises the idea of orthogonality that allows the new features to be associated with any structural constructs of the data model since these constructs of conceptual modelling (i.e. entity types, relationship type and attributes) are orthogonal to space and time.

Data Structures. The base data structures of DISTIL are *entity classes* (entity types in the ER model), *interaction relationships* (relationship types in the ER model) and *attributes* that are used for developing a core schema. Once the basic structure has been established, the spatial and temporal aspects of the application are then identified and annotated in the core schema. Refer to Ram et al. (2001) for a detailed description of the methods used to annotate such a schema.

Integrity Constraints. The detailed description of how constraints are used in DISTIL is not provided in the presented model (Ram et al., 2001).

Languages. An adequate language to support DISTIL is not provided in the presented model. However, it does provide a *logical mapper* to convert a conceptual schema to a relational schema with spatial support. This is implementable in a relational DBMS and the mapper can generate an SQL3/Temporal Logical Schema that can be implemented in a temporal DBMS.

In summary, DISTIL is an annotation-based approach to spatio-temporal conceptual modelling capturing various aspects related to temporality and spatiality such as valid time, transaction time, events, states, position, geometry and shape. In particular, DISTIL provides a mechanism to capture semantics related to granularity and indeterminacy. DISTIL provides the benefit that the model can be saved as XML schemata (Khatri et al., 2006). Refer to Khatri et al. (2006) and Ram et al. (2001) for details related to this prototype system.

B.6 The starER Model

The starER model (Tryfona et al., 1999) combines the constructs of the ER model (Chen, 1976) with the star structure that is used to manage the data in data warehouses. It has been proposed that this model can be used in the conceptual modelling of data warehouses and has already been tested in a mortgage business environment.

Data Structures. The modelling constructs of the starER model are as follows:

- **Fact sets.** A fact set, depicted as a circle (Figure B.16), represents a set of real-world facts sharing the same characteristics (or properties). A fact set is *always* associated with time. From a semantic viewpoint, this can be explained by the reason that a fact set refers to data that has been generated over time, i.e. data is generated in terms of facts that occur over discrete instances of time.

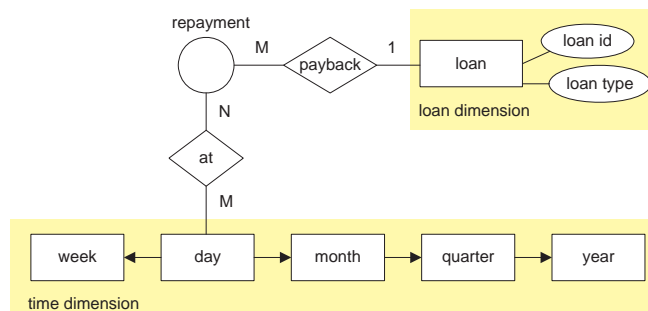


Figure B.16: An example of a starER diagram (from Tryfona et al. (1999)).

- **Entity sets.** This construct, depicted as rectangle (Figure B.16), represents a set of real-world objects with similar properties. It has the same meaning and symbol as in conventional ER modelling. Additionally, *time*, as well as

the other entity sets associated with a fact, are the dimensions of that fact. Dimensions consist of hierarchies and/or other relationships among other entity sets.

- **Relationship sets.** A relationship set represents a set of associations among entity sets and fact sets and is depicted as a diamond (Figure B.16). Relationship sets among entity sets can be of type specialisation/generalisation, aggregation or membership as shown in Figure B.17.

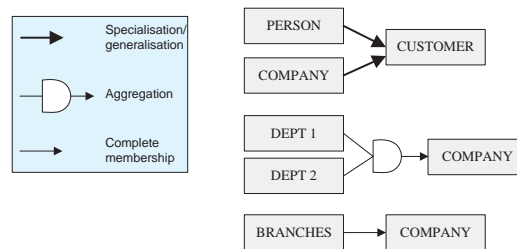


Figure B.17: Examples of specialisation, aggregation and membership relationship types (from Tryfona et al. (1999)).

- **Attributes.** Attributes are static properties of entity sets, relationship sets and/or fact sets and are illustrated by ovals (Figure B.16). Fact set properties can be of type stock, flow or value-per-unit. This is indicated by an ‘S’, ‘F’, or ‘V’ on the left of the attribute symbol, respectively.

Integrity Constraints. No integrity constraints are considered by starER.

Languages. No query and manipulation languages nor any mappings are provided by starER.

In summary, starER addresses the modelling requirements of a data warehouse and combines the star structure with the constructs of the ER model. Special types of relationship have been included to support hierarchies.

B.7 The MultiDimER Model

The MultiDimER model (Malinowski and Zimányi, 2006) is a conceptual multi-dimensional model that includes constructs for data warehouse and OLAP modelling. The MultiDimER model is mainly based on the existing ER model constructs. The model offers some important features for representing different kinds of hierarchies, levels and fact relationships.

Data Structures. A MultiDimER schema is defined as a finite set of dimension and fact relationships. A *dimension* is an abstract concept for grouping data that shares common semantics within the domain being modelled. It represents either a level or one or more hierarchies. Levels are presented as entity types. Every instance of a level is called a member.

- **Hierarchies.** A hierarchy contains several related levels that are used for roll-up and drill-down operations. A representation of hierarchies is shown in Figure B.18(b). For two consecutive levels of a hierarchy, the higher and lower levels are termed parent and child, respectively. A hierarchy level that has no child level is called a leaf. The top most level that has no parent is termed the root. The root represents the most general view of data. MultiDim represents different types of hierarchies at both a conceptual and logical level such as symmetric hierarchies, asymmetric hierarchies, generalised hierarchies, non-strict hierarchies, multiple alternative hierarchies and parallel dependent hierarchies.

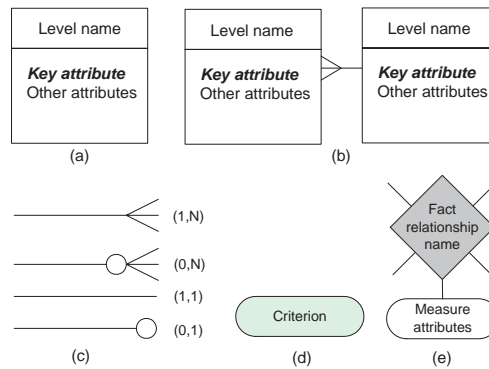


Figure B.18: Notations for a multidimensional model: (a) level, (b) hierarchy, (c) cardinalities, (d) analysis criterion, and (e) fact relationship (from Malinowski and Zimányi (2006)).

- **Level types.** A level type corresponds to a regular entity type in the ER model. Levels contain one or more key attributes and may also have other descriptive attributes. The granularity of measurement in a fact relationship is determined by the key attributes in a leaf level or the level forming a dimension without any hierarchy.
- **Relationship types.** A relationship type between child and parent levels of a hierarchy corresponds to a binary relationship in the ER model. A relationship joining child and parent levels is characterised by the cardinalities and

the analysis criterion. Cardinalities (Figure B.18(c)) indicate the minimum and the maximum numbers of members in one level that can be related to a member in another level. The analysis criterion (Figure B.18(d)) expresses different structures used for analysis, e.g. for a geographical location or an organisational structure.

- **Fact relationship types.** A fact relationship type corresponds to an n -ary relationship type in the ER model. A fact relationship (Figure B.18(e)) represents an n -ary relationship between leaf levels. These are often the focus of data mining analysis and may contain attributes commonly called measures. Measures usually store numerical data applicable to leaf members that reflect aggregated values of a hierarchy traversal. An example of the use of fact relationship types is illustrated in the conceptual model of a Sales Data Warehouse as shown in Figure B.19.

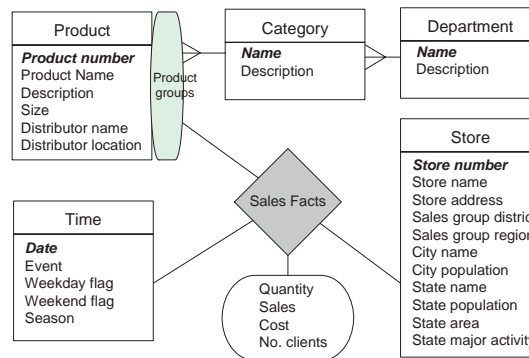


Figure B.19: A conceptual multidimensional schema of a Sales Data Warehouse including a hierarchy in the Product dimension (from Malinowski and Zimányi (2006)).

Integrity constraints. MultiDimER proposes *exclusive* constraints depicted by the symbol \otimes that are used to indicate that the paths for each member are exclusive. These constraints are necessary as the generalised hierarchy that is defined at the schema level contains multiple exclusion paths sharing the same levels. All of these paths represent one hierarchy and are shared whenever the same analysis criteria is used. At the instance level, each member of the hierarchy only belongs to one path.

Languages. No particular query language has been designed for MultiDimER, however, the proposal does provide for a general mapping of the Multi-DimER model to the relational model.

In summary, MultiDimER represents graphic notations for different types of hierarchies at a conceptual level. This allows for a clear distinction of each type of hierarchy taking into account their differences at the schema as well as the instance levels. MultiDimER provides for advanced features of a multidimensional model by accommodating hierarchical structures that can be used for data analysis. The model caters for the mapping of these hierarchies in the relational model.

B.8 The FuzzyEER Model

FuzzyEER (Galindo et al., 2006) extends the EER model by incorporating fuzzy semantics and fuzzy notations to represent imprecision and uncertainty in entities, attributes and relationships.

Data Structures. The principal constructs used in this model are fuzzy attributes, fuzzy entities and fuzzy relationships. A detailed discussion on all the constructs supported by the FuzzyEER proposal are provided by Galindo et al. (2006).

- **Fuzzy attributes.** A fuzzy attribute is a property of entities or relationships that have fuzzy (imprecise) values. It is classified into 4 types: Type 1, 2, 3 and 4.
 - **Fuzzy attribute Type 1.** The representation of attributes¹ in the FuzzyEER model is different from the classical attributes of the EER model. This type is represented by a normal circle with a line that joins it to the entity. The T1 notation is placed before the name of an attribute as shown in Figure B.20.

The fuzzy attributes can be defined with an optional value list, represented by {L1,L2,...} that is included next to attribute names (Figure B.20). These labels are defined in the data dictionary of the model. Note that the Type 1 also allows the declaration of fuzzy labels that can be used in different operations (e.g. queries). However, the Type

¹Attributes in the EER model are represented by the oval symbol notation. Attribute names are enclosed in oval symbols and are attached to their entity type by straight lines. However, in the FuzzyEER model, these symbols are replaced by a circle, and attribute names are placed outside the circle.

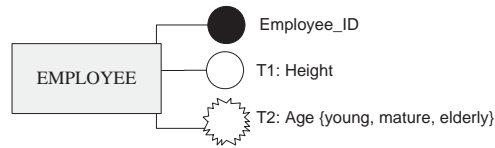


Figure B.20: Entity *employee* with fuzzy attributes Type 1 and Type 2 (from Galindo et al. (2006)).

1 is different from other fuzzy types in that it does not validate fuzzy information, and is restricted to simply allowing for fuzzy processing. In the example of Figure B.20, there are two fuzzy attributes of the *employee* entity: Height and Age. Height is defined as a fuzzy Type 1 attribute and its domain covers the set of values in the interval, for example, (0, 2.5). Height is a precise attribute, however, it can be defined using linguistic labels (such as ‘short’, ‘medium_height’ and ‘tall’) when manipulating this data in queries. In this example, it is presumed that the height is known, but if this is not the case, the null value is stored.

- Fuzzy attribute Type 2, 3 and 4. This representation is similar to Type 1, however a circle of zigzag lines and the notations T2, T3 or T4 are used instead. This is illustrated in the example of Figure B.20, where Age is a fuzzy Type 2 attribute, and its domain corresponds to a fuzzy set of ages considered as numerical values (ordered referential), allowing linguistic labels such as ‘young’, ‘mature’ and ‘elderly’.

These labels are defined as *possibility distributions* as shown in Figure B.21. For example, the label ‘young’ is defined as a trapezoidal function with four characteristic values of (0/15, 1/20, 1/25/, 0/30). This example also demonstrates a degree of confidence with this label, such as where the age 26 belongs to the range ‘young’ with a degree of confidence of 0.8.

- Fuzzy entities. A fuzzy entity is an entity with an attribute that possesses a membership degree (with any meaning). The notation of a fuzzy entity type is displayed as a rectangle with dashed lines. It has a fuzzy attribute representing the degree of certainty, which is displayed as a dashed circle connected to the entity with a dashed line as shown in Figure B.22.

In this example, the *employee* fuzzy entity is associated with an attribute that stores the total number of hours worked per week, where $Q(h)$ is the

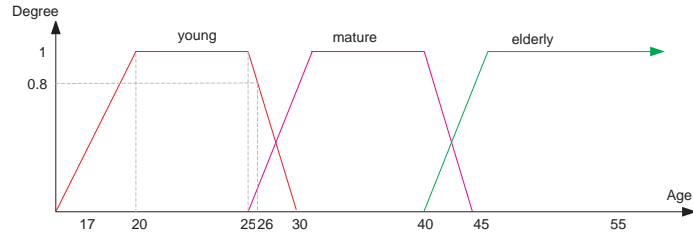


Figure B.21: Possibility distributions for the linguistic labels of the fuzzy T2 attribute: Age (from Galindo et al. (2006)).

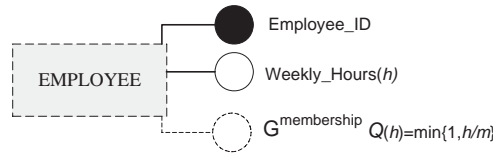


Figure B.22: Fuzzy entity with a membership degree (adapted from Galindo et al. (2006)).

calculus of the degree of membership and h represents the number of hours worked per week. Consider $Q(h)=\min\{1, h/m\}$, where m is the minimum number of hours for total membership. If $m = 35$, then an employee who works for 15 hours will be considered as an employee with a degree of 0.43 (the result of the division $15/35$). This demonstrates how the degree can be maintained in diverse calculations.

- **Fuzzy relationships.** A fuzzy relationship is a relationship that links between one or more entities and has at least a fuzzy degree defined by the expression G^n , where n is the meaning of a fuzzy degree in each specific context (refer to Galindo et al. (2006) for a clarification of fuzzy degree associated with each value of an attribute). The notation of a fuzzy relationship type is displayed as a diamond with a dashed line with a degree attribute as shown in Figure B.23.

In this example, the **district** entity can have the attributes (District.ID, Name, Quality). The Quality attribute is defined as a fuzzy attribute Type 3 with the labels: {Low, Regular, Good, Excellent}. The relationship of proximity of the neighbourhoods can be represented as the fuzzy relationship **Close_To**. This expresses that a proximity degree exists between any two districts. It is represented by the degree $G^{Proximity}$.

Integrity Constraints. The major constraints that are introduced in Fuzzy-EER are as follows.

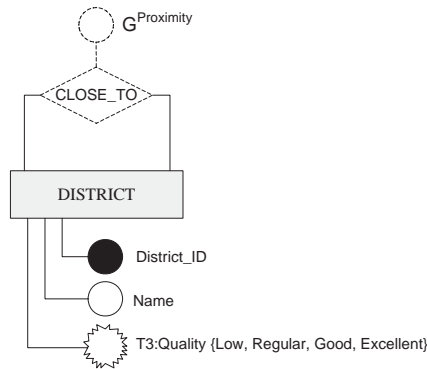


Figure B.23: A portion of an example of a fuzzy relationship (from Galindo et al. (2006)).

- **Fuzzy participation constraints.** A fuzzy participation constraint of an entity in a relationship can be made fuzzy by using a relative fuzzy quantifier. Let E_1 and E_2 be two entities and R a relationship between them. A fuzzy participation constraint of E_1 in R is represented by using a zigzag line joining E_1 and R , with a quantifier Q labelling this line, followed optionally by one or two thresholds, $[\gamma]$ or $[\gamma, \delta]$.

As discussed by Galindo et al. (2006), a fuzzy quantifier can be written in three ways:

1. quantifier without a threshold (a default threshold is $\gamma = 0.5$). For example, `approx_2`;
2. quantifier with a threshold γ . For example, `approx_8[0.25]`; or
3. quantifier with two thresholds γ and δ , with $\gamma < \delta$. For example, `approx_3[0.25, 0.75]`.

The threshold $\gamma = 0.2$ in ‘almost_all[0.2]’ (Figure B.24) indicates the minimum degree with which this quantifier must be fulfilled in the database. This example illustrates how fuzzy qualifiers such as ‘almost all’ have been used to define the participation of **employee** in this relationship. Refer to Galindo et al. (2006) for a detailed discussion on the fuzzy participation constraint in relationships.



Figure B.24: An example of a fuzzy participation constraint, using the fuzzy quantifier `almost_all` (from Galindo et al. (2006)).

- **Fuzzy cardinality constraints.** A fuzzy cardinality constraint is defined with two quantifiers, separate by a colon (:), named as $Q_1:Q_2$. Fuzzy cardinality constraints establish fuzzy conditions on each instance in a specific and unique way (refer to Galindo et al. (2006) for a detailed discussion about fuzzy conditions).

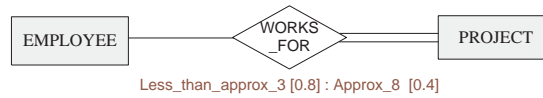


Figure B.25: An example of a fuzzy cardinality constraint in a FuzzyEER model. (from Galindo et al. (2006)).

In the example of Figure B.25, these constraints express the condition that each employee works for a maximum of approximately three projects, and each project has approximately eight employees, thus requiring both constraints to be satisfied with the minimum fulfilment degrees indicated in square brackets.

Fuzzy participation and cardinality constraints can be expressed as a fuzzy (min,max) notation on relationships as shown in Figure B.26, where both min and max can have values, and are both fuzzy quantifiers (Galindo et al., 2006).

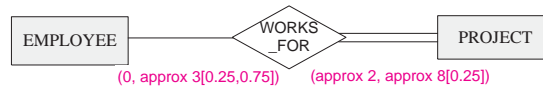


Figure B.26: An example of fuzzy (min,max) notation in a FuzzyEER model. (from Galindo et al. (2006)).

In the example, the constraint on the employee side indicates that an employee may work for no projects (0 as *minimum*) and up to approximately three projects as a *maximum*. The two values (0.25 and 0.75) indicates the minimum and maximum degree of a quantifier. The constraint cannot be fulfilled to a degree of less than 0.25, but can be fulfilled where the degree is greater than or equal to 0.75. At the same time, the number of employees in each project is restricted to a maximum of approximately eight (with a minimum degree of 0.25).

Languages. The concepts of the FuzzyEER model can be translated to the relational model, and a mapping algorithm is provided to facilitate this. Some of the FuzzyEER notations may be used in the FSQL (FuzzySQL) server that enhances the basic SQL to deal with fuzzy queries and operations

(Galindo et al., 2006). This proposal also provides a tool called FuzzyCASE to help with system design.

In summary, FuzzyEER model is an extension of the EER model with fuzzy semantics and notations that allows imprecise and uncertain information to be represented at a conceptual level. Other extensions to the ER model that have attempted to deal with fuzzy (or vague) data have been researched (Zvieli and Chen, 1986; Chen and Kerre, 1998; Ma et al., 2001; Ma, 2005, 2006), but none of these refer to the possibility of expressing constraints using the tools offered by fuzzy set theory (Galindo et al., 2006). FuzzyEER presents various fuzzy features for fuzzy modelling, e.g. fuzzy attribute values, membership degrees, fuzzy entities, fuzzy relations, fuzzy aggregation and fuzzy constraints.

B.9 The ER extended for XML (EReX) Model

The EReX model (Mani, 2004) is an extension to the ER model which includes categories, coverage constraints and order constraints.

Data Structures. This approach proposes *categories* of entity types that can be modelled using category relationship types. Category relationship types are similar to *is-a* relationship types in the EER model and are depicted as an arrow from its category entity type to its categorised entity type as shown in Figure B.27. These differ from the *is-a* relationship type in that a categorised entity type may have an empty key. The example in Figure B.27 shows the categorised *person* entity and its *reviewer* and *author* categories. The key of *person* is empty.

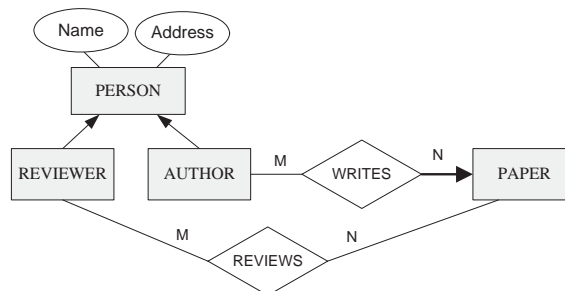


Figure B.27: An example of an EReX schema.

Integrity Constraints. EReX introduces *coverage* and *order* constraints. There are two kinds of coverage constraints, total coverage and exclusive coverage, which can be specified for categories and for roles of entity types in relationship types. Coverage constraints have been previously discussed in Elmasri et al. (1985).

- **Total coverage constraints.** A total coverage constraint specifies that the union of sets of instances of all included categories or roles must be the same as a set of instances of the categorised entity type or the entity type with the included roles.

Note that in the EReX diagram there are no specific graphic notations for expressing the coverage constraints. Instead, these are represented as a text notation with mathematical operations. In the example schema in Figure B.27, the total coverage constraint is defined as $Reviewer + Author = Person$ that specifies that each person is either a reviewer or an author as no other roles are allowed.

- **Exclusive coverage constraints.** An exclusive coverage constraint specifies the disjunction between the sets of instances of the included categories or roles.
- **Order constraints.** An order constraint is specified for entity types in a relationship type. For instance, if an ordering on entity type E in relationship type R is specified, then for a given entity e of E the relationship instances in R with e as a participant are ordered. This constraint is depicted as a thick line between E and R , as indicated in Figure B.27. This example shows that there is an ordering specified on the **paper** entity type in the **writes** relationship type which means that the authors of a given paper are ordered.

Languages. This model provides an algorithm to translate an EReX schema to an XML schema that can be used as a logical model for an XML database. However, this proposal does not provide any additional operators or inferring rules for querying and manipulating XML data.

In summary, EReX extends the ER model with additional XML features providing structural and constraint specifications that can be modelled using traditional ER modelling methodologies. A method of translating an EReX schema to XML is also provided. While XML utilises the notion of document order, where all the elements are ordered, there is no equivalent use of these concepts in EReX.

B.10 The XSEM-ER Model

The XSEM-ER model (Nečaský, 2007) extends the classical ER model to model XML data. This model is generated in two stages associated with the XSEM model (Nečaský, 2007). The first stage is used to create an overall conceptual schema of the application domain that has been designed using the XSEM-ER model. The second stage uses XSEM-H to convert the structures generated by XSEM-ER into hierarchical organisations of the data. As this thesis is only concerned with ER modelling extensions, only the first part of the XSEM model is described, namely the XSEM-ER model.

Data Structures. The basic modelling constructs (entity types, relationships type and attributes) of the XSEM-ER model are essentially the same as those of the ER model. These XSEM-ER constructs differ from the ER model in the following 3 ways: (1) a list of attributes of an entity type and a list of attributes of a relationship type are ordered; (2) an entity type can have an empty key or it can have a non-empty key composed of optional attributes; and (3) a weak entity type is depicted using a hexagon appended to the entity box, rather than surrounding the box with double lines as in the ER model. This approach also proposed new modelling constructs, *data node types* and *cluster types*, for modelling semi-structured, irregular and heterogeneous data.

- **Data node types.** For modelling semi-structured data, a data node type is used to represent unstructured data values assigned to a given entity. It is depicted as an ellipse with a name of data node type, such as a VisitDesc data node type as shown in Figure B.28. This example shows the situation of a patient visiting a physician as discussed by Nečaský (2007). For each visit date, the physician writes a descriptive log of the visit details. This description is unstructured text that is assigned to the visit; it is not an attribute value of the visit. This log is intermixed with anamneses and new examination details from the visit. To model this mixture of descriptions, the VisitDesc data node type is used as shown in Figure B.28.
- **Cluster types.** A cluster type is used to model irregular and heterogeneous data and to group relevant constructs used to model mixed content in XML documents. These can be classified as *outgoing* and *incoming* cluster types as described below:

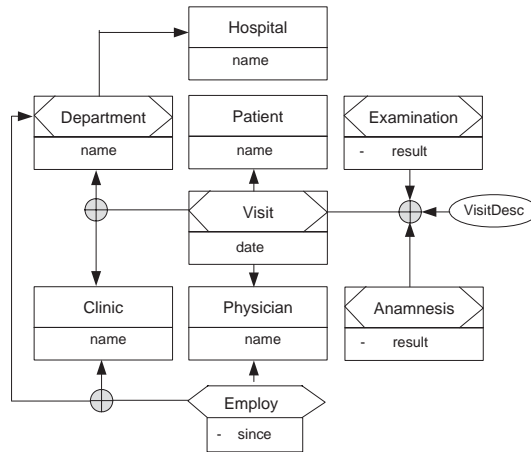


Figure B.28: XSEM-ER data node types and cluster types (from Nečaský (2007)).

- **Outgoing cluster types.** An outgoing cluster type represents a union of entity types, and is used to model irregular and heterogeneous data. It can be used as a participant of another outgoing cluster type, a participant of a relationship type or as a determinant of a weak entity type. Outgoing cluster types that have evolved from the notion of HERM cluster types (Thalheim, 2000) are depicted as \oplus and are connected by a solid line with a relationship type or weak entity type in which they participate as shown in Figure B.28. Each participant of the cluster type is connected by an arrow emanating from the circle to the participant. This example illustrates an irregular modelling structure with patients able to visit physicians at hospital departments as well as at separate clinics. To model this situation, an outgoing cluster type $\text{Department} \oplus \text{Clinic}$ is used as shown in Figure B.28.
- **Incoming cluster types.** Incoming cluster types are used for grouping relationship types, weak entity types and data node types that have the same participant/determinant/parent. These are called *parent* of the incoming cluster type, and are depicted as \oplus and is connected by a solid line with its parent. Each participant/determinant/parent is connected by an arrow going to this cluster type circle. In conjunction with data node types and ordering constraints, incoming cluster types are used to model mixed content in XML documents. For example, the incoming cluster type $(\text{Visit}, \text{Examination} \oplus \text{Anamnesis} \oplus \text{VisitDesc})$ can be used to create a description of an overall medical situation for a patient, covering current examination and visit details, as well as their

anamnesis as shown in Figure B.28. An example of XML representation of such a description as discussed by Nečaský (2007) is shown below:

```

<visit><date>26-09-2008</date>
<patient>
<name>John Black</name>
</patient>
<physician>
<name>Bill White</name>
</physician>
<department><name>Department A</name>
<hospital>
<name>Hospital B</name>
</hospital>
</department>
Because of the family
anamnesis (<anamnesis>...</anamnesis>)
there is a suspicion of liver cancer.
Thus, I made a laboratory examination
(<examination>...</examination>)...
</visit>

```

Integrity Constraints. This model proposes the use of *ordering constraints* that extend the notion of cardinality constraints in the ER model. There are two types of ordering constraints. The first specifies ordering on hierarchical projections of relationship types and the second specifies ordering on incoming cluster types.

Languages. The XSEM-ER model does not provide for any additional operators or inferencing rules for querying and manipulating XML data. This research does suggest that future work can investigate new algorithms that can be used for the translation of XSEM-H schemata to the logical XML level.

In summary, the main features of XML data are its hierarchical and irregular structure, with both ordered and mixed content. XSEM extends the ER model to represent these features by allowing XSEM-ER to model an overall non-hierarchical conceptual schema of a domain, followed by the use of XSEM-H to model a hierarchical organisation of the structures. Thus, the hierarchical organisation is derived from the transformation of the XSEM-ER constructs to XSEM-H constructs.

Appendix C

The Full Schema Mapping for MDER and OntoER models

As support for Chapter 8, this appendix provides the full mapping of MDER and OntoER model constructs for relational schemata. The extended mapping algorithm includes additional steps of an algorithm that are used for converting the mesodata constructs in the MDER model and ontological constructs in the OntoER model into relational schemata.

The results of the full mapping of the INVENTORY MDER schema and the MEDICAL OntoER schema into a relational database schema are presented in Figures C.1 and C.2. Their results also include the schemata that result from using the basic ER/EER-to-relational mapping algorithm.

C.1 The Result of the Full Mapping of the INVENTORY MDER Schema

The result of a full mapping of the INVENTORY MDER schema in Figure 8.3 into a relational schema is shown in Figure C.1.

C.2 The Result of the Full Mapping of the MEDICAL OntoER Schema

The result of a full mapping the MEDICAL OntoER schema in Figure 8.6 into a relational schema is shown in Figure C.2.

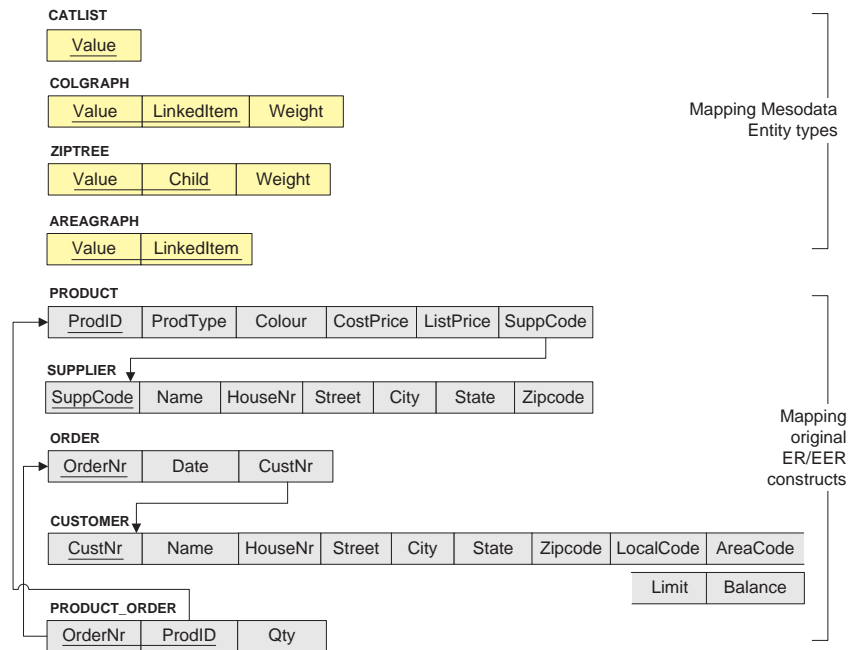


Figure C.1: Result of mapping the INVENTORY MDER schema into a relational schema.

C.3 Correcting the Schema: Normalisation

The transformation from an ER schema only guarantees the first normal form (1NF) relational database schema, and thus the relational schemata may need further normalisation¹. This may be required as the relationship types in the ER model are only between entity types, while the relationships between the attributes (functional dependencies) which can lead to redundancies are not taken into consideration.

One of the key concepts of relational database design is to prevent unnecessary duplication of data. Consider the **PATIENT** and **PHYSICIAN** schema from Figure C.2 that is shown with functional dependencies in Figure C.3. To understand why the **PATIENT** and **PHYSICIAN** schema may need further normalisation, it is first required to understand the concept of a *functional dependency* (FD) (refer to detailed discussions in Date (2003) and Elmasri and Navathe (2007)). In the example, both the **PATIENT** and **PHYSICIAN** schemata are not in third normal form (3NF) as they have a non-key attribute functionally determined by other non-key attributes.

This is illustrated with the Zipcode attributes that can be used to identify the

¹Normalisation is a methodology used to analyse an association between attributes to extract functional and join dependencies from the real-world semantics.

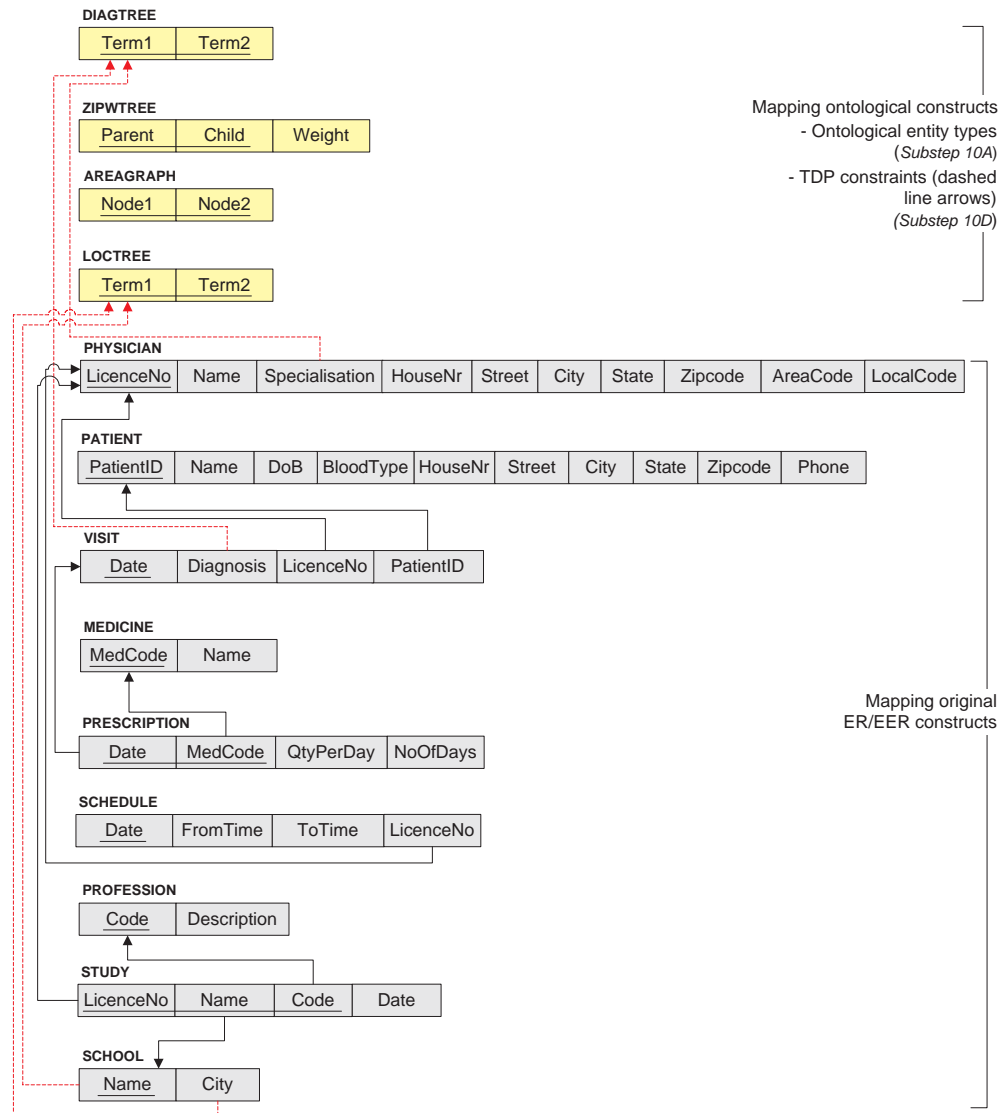


Figure C.2: Result of mapping the MEDICAL OntoER schema into a relational schema.

State attribute. In other words, this use of the State attribute is redundant as it can be inferred from Zipcode values. Thus, the notation FD: Zipcode \rightarrow State holds in both PATIENT and PHYSICIAN.

Normalisation is used to remedy this problem resulting in the schema decomposition as shown in Figure C.4. This method includes:

1. Decompose and set up the new schema that includes the non-key attribute(s) that functionally determine(s) other non-key attribute(s). The determinant attribute (the left-hand side of the FD), becomes the primary key of the new schema. In this example, Zipcode becomes a primary key of

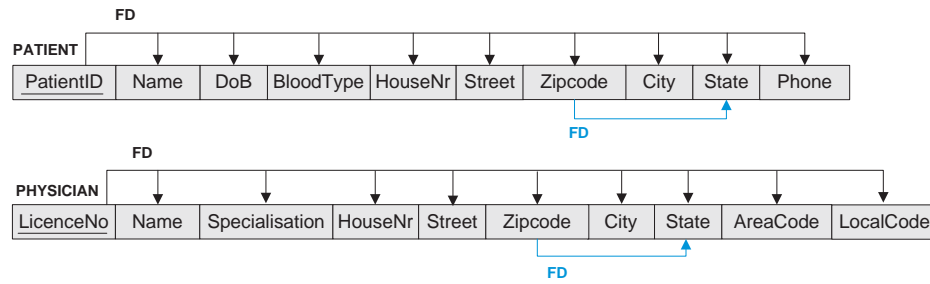


Figure C.3: The PATIENT and PHYSICIAN schema with their functional dependencies.

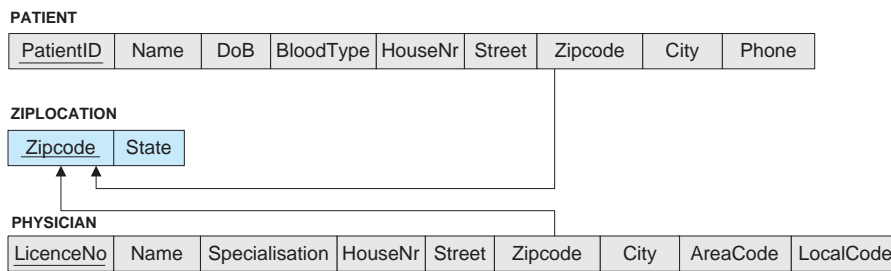


Figure C.4: Normalisation into 3NF.

a new schema ZIPLOCATION.

2. Leave the determinant attribute in the original schema, where it now becomes a foreign key into the new schema.

To normalise PATIENT schema into 3NF, it is decomposed into the two schemata: new PATIENT and ZIPLOCATION. The new PATIENT and PHYSICIAN schemata are constructed by removing the State attributes. The new ZIPLOCATION schema, shown in Figure C.4, contains the Zipcode and State attributes. Both the PATIENT and ZIPLOCATION schemata are in 3NF. Normalisation of the PHYSICIAN schema into 3NF can be achieved in the same way as the PATIENT schema. Since the two new ZIPLOCATION schemata derived from decomposing the PATIENT and PHYSICIAN schemata have exactly the same key attribute and a non-key attribute, the single ZIPLOCATION schema can be used that is referenced by both the PATIENT and PHYSICIAN schemata as shown in Figure C.4. Note that the schema in Figure C.4 is also in fifth normal form (5NF).

A result of the normalised MEDICAL OntoER schema is shown in Figure C.5. All relational schemata in Figure C.5 are in 5NF.

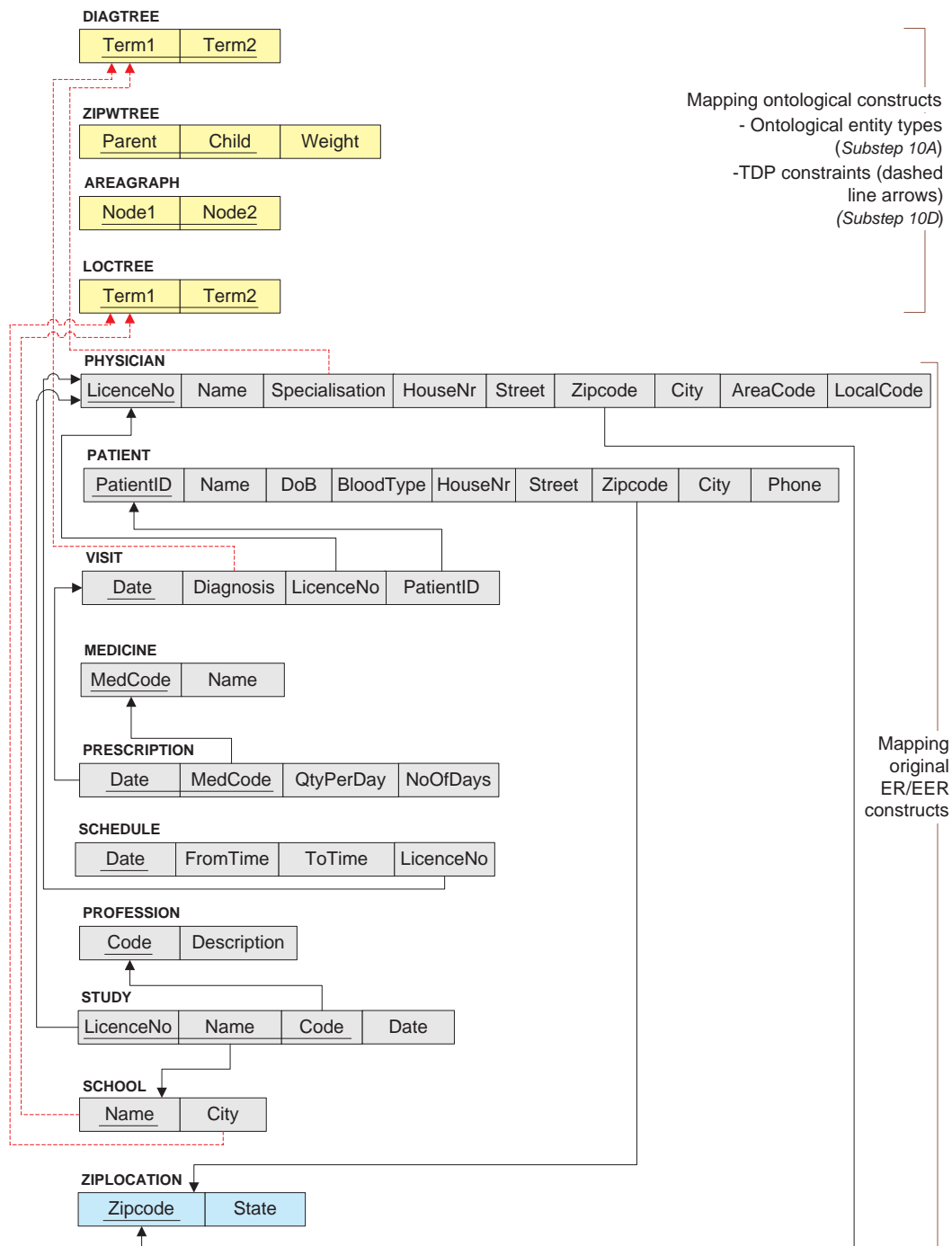


Figure C.5: A normalised schema of the MEDICAL OntoER schema.

Appendix D

The Full SQL Data Definition Language (MDDL and DDL)

This appendix details the full data definition SQL commands consisting of MDDL (Mesodata DDL) and DDL for the `INVENTORY` and `MEDICAL` schemata examples that are based on the `MDER` schema (Figure 8.3) and `OntoER` schema (Figure 8.6), respectively. The `CREATE` statement can be used to create tables, mesodata domains, ontological domains, domain's references for attributes and a source table for the domains as well as other constructs (such as total mesodata domain participation (TMDP) or total domain participation (TDP) constraints). The method of defining either mesodata or ontological data is described below:

1. Create the source relational schema corresponding to the structure of mesodata types (or ontological data types/common domain structures) selected using the `CREATE TABLE` command. An example of the source relational schema for describing the mesodata type `WTREE` is created as below:

```
CREATE TABLE ziptree (
  Value      CHAR(15) NOT NULL,
  Child      CHAR(15) NOT NULL,
  Weight     NUMERIC,
  PRIMARY KEY (Value, Child));
```

2. Create the domain over the source relational schema using the `CREATE DOMAIN` command.

```
CREATE DOMAIN ZIPCODES
  AS      wtree
  OF      CHAR(15)
  OVER    ziptree;
```

3. Reference attribute to the domain created in step 2, using the `CREATE TABLE` command. Example of Zipcode attribute referencing the `ZIPCODES` domain is as below:

```
CREATE TABLE customer (
  CustNr CHAR(6) NOT NULL,
  Name CHAR(40),
  HouseNr CHAR(6),
  Street CHAR(15),
  City CHAR(15),
  State CHAR(15),
  Zipcode ZIPCODES,
  LocalCode CHAR(10),
  :
  :
```

4. Handle integrity constraints that are imposed on the models. Consider the following two cases:

- to handle the `TMDP` constraints in the `MDER` model (or the `MMR` constraints in the `MDORM` model), the SQL Clause `CLOSED` can be used as shown below:

```
CREATE TABLE product (
  ProdID CHAR(6) NOT NULL,
  ProdType CATEGORIES CLOSED,
  Colour COLOURS CLOSED,
  ListPrice NUMERIC,
  CostPrice NUMERIC,
  SuppCode CHAR(5),
  PRIMARY KEY (ProdId),
  :
  :
```

- to handle the `TDP` constraints in the `OntoER` model (or the `OMR` constraints in the `OntoORM` model or the `OT` constraints in the `OntoUML` model), the `FOREIGN KEY` statements can be used as shown below:

```
CREATE TABLE school (
  Name LOCATIONS NOT NULL ,
  City LOCATIONS,
  PRIMARY KEY (Name),
  FOREIGN KEY (Name) REFERENCES loctree(Term1),
  FOREIGN KEY (City) REFERENCES loctree(Term1));
```

D.1 SQL Data Definition Statements for Defining the INVENTORY Schema

The relational schema from the mapping of the INVENTORY MDER schema in Figure C.1 (Page 259) is defined using the following statements:

```

CREATE TABLE  catlist      (
    Value      CHAR(50) NOT NULL,
    PRIMARY KEY(Value));

CREATE DOMAIN  CATEGORIES
    AS        list
    OF        CHAR(50)
    OVER      catlist;

CREATE TABLE  colgraph    (
    Value      CHAR(15) NOT NULL,
    LinkedItem CHAR(15) NOT NULL,
    Weight     NUMERIC,
    PRIMARY KEY (Value, LinkedItem));

CREATE DOMAIN  COLOURS
    AS        wgraph
    OF        CHAR(15)
    OVER      colgraph;

CREATE TABLE  ziptree     (
    Value      CHAR(15) NOT NULL,
    Child      CHAR(15) NOT NULL,
    Weight     NUMERIC,
    PRIMARY KEY (Value, Child));

CREATE DOMAIN  ZIPCODES
    AS        wtree
    OF        CHAR(15)
    OVER      ziptree;

CREATE TABLE  areagraph   (
    Value      NUM(2) NOT NULL,
    LinkedItem NUM(2) NOT NULL,
    PRIMARY KEY (Value, LinkedItem));

CREATE DOMAIN  AREACODES
    AS        graph
    OF        NUM(2)
    OVER      areagraph;

CREATE TABLE  product     (
    ProdID     CHAR(6) NOT NULL,
    ProdType   CATEGORIES CLOSEDa,
    Colour     COLOURS CLOSED,
    ListPrice  NUMERIC,
    CostPrice  NUMERIC,
    SuppCode   CHAR(5),
    PRIMARY KEY (ProdId),
    FOREIGN KEY (SuppCode) REFERENCES supplier(SuppCode));

```

^aThe SQL Clause CLOSED indicates that there is a TMDP constraint on the attribute.


```

CREATE TABLE customer
(
  CustNr      CHAR(6) NOT NULL,
  Name        CHAR(40),
  HouseNr     CHAR(6),
  Street      CHAR(15),
  City        CHAR(15),
  State       CHAR(15),
  Zipcode     ZIPCODES,
  LocalCode   CHAR(10),
  AreaCode    AREACODES,
  Limit       NUMERIC,
  Balance     NUMERIC,
  PRIMARY KEY (CustNr));

CREATE TABLE supplier
(
  SuppCode    CHAR(5) NOT NULL,
  Name        CHAR(40),
  HouseNr     CHAR(6),
  Street      CHAR(15),
  City        CHAR(15),
  State       CHAR(15),
  Zipcode     ZIPCODES,
  PRIMARY KEY (SuppCode));

CREATE TABLE order
(
  OrderNr     NUMERIC NOT NULL,
  Date        DATE,
  CustNr      CHAR(6) NOT NULL,
  PRIMARY KEY (OrderNr),
  FOREIGN KEY (CustNr) REFERENCES customer(CustNr));

CREATE TABLE product_order
(
  OrderNr     NUMERIC NOT NULL,
  ProdId      CHAR(6) NOT NULL,
  Qty         NUMERIC,
  PRIMARY KEY (OrderNr, ProdId),
  FOREIGN KEY (OrderNr) REFERENCES order(OrderNr),
  FOREIGN KEY (ProdID) REFERENCES product(ProdID));

```

D.2 SQL Data Definition Statements for Defining the MEDICAL Schema

The normalised relational schema of the MEDICAL OntoER schema in Figure C.5 (Page 262) is defined using the following statements:

```

CREATE TABLE diagtree
(
  Term1       CHAR(30) NOT NULL,
  Term2       CHAR(30) NOT NULL,
  PRIMARY KEY (Term1, Term2));

CREATE DOMAIN DIAGNOSES
AS tree
OF CHAR(50)
OVER diagtree;

```

```

CREATE TABLE zipwtree (
  Parent      CHAR(30) NOT NULL,
  Child       CHAR(30) NOT NULL,
  Weight      NUMERIC,
  PRIMARY KEY (Parent, Child));

CREATE DOMAIN ZIPCODES
  AS          wtree
  OF          CHAR(15)
  OVER       zipwtree;

CREATE TABLE areagraph (
  Node1       CHAR(30) NOT NULL,
  Node2       CHAR(30) NOT NULL,
  PRIMARY KEY (Node1, Node2));

CREATE DOMAIN AREACODES
  AS          graph
  OF          CHAR(15)
  OVER       areagraph;

CREATE TABLE loctree (
  Term1       CHAR(30) NOT NULL,
  Term2       CHAR(30) NOT NULL,
  PRIMARY KEY (Term1, Term2));

CREATE DOMAIN LOCATIONS
  AS          tree
  OF          CHAR(50)
  OVER       loctree;

CREATE TABLE ziplocation (
  Zipcode     ZIPCODES NOT NULL,
  State       CHAR(15),
  PRIMARY KEY (Zipcode));

CREATE TABLE patient (
  PatientID   CHAR(6) NOT NULL,
  Name        CHAR(40),
  DoB         DATE,
  BloodType   CHAR(2),
  HouseNr     CHAR(6),
  Street      CHAR(15),
  Zipcode     ZIPCODES,
  City        CHAR(15),
  Phone       CHAR(10),
  PRIMARY KEY (PatientId),
  FOREIGN KEY (Zipcode) REFERENCES ziplocation(Zipcode));

CREATE TABLE physician (
  LicenceNo   CHAR(6) NOT NULL,
  Name        CHAR(40),
  Specialisation  DIAGNOSES,
  HouseNr     CHAR(6),
  Street      CHAR(15),
  Zipcode     ZIPCODES,
  City        CHAR(15),
  LocalCode   CHAR(10),
  AreaCode    AREACODES,
  PRIMARY KEY (LicenceNo),

```

```

FOREIGN KEY (Specialisation) REFERENCES diagtree(Term1),
FOREIGN KEY (Zipcode) REFERENCES ziplocation(Zipcode);

CREATE TABLE visit          (
Date                        DATE NOT NULL,
Diagnosis                   DIAGNOSES,
PatientID                   CHAR(6),
LicenceNo                   CHAR(6),
PRIMARY KEY (DATE),
FOREIGN KEY (PatientID) REFERENCES patient(PatientID),
FOREIGN KEY (LicenceNo) REFERENCES physician(LicenceNo),
FOREIGN KEY (Diagnosis) REFERENCES diagtree(Term1));

CREATE TABLE medicine      (
MedCode                     CHAR(10) NOT NULL,
Name                        CHAR(50)
PRIMARY KEY (MedCode));

CREATE TABLE prescription  (
Date                        DATE NOT NULL,
MedCode                     CHAR(10) NOT NULL,
QtyPerDay                   CHAR(6),
NoOfDay                     INTEGER(2),
PRIMARY KEY (Date, MedCode),
FOREIGN KEY (Date) REFERENCES visit(Date);
FOREIGN KEY (MedCode) REFERENCES medicine(MedCode));

CREATE TABLE schedule     (
Date                        DATE NOT NULL,
FromTime                    Time,
ToTime                      Time,
PRIMARY KEY (Date),
FOREIGN KEY (LicenceNo) REFERENCES physician(LicenceNo));

CREATE TABLE profession   (
Code                        CHAR(10) NOT NULL,
Description                  CHAR(40),
PRIMARY KEY (Code));

CREATE TABLE school       (
Name                        LOCATIONS NOT NULL ,
City                        LOCATIONS,
PRIMARY KEY (Name),
FOREIGN KEY (Name) REFERENCES loctree(Term1),
FOREIGN KEY (City) REFERENCES loctree(Term1));

CREATE TABLE study        (
LicenceNo                   CHAR(6) NOT NULL,
Name                        DIAGNOSES NOT NULL,
Code                        CHAR(10) NOT NULL,
Date                        DATE,
PRIMARY KEY (LicenceNo, Name, Code),
FOREIGN KEY (LicenceNo) REFERENCES physician(LicenceNo),
FOREIGN KEY (Name) REFERENCES school(Name),
FOREIGN KEY (Code) REFERENCES profession(Code));

```

Bibliography

- Abelló, A., Samos, J. and Saltor, F. (2006), ‘YAM²: A multidimensional conceptual model extending UML’, *Information Systems* **31**(6), 541–567.
- Abraham, T. and Roddick, J. F. (1999), ‘Survey of spatio-temporal databases’, *Geoinformatica* **3**(1), 61–99.
- Abrial, J.-R. (1974), Data semantics, in J. W. Klimbie and K. L. Koffeman, eds, ‘IFIP Working Conference Data Base Management’, North-Holland, pp. 1–60.
- Akehurst, D. H. and Bordbar, B. (2001), On querying UML data models with OCL, in ‘Proceedings of the 4th International Conference on the Unified Modeling Language, Modeling Languages, Concepts, and Tools’, Vol. 2185 of *Lecture Notes in Computer Science*, Springer, pp. 91–103.
- Al-Kamha, R., Embley, D. W. and Liddle, S. W. (2007), Augmenting traditional conceptual models to accommodate XML structural constructs, in C. Parent, K.-D. Schewe, V. C. Storey and B. Thalheim, eds, ‘26th International Conference on Conceptual Modeling (ER 2007)’, Auckland, New Zealand, pp. 518–533.
- Al-Taha, K. K., Snodgrass, R. T. and Soo, M. D. (1993), ‘Bibliography on spatiotemporal databases’, *ACM SIGMOD Record* **22**(1), 59–67.
- Al-Taha, K. K., Snodgrass, R. T. and Soo, M. D. (1994), ‘Bibliography on spatiotemporal databases’, *International Journal of Geographical Information Systems (IJGIS)* **8**(1), 95–103.
- Amann, B. and Scholl, M. (1992), Gram: A graph data model and query language, in ‘European Conference on Hypertext Technology (ECHT)’, ACM, pp. 201–211.
- Angles, R. and Gutierrez, C. (2008), ‘Survey of graph database models’, *ACM Computing Surveys* **40**(1), 1–39.

- ANSI/X3/SPARC (1975), ‘Interim report: ANSI/X3/SPARC study group on data base management systems 75-02-08’, *FDT — Bulletin of ACM SIGMOD* **7**(2), 1–140.
- Bachman, C. W. (1996), Impact of object-oriented thinking on ER modeling, in B. Thalheim, ed., ‘Conceptual Modeling — ER’96’, Vol. 1157 of *Lecture Notes in Computer Science*, Springer, pp. 1–4.
- Badia, A. (2000), Extending entity-relationship models with higher-order operators, in ‘Proceedings of the 12th International Symposium on Foundations of Intelligent Systems’, Vol. 1932 of *Lecture Notes in Computer Science*, pp. 321–330.
- Badia, A. (2004), ‘Entity-relationship modeling revisited’, *SIGMOD Record* **33**(1), 77–82.
- Balaban, M. and Shoval, P. (2002), ‘MEER — an EER model enhanced with structure methods’, *Information Systems* **27**, 245–275.
- Batini, C., Ceri, S. and Navathe, S. (1992), *Conceptual Database Design: An Entity Relationship Approach*, Benjamin Cummings.
- Battista, G. D. and Lenzerini, M. (1993), ‘Deductive entity-relationship modeling’, *IEEE Transactions on Knowledge and Data Engineering* **5**(3), 439–450.
- Baumann, P. (1989), Valences: A new relationship concept for the entity-relationship model, in F. H. Lochovsky, ed., ‘Proceedings of the 8th International Conference on the Entity-Relationship Approach’, North-Holland, Toronto, Canada, pp. 59–72.
- Beneventano, D., Bergamaschi, S., Guerra, F. and Vincini, M. (2003), ‘Synthesizing an integrated ontology’, *IEEE Internet Computing Magazine* **7**(5), 42–51.
- Bergholtz, M. and Johannesson, P. (2001), Classifying the semantics of relationships in conceptual modeling by categorization of roles, in ‘Proceedings of the 6th International Workshop on Applications of Natural Language to Information Systems (NLDB’01)’, Madrid, Spain, pp. 199–203.
- Berild, S. (2004), Conceptual modeling — some thoughts, Technical report, Santa Anna IT Research Institute AB.
- Bird, L., Goodchild, A. and Halpin, T. (2000), Object role modelling and XML-schema, in A. H. F. Laender, S. W. Liddle and V. C. Storey, eds, ‘Proceedings of

- the 19th International Conference on Conceptual Modeling — ER 2000', Vol. 1920 of *Lecture Notes in Computer Science*, Springer, Salt Lake City, Utah, USA, pp. 661–705.
- Bloesch, A. and Halpin, T. (1996), ConQuer: A conceptual query language, *in* '15th International Conference on Conceptual Modeling', Vol. 1157 of *Lecture Notes in Computer Science*, Springer, pp. 121–133.
- Böhlem, M., Busatto, R. and Jensen, C. S. (1998), Point-versus interval-based temporal data models, *in* 'Proceedings of the 14th International Conference on Data Engineering', IEEE Computer Society, pp. 192–200.
- Bolour, A., Anderson, T. L., Dekeyser, L. J. and Wong, H. K. T. (1982), 'The role of time in information processing: A survey', *ACM SIGMOD Record* **12**(3), 27–50.
- Booch, G., Jacobson, I. and Rumbaugh, J. (2005), *Unified Modelling Language User Guide*, 2nd edn, Addison-Wesley.
- Booch, G., Rumbaugh, J. and Jacobson, I. (1999), *The Unified Modeling Language User Guide*, Addison-Wesley, Reading, MA.
- Borges, K. A., Davis Jr, C. A. and Laender, A. H. F. (2001), 'OMT-G: An object-oriented data model for geographic applications', *Geoinformatica* **5**(3), 221–260.
- Borst, W. N. (1997), Construction of engineering ontologies, PhD thesis, University of Twente.
- Borysowich, C. (2007), 'E-R diagram relationship integrity constraints', (<http://blogs.ittoolbox.com/eai/implementation/archives/er-diagram-relationship-integrity-constraints-14218>), accessed May 27, 2008.
- Bowers, S., Delcambre, L. and Maier, D. (2003), Superimposed schematics: Introducing E-R structure for *in-situ* information selections, *in* S. Spaccapietra, S. T. March and Y. Kambayashi, eds, 'Conceptual Modeling — 2002', Vol. 2503, Springer, pp. 90–104.
- Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E. and Yergeau, F. (2006), 'Extensible markup language (XML) 1.0 (5th edn)', (<http://www.w3.org/TR/REC-xml/>), accessed November 30, 2008.

- Bubenko jr, J. A. (2007), From information algebra to enterprise modelling and ontologies — a historical perspective on modelling for information systems, *in* J. Krogstie, A. L. Opdahl and S. Brinkkemper, eds, ‘Conceptual Modelling in Information Systems Engineering’, Springer, pp. 1–18.
- Buneman, P. (1997), Semistructured data, *in* ‘Proceedings of the 16th Symposium on Principles of Database Systems’, ACM Press, pp. 117–121.
- Campbell, D. M. and Embley, D. W. (1985), A relationally complete query language for an entity-relationship model, *in* ‘Proceedings of the 4th International conference on the Entity-Relationship Approach’, pp. 90–97.
- Campbell, D. M., Embley, D. W. and Czejdo, B. (1987), ‘Graphical query formulation for an entity-relationship model’, *Data & Knowledge Engineering* **2**(2), 89–121.
- Camps Pare, R. (2002), ‘From ternary relationships to relational tables: A case against common beliefs’, *SIGMOD Record* **31**(2).
- Cardelli, L. and Wegner, P. (1985), ‘On understanding types, data abstraction, and polymorphism’, *ACM Computing Surveys* **17**, 471–523.
- Ceglar, A. and Roddick, J. F. (2006), ‘Association mining’, *ACM Computing Surveys* **38**(2).
- Chakrabarti, D. and Faloutsos, C. (2006), ‘Graph mining: Laws, generators, and algorithms’, *ACM Computing Surveys* **38**(2).
- Chan, D. and Roddick, J. F. (2006), Local nulls in summarised mobile and distributed databases, *in* ‘Data Mining and Information Engineering 2006: The 7th International Conference on Data, Text and Web Mining and their Business Applications and Management Information Engineering’, Vol. 37 of *WIT Transactions on Information and Communication Technology*, WIT Press, Prague, Czech Republic, pp. 407–416.
- Chankuang, N. and Chittayasothorn, S. (2004), An object and XML database schema design tool, *in* ‘International Conference on Information Technology: Coding and Computing (ITCC 2004)’, Vol. 2, IEEE Computer Society, Las Vegas, Nevada, pp. 421–427.
- Chaudhri, A. B., Rashid, A. and Zicari, R. (2003), *XML Data Management: Native XML and XML-Enabled Database Systems*, Addison-Wesley.

- Chaudhuri, S. and Dayal, U. (1997), ‘An overview of data warehousing and OLAP technology’, *ACM SIGMOD Record* **26**(1), 65–74.
- Chen, C., Song, I.-Y. and Zhu, W. (2007), Trends in conceptual modelling: Citation analysis of the ER conference papers (1979-2005), in ‘The 11th International Conference of the International Society for Scientometrics and Informetrics (ISSI 2007)’, Madrid, Spain, pp. 189–2000.
- Chen, G. and Kerre, E. E. (1998), Extending ER/EER concepts towards fuzzy conceptual data modeling, in ‘Proceedings of the 1998 IEEE International Conference on Fuzzy Systems’, Vol. 2, pp. 1320–1325.
- Chen, G., Ren, M., Yan, P. and Guo, X. (2007), ‘Enriching the ER model based on discovered association rules’, *Information Sciences: An International Journal* **177**(7), 1558–1566.
- Chen, H., Zeng, D., Atabakhsh, H., Wyzga, W. and Schroeder, J. (2003), ‘COPLINK: Managing law enforcement data and knowledge’, *Communications of the ACM* **46**, 28–34.
- Chen, P. P. (1976), ‘The entity-relationship model — toward a unified view of data’, *ACM Transactions on Database Systems* **1**(1), 9–36.
- Chen, P. P. (1977), The entity relationship model — a basis for the enterprise view of data, in ‘Proceedings of National Computer Conference’, AFIPS Press, pp. 77–84.
- Chen, P. P. (2002), ‘Entity-relationship modeling: historical events, future trends, and lessons learned’, *Software Pioneers: Contributions to Software Engineering* pp. 100–114.
- Chen, P. P. (2006), Suggested research directions for a new frontier — active conceptual modeling, in D. W. Embley, A. Olivé and S. Ram, eds, ‘25th International Conference on Conceptual Modeling (ER 2006)’, Vol. 4215 of *Lecture Notes in Computer Science*, Springer, Tucson, Arizona, pp. 1–4.
- Chen, P. P., Thalheim, B. and Wong, L. Y. (1999), Future directions of conceptual modeling, in P. P. Chen, J. Akoka, H. Kangassalo and B. Thalheim, eds, ‘Conceptual Modeling: Current Issues and Future Directions’, Vol. 1565 of *Lecture Notes in Computer Science*, Springer, pp. 287–301.

- Chen, P. P., Wong, L. Y., Delcambre, L., Akoka, J., Sølvsberg, A. and Liuzzi, R. (2008), Research issues in active conceptual modeling of learning: Summary of panel discussion in two workshops (May 2006) and (November 2006), *in* P. P. Chen and L. Y. Wong, eds, 'Active Conceptual Modeling of Learning', Vol. 4512 of *Lecture Notes in Computer Science*, Springer, pp. 217–225.
- Chong, E. I., Das, S., Eadon, G. and Srinivasan, J. (2006), Supporting keyword columns with ontology-based referential constraints in DBMS, *in* 'Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)', IEEE Computer Society, p. 95.
- Codd, E. F. (1970), 'A relational model of data for large shared data banks', *Communications of the ACM* **13**(6), 377–387.
- Codd, E. F. (1979), 'Extending the database relational model to capture more meaning', *ACM Transactions on Database Systems* **4**(4).
- Codd, E. F. (1980), Data models in database management, *in* 'Proceedings of the 1980 Workshop on Data Abstraction, Database and Conceptual Modeling', Colorado, United States, pp. 112–114.
- Codd, E. F. (1986), 'Missing information (applicable and inapplicable) in relational databases', *ACM SIGMOD Record* **15**(4), 53–53.
- Combi, C. and Oliboni, B. (2006), Conceptual modeling of XML data, *in* 'Proceedings of the 2006 ACM Symposium on Applied Computing', ACM, Dijon, France, pp. 467–473.
- Connolly, T. M. and Begg, C. E. (2004), *Database Systems: A Practical Approach to Design, Implementation, and Management*, 4 edn, Addison-Wesley.
- Corcho, O., Fernández-López, M. and Gómez-Pérez, A. (2003), 'Methodologies, tools and languages for building ontologies. Where is their meeting point?', *Data & Knowledge Engineering* **46**(1), 41–64.
- Creasy, P. (1989), ENIAM: A more complete conceptual schema language, *in* 'Proceedings of the 15th International Conference on Very Large Data Bases', Morgan Kaufmann, Amsterdam, The Netherlands, pp. 104 – 114.
- Cui, Z. and O'Brien, P. (2000), Domain ontology management environment, *in* 'Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS 2000)', Vol. 8, IEEE Computer Society Press, Maui, Hawaii, p. 8015.

- Dahchour, M., Pirotte, A. and Zimányi, E. (2005), Generic relationships in information modeling, *in* S. Spaccapiera, ed., ‘Journal on Data Semantics IV’, Vol. 3730 of *Lecture Notes in Computer Science*, Springer, pp. 1–34.
- Dameron, O., Noy, N. F., Knublauch, H. and Musen, M. A. (2004), Accessing and manipulating ontologies using web services, *in* ‘the 3rd International Conference on the Semantic Web (ISWC-2004)’, Hiroshima, Japan.
- Danielsen, A. (1998), ‘The evolution of data models and approaches to persistence in database systems’, (http://www.fing.edu.uy/inco/grupos/csi/esp/Cursos/cursos_act/2000/DAP_DisAvDB/documentacion/OO/Evol_DataModels.html), accessed November 5, 2008.
- Das, S., Chong, E. I., Eadon, G. and Srinivasan, J. (2004), Supporting ontology-based semantic matching in RDBMS, *in* ‘Proceedings of the 30th Very Large Data Bases (VLDB) Conference’, Toronto Canada, pp. 1054–1065.
- Date, C. J. (2003), *An introduction to Database Systems*, 8th edn, Addison-Wesley.
- Date, C. J. and Darwen, H. (1992), *Relational Database Writings, 1989-1991*, Addison-Wesley.
- Date, C. J., Darwen, H. and Lorentzos, N. A. (2003), *Temporal Data and the Relational Model: A detailed Investigation into the Application of Interval and Relation Theory to the Problem of Temporal Database Management*, Morgan Kaufmann.
- Davies, J., Studer, R. and Warren, P. (2006), *Semantic Web Technologies: Trends and Research in Ontology-Based Systems*, John Wiley & Sons.
- DBTG (1971), Report of the CODASYL data base task group, Technical report, ACM.
- de Tré, G., de Caluwe, R. and Prade, H. (2004), Null values revisited in prospect of data integration, *in* M. Bouzeghoub, C. A. Goble, V. Kashyap and S. Spaccapiera, eds, ‘1st International IFIP Conference on Semantics of a Networked World: ICSNW 2004’, Vol. 3226 of *Lecture Notes in Computer Science*, Springer, Paris, France, pp. 79–90.
- De Troyer, O. (1989), RIDL*: A tool for the computer-assisted engineering of large databases in the presence of integrity constraints, *in* J. Clifford, B. G.

- Lindsay and D. Maier, eds, 'Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data', ACM Press, Portland, Oregon, pp. 418–429.
- De Troyer, O., Casteleyn, S. and Plessers, P. (2005), Using ORM to model web systems, *in* 'On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops', Vol. 3762 of *Lecture Notes in Computer Science*, Springer, Agia Napa, Cyprus, pp. 700–709.
- de Vries, D. (2006), MESODATA: Engineering domains for attribute evolution and data integration, PhD thesis, Flinders University.
- de Vries, D., Rice, S. and Roddick, J. F. (2004), In support of mesodata in database management systems, *in* F. Galindo, M. Takizawa and R. Traunmüller, eds, '15th International Conference on Database and Expert Systems Applications (DEXA 2004)', Vol. 3180 of *Lecture Notes in Computer Science*, Springer, Zaragoza, Spain, pp. 663–674.
- de Vries, D. and Roddick, J. F. (2004), Facilitating database attribute domain evolution using mesodata, *in* S. Wang, K. Tanaka, S. Zhou, T. W. Ling, J. Guan, D. Yang, F. Grandi, E. Mangina, I.-Y. Song and H. C. Mayr, eds, 'ER 2004 Workshops on Conceptual Modeling for Advanced Application Domains', Vol. 3289 of *Lecture Notes in Computer Science*, Springer, Shanghai, China, pp. 429–440.
- de Vries, D. and Roddick, J. F. (2007), 'The case for mesodata: An empirical investigation of an evolving database system', *Information and Software Technology* **49**(9-10), 1061–1072.
- DeMers, M. N. (2005), *Fundamentals of Geographic Information Systems*, John Wiley.
- Demey, J., Jarrar, M. and Meersman, R. (2002), A markup language for ORM business rules, *in* M. Schroeder and G. Wagner, eds, 'Proceedings of the International Workshop on Rule Markup Languages for Business Rules on the Semantic Web (RuleML 2002)', Vol. 60, CEUR-WS.org, Sardinia, Italy.
- Demo, G. B., Di Leva, A. and Giolito, P. (1985), An entity-relationship query language, *in* 'Proceedings of the IFIP WG 8.1 Working Conference on Information Systems — Theoretical and Formal Aspects (TFAIS' 85)', Sitges, Spain, pp. 19–32.

- Demuth, B. and Hussmann, H. (1999), Using UML/OCL constraints for relational database design, *in* ‘Proceedings of the 2nd International Conference on the Unified Modeling Language’, Vol. 1723 of *Lecture Notes in Computer Science*, Springer, Fort Collins, Colorado, USA, pp. 598–613.
- Dey, D., Storey, V. C. and Barron, T. M. (1999), ‘Improving database design through the analysis of relationships’, *ACM Transactions on Database Systems* **24**(4), 453–486.
- Dumpala, S. R. and Arora, S. K. (1981), Schema translation using the entity-relationship approach, *in* P. P. Chen, ed., ‘Proceedings of the 2nd International Conference on the Entity-Relationship Approach to Information Modeling and Analysis’, North-Holland, pp. 337–356.
- El-Ghalayini, H., Odeh, M. and McClatchey, R. (2006), Engineering conceptual data models from domain ontologies: A critical evaluation, *in* ‘International Conference on Computer Science and Information Technology (CSIT’06)’, Amman, Jordan.
- Elangovan, K. (2006), *GIS: Fundamentals, Applications and Implementations*, 1st edn, New India Publishing Agency.
- Elmasri, R., El-Assal, I. and Kouramajian, V. (1990), Semantics of temporal data in an extended ER model, *in* H. Kangassalo, ed., ‘Proceedings of the 9th International Conference on the Entity-Relationship Approach (ER’90)’, ER Institute, Lausanne, Switzerland, pp. 249–264.
- Elmasri, R. and Kouramajian, V. (1993), A temporal query language for a conceptual model, *in* N. R. Adam and B. K. Bhargava, eds, ‘Advanced database systems’, Vol. 759 of *Lecture Notes in Computer Science*, Springer, pp. 175–195.
- Elmasri, R. and Larson, J. A. (1985), A graphical query facility for ER databases, *in* P. P. Chen, ed., ‘The Use of ER Concept in Knowledge Representation, Proceedings of the 4th International Conference on the Entity-Relationship Approach’, IEEE Computer Society and North-Holland, pp. 236–245.
- Elmasri, R. and Navathe, S. B. (1994), *Fundamentals of Database Systems*, Benjamin/Cummings.
- Elmasri, R. and Navathe, S. B. (2007), *Fundamentals of Database Systems*, 5th edn, Pearson/Addison Wesley, Boston.

- Elmasri, R., Weeldreyer, J. A. and Hevner, A. R. (1985), ‘The category concept: An extension to the entity-relationship model’, *Data & Knowledge Engineering* **1**(1), 75–116.
- Elmasri, R. and Wiederhold, G. (1981), GORDAS: A formal high-level query language for the entity-relationship model, *in* ‘Proceedings of the 2nd International Conference on the Entity-Relationship Approach (ER’ 81)’, pp. 49–72.
- Elmasri, R. and Wu, G. T. J. (1990), A temporal model and query language for ER databases, *in* ‘Proceedings of the 6th International conference on Data Engineering’, pp. 76–83.
- Elmasri, R., Wu, G. T. J. and Kouramajian, V. (1993), A temporal model and query language for EER databases, *in* ‘Temporal Databases: Theory, Design, and Implementation’, Database System and Applications Series, Benjamin/Cummings, pp. 219–229.
- Embley, D. W. (2004), Towards semantic understanding — an approach based on information extraction ontologies, *in* K. D. Schewe and H. E. Williams, eds, ‘Proceedings of the 15th Australasian Database Conference (ADC 2004)’, Vol. 27 of *Conferences in Research and Practice in Information Technology*, ACS, Dunedin, New Zealand.
- Embley, D. W. and Ling, T. W. (1989), Synergistic database design with an extended entity-relationship model, *in* F. H. Lochovsky, ed., ‘Proceedings of the 8th International Conference on the Entity-Relationship Approach’, North-Holland, Toronto, Canada, pp. 111–128.
- Embley, D. W., Olivé, A. and Ram, S. (2006), Preface, *in* D. W. Embley, A. Olivé and S. Ram, eds, ‘Conceptual Modeling — ER 2006’, Vol. 4215 of *Lecture Notes in Computer Science*, Springer, Tucson, Arizona, USA.
- Engels, G., Gogolla, M., Hohenstein, U., Hülsmann, K., Löhr-Richter, P., Saake, G. and Ehrich, H.-D. (1992), ‘Conceptual modelling of database applications using extended ER model’, *Data & Knowledge Engineering* **9**, 157–204.
- Esculier, C. and Friesen, O. (1995), Conceptual modeling: A critical survey and a few perspectives, *in* ‘Conference Proceedings of the 1995 IEEE 14th Annual International Phoenix Conference on Computers and Communications, 1995’, Scottsdale, Arizona, USA, pp. 319–325.

- Evermann, J. and Wand, Y. (2005), 'Ontology based object-oriented domain modelling: fundamental concepts', *Requirements Engineering Journal* **10**(2), 146–160.
- Fahrner, C. and Vossen, G. (1995), 'A survey of database design transformations based on the entity-relationship model', *Data & Knowledge Engineering* **15**(3), 213–250.
- Falkenberg, E. D. (1976), Concepts for modelling information, in G. M. Nijssen, ed., 'Proceedings of the IFIP Working Conference on Modelling in Data Base Management Systems', North-Holland Publishing, pp. 95–409.
- Falkenberg, E. D. (1993), DETERM: A deterministic event-tuned entity-relationship modeling, in R. A. Elmasri, V. Kouramajian and B. Thalheim, eds, 'Proceedings of the 12th International Conference on the Entity-Relationship Approach', Vol. 823 of *Lecture Notes in Computer Science*, Springer-Verlag, Arlington, Texas, USA, pp. 230–241.
- Farquhar, A., Fikes, R. and Rice, J. (1997), 'The ontolingua server: A tool for collaborative ontology construction', *International Journal of Human and Computer Studies* **46**, 707–728.
- Fellbaum, C. (1998), *WordNet: An Electronic Lexical Database*, MIT Press.
- Fensel, D. (2001), *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*, Springer.
- Ferg, S. (1985), Modeling the time dimension in an entity-relationship diagram, in P. P. Chen, ed., 'The Use of ER Concept in Knowledge Representation, Proceedings of the 4th International Conference on the Entity-Relationship Approach', IEEE Computer Society and North-Holland, Chicago, Illinois, USA, pp. 280–286.
- Ferg, S. (1991), Cardinality constraints in entity-relationship modeling, in T. J. Teorey, ed., 'Proceedings of the 10th International Conference on the Entity-Relationship Approach (ER'91)', ER Institute, San Mateo, California, USA, pp. 1–30.
- Feyer, T. and Thalheim, B. (1999), E/R based scenario modeling for rapid prototyping of web information services, in 'Advances in Conceptual Modeling', Vol. 1727 of *Lecture Notes in Computer Science*, Springer, pp. 253–263.

- Flory, A. and Giard, V. (1988), Modelling requirements of a manufacturing design application using an E/R schema, *in* S. T. March, ed., ‘Entity-Relationship Approach — ER’ 87’, North-Holland, Amsterdam, pp. 249–267.
- Fonseca, F., Davis, C. and Câmara, G. (2003), ‘Bridging ontologies and conceptual schemas in geographic information integration’, *Geoinformatica* **7**(4), 355–378.
- Fonseca, F. and Martin, J. (2007), ‘Learning the differences between ontologies and conceptual schemas through ontology-driven information systems’, *Journal of the Association for Information Systems — Special Issue on Ontologies in the Context of IS* **8**(2), 129–142.
- Franconi, E. and Sattler, U. (1999), A data warehouse conceptual data model for multidimension aggregation, *in* S. Gatzju, M. A. Jeusfeld, M. Staudt and Y. Vassiliou, eds, ‘Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW’99)’, Heidelberg, Germany, pp. 13.1–13.10.
- Friis-Christensen, A., Tryfona, N. and Jensen, C. S. (2001), Requirements and research issues in geographic data modeling, *in* ‘Proceedings of the 9th ACM international symposium on advances in geographic information systems’, ACM, Atlanta, Georgia, USA, pp. 2–8.
- Fromm, K. R., Polikoff, I., Obrst, L., Daconta, M. C., Murphy, R. and Morrison, J.-H. (2005), Introducing semantic technologies and the vision of the semantic web, Technical Report White Paper Series Module 1, SICO P: Semantic Interoperability Community of Practice.
- Gailly, F. and Poels, G. (2007), Towards ontology-driven information systems: Redesign and formalization of the REA ontology, Technical report, Ghent University.
- Galindo, J. (2008), *Handbook of Research on Fuzzy Information Processing in Databases*, Information Science Reference, Hershey, Pennsylvania, USA.
- Galindo, J., Urrutia, A. and Piattini, M. (2006), *Fuzzy Databases: Modeling, Design and Implementation*, Idea Group Inc (IGI).
- Gangemi, A., Guarino, N., Masolo, C., Oltramari, A. and Schneider, L. (2002), Sweetening ontologies with DOLCE, *in* ‘Proceedings of the 13th Interna-

- tional Conference on Knowledge Engineering and Knowledge Management (EKAW02)', Vol. 2473 of *Lecture Notes in Computer Science*, Sigüenza, Spain.
- Garzotto, F., Mainetti, L. and Paolini, P. (1994), HDM2: Extending the E-R approach to hypermedia application design, *in* R. A. Elmasri, V. Kouramajian and B. Thalheim, eds, 'Entity-Relationship Approach — ER' 93', Vol. 823 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 178–189.
- Garzotto, F., Paolini, P. and Schwabe, D. (1993), 'HDM: A model based approach to hypermedia application design', *ACM Transactions on Office Information Systems* **11**(1), 1–26.
- Gemis, M., Paredaens, J., Thyssens, I. and den Bussche, J. V. (1993), GOOD: A graph-oriented object database system, *in* 'Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data', ACM Press, pp. 505–510.
- Genesereth, M. R. (1991), Knowledge interchange format, *in* 'Proceedings of the 2nd International Conference on the Principles of Knowledge Representation and Reasoning (KR-91)', Morgan Kaufman, pp. 238–249.
- Génova, G., Llorens, J. and Martinez, P. (2001), Semantics of the minimum multiplicity in ternary associations in UML, *in* 'Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools', Vol. 2185 of *Lecture Notes in Computer Science*, Springer, pp. 329–341.
- Gessert, G. H. (1990), 'Four valued logic for relational database systems', *SIGMOD Record* **19**(1), 29–35.
- Gogolla, M. and Hohenstein, U. (1991), 'Towards a semantic view of an extended entity-relationship model', *ACM Transactions on Database Systems* **16**(3), 369–416.
- Golfarelli, M., Maio, D. and Rizzi, S. (1998), The dimensional fact model: A conceptual model for data warehouses, *in* 'International Journal of Cooperative Information Systems (IJCIS)', Vol. 7, pp. 215–247.
- Gómez-Pérez, A., Fernández-López, M. and Corcho, O. (2004), *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web*, 1st edn, Springer, London.

- Gorman, K. and Choobineh, J. (1991), ‘The object-oriented entity-relationship model (OOERM)’, *Journal of Management Information Systems* **7**(3), 41–65.
- Greenspan, S. J., Borgida, A. and Mylopoulos, J. (1986), ‘A requirement modeling language and its logic’, *Information Systems* **11**(1), 9–23.
- Gregersen, H. (2005), *TimeERplus: A temporal EER model supporting schema changes*, in M. Jackson, D. Nelson and S. Stirk, eds, ‘Database: Enterprise, Skills and Innovation’, Vol. 3567 of *Lecture Notes in Computer Science*, Springer, pp. 41–59.
- Gregersen, H. and Jensen, C. S. (1998), *Conceptual modeling of time-varying information*, Technical Report TR-35, TimeCenter.
- Gregersen, H. and Jensen, C. S. (1999), ‘Temporal entity-relationship models — a survey’, *IEEE Transactions on Knowledge and Data Engineering* **11**(3), 464–497.
- Gregersen, H. and Jensen, C. S. (2004), *Conceptual modeling of time-varying information*, in ‘Proceedings of International Conference on Computing, Communications and Control Technologies’, pp. 248–255.
- Gregersen, H., Mark, L. and Jansen, C. S. (1998), *Mapping temporal ER diagrams to relational schemas*, Technical Report TR39, TimeCenter.
- Griffiths, T., Fernandes, A. A. A., Paton, N. W. and Barr, R. (2004), ‘The tripod spatio-historical data model’, *Data & Knowledge Engineering* **49**(1), 23–65.
- Grinev, M. N. and Kuznetsov, S. D. (2002), ‘UQL: A UML-based query language for integrated data’, *Programming and Computing Software* **28**(4), 189–196.
- Gruber, T. R. (1993), ‘A translation approach to portable ontology specification’, *Knowledge Acquisition* **5**, 199–220.
- Guarino, N. (1998), *Formal ontology and information systems*, in ‘Proceedings Conference on Formal Ontology (FOIS’ 98)’, Trento, Italy.
- Guizzardi, G. (2005), *Ontological Foundations for Structural Conceptual Models*, Telematica Instituut Fundamental Research Series No. 15, The Netherlands.
- Guizzardi, G. (2006), *The role of foundational ontology for conceptual modeling and domain ontology representation*, in ‘Proceedings of the 7th International Baltic Conference on Database and Information Systems’, Vilnius, Lithuania, pp. 17–25.

- Guizzardi, G., Herre, H. and Wagner, G. (2002), On the general ontological foundations of conceptual modeling, *in* S. Spaccapietra, S. T. March and Y. Kambayashi, eds, 'Proceedings of the 21st International Conference on Conceptual Modeling — ER 2002', Vol. 2503 of *Lecture Notes in Computer Science*, Springer, Tampere, Finland, pp. 65–78.
- Gütting, R. H. (1994), GraphDB: Modeling and querying graphs in databases, *in* 'Proceedings of the 20th International Conference on Very Large Data Base (VLDB)', Morgan Kaufmann, pp. 297–308.
- Gyssens, M. and Lakshmanan, L. V. S. (1997), A foundation for multi-dimensional databases, *in* 'Proceedings of the International Conference on Very Large Database (VLDB'97)', Athens, Greece, pp. 106–115.
- Hadzilacos, T. and Tryfona, N. (1997), 'An extended entity-relationship model for geographic applications', *ACM SIGMOD Record* **26**(3), 24–29.
- Halper, M., Liu, L., Geller, J. and Perl, Y. (2003), 'Frameworks for incorporating semantic relationships into object-oriented database systems', *Concurrency and Computation: Practice and Experience* **15**(15), 1337–1362.
- Halpin, T. A. (1993), What is an elementary fact?, *in* G. Nijssen and J. S. Utrecht, eds, 'Proceedings of First NIAM-ISDM Conference'.
- Halpin, T. A. (1998), Object-role modeling (ORM/NIAM), *in* P. Bernus, K. Mertins and G. Schmidt, eds, 'Handbook on Architectures of Information Systems.', Springer, Berlin, pp. 81–101.
- Halpin, T. A. (1998-1999a), 'UML data models from an ORM perspective: Parts 1-10', *Journal of Conceptual Modeling* .
- Halpin, T. A. (1999b), 'UML data models from an ORM perspective: Parts 1-10', *Journal of Conceptual Modeling, InConcept* **8**.
- Halpin, T. A. (2000), Modeling collections in UML and ORM, *in* 'Proceedings of the 5th IFIP WG8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD'00)', Sweden.
- Halpin, T. A. (2001), *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*, Morgan Kaufmann.
- Halpin, T. A. (2004), 'Comparing metamodels for ER, ORM and UML data models', *Advanced Topics in Database Research* **3**, 23–44.

- Halpin, T. A. and Bloesch, A. (1999), ‘Data modeling in UML and ORM: a comparison’, *Journal of Database Management* **10**(4), 4–13.
- Halpin, T. A. and Morgan, T. (2008), *Information Modeling and Relational Databases*, 2nd edn, Morgan Kaufmann.
- Halpin, T. A. and Proper, H. A. (1995), Database schema transformation and optimization, in M. P. Papazoglou, ed., ‘Proceedings of the 14th International Conference on Object-Oriented and Entity-Relationship Modelling (OOER’95)’, Vol. 1021 of *Lecture Notes in Computer Science*, Springer, Gold Coast, Australia, pp. 191–203.
- Halpin, T. A. and Wagner, G. (2003), Modeling reactive behavior in ORM, in I.-Y. Song, S. W. Liddle, T. W. Ling and P. Scheuermann, eds, ‘International Conference on Conceptual Modeling — ER 2003’, Vol. 2813 of *Lecture Notes in Computer Science*, Springer, Chicago, Illinois, USA, pp. 567–569.
- Hammer, M. and McLeod, D. (1981), ‘Database description with SDM: A semantic database model’, *ACM Transactions on Database Systems* **6**(3), 351–386.
- Han, J.-W. and Li, Z.-N. (1992), ‘Deductive-ER: Deductive entity-relationship data model and its data language’, *Information and Software Technology* **34**(3), 192–204.
- Han, L. and Qing-zhong, L. (2004), ‘Ontology based resolution of semantic conflicts in information integration’, *Wuhan University Journal of Natural Sciences* **9**(5), 606–610.
- Hartmann, S. (2003), Reasoning about participation constraints and Chen’s constraints, in K.-D. Schewe and X. Zhou, eds, ‘Proceedings of the 14th Australasian Database Conference (ADC2003)’, Vol. 17, ACS, Adelaide, Australia, pp. 105–113.
- Henricksen, K., Indulska, J. and McFadden, T. (2005), Modelling context information with ORM, in R. Meersman, Z. Tari and P. Herrero, eds, ‘OTM Workshops 2005’, Vol. 3762 of *Lecture Notes in Computer Science*, Springer, Agia Napa, Cyprus, pp. 626–635.
- Hohenstein, U. and Engels, G. (1992), ‘SQL/EER — syntax and semantics of an entity-relationship-based query language’, *Information Systems* **17**(3), 209–242.

- Howard, P. (2008), ‘The importance of a common data model’, (<http://www.it-director.com/technology/applications/content.php?cid=10292>), accessed November 25, 2008.
- Hull, R. and King, R. (1987), ‘Semantic database modelling: survey, application, and research issues’, *ACM Computing Surveys* **19**(3), 201–260.
- Imhoff, C., Gallemmo, N. and Geiger, J. G. (2003), *Mastering Data Warehouse Design: Relational and Dimensional Techniques*, John Wiley and Sons.
- Imieliński, T. and Lipski, W. (1984), ‘Incomplete information in relational databases’, *Journal of the ACM* **31**(4), 761–791.
- Jarrar, M., Demey, J. and Meersman, R. (2003), ‘On using conceptual data modeling for ontology engineering’, *Journal on Data Semantics* **2800**, 185–207.
- Jensen, C. S., Kligys, A., Pedersen Bach, T. and Timko, I. (2004), ‘Multidimensional data modeling for location-based services’, *The International Journal on Very Large Data Bases* **13**(1), 1–21.
- Jensen, C. S. and Snodgrass, R. T. (1996), ‘Semantics of time-varying information’, *Information Systems* **21**(4), 311–352.
- Jensen, C. S. and Snodgrass, R. T. (2000), Temporally enhanced database design, in M. P. Papazoglou, S. Spaccapietra and Z. Tari, eds, ‘Advances in Object-Oriented Data Modeling’, MIT Press, pp. 163–193.
- Jewett, T. (2002-2006*b*), ‘Database design with UML and SQL’, (<http://www.tomjewett.com/dbdesign/dbdesign.php?page=intro.html>), accessed November 19, 2008.
- Jewett, T. (2006*a*), ‘Database design — subclasses’, (<http://www.tomjewett.com/dbdesign/dbdesign.php?page=subclass.php>), accessed June 20, 2008.
- Jiménez, L. G. (2006), ‘REERM: Reenhancing the entity-relationship model’, *Data & Knowledge Engineering* **58**(3), 410–435.
- Kamble, A. S. (2008), A conceptual model for multidimensional data, in A. Hinze and M. Kirchberg, eds, ‘Proceedings of the 5th Asia-Pacific Conference on Conceptual Modelling (APCCM 2008)’, Vol. 79 of *Conferences in Research and Practice in Information Technology*, Wollongong, Australia, pp. 29–38.

- Kangassalo, H., Jaakkola, H., Ohsuga, S. and Wangler, B. (1995), *Information Modelling and Knowledge Bases VI*, IOS Press.
- Kappel, G. and Schrefl, M. (1988), A behavior-integrated entity-relationship approach for the design of object-oriented databases, *in* C. Batini, ed., 'Proceedings of the 7th International Conference on the Entity-Relationship Approach', North-Holland, Rome, Italy, pp. 311–328.
- Karlapalem, K., Dani, A. R. and Krishna, P. R. (2001), A frame work for modeling electronic contracts, Vol. 2224 of *Lecture Notes in Computer Science*, Springer, pp. 193–207.
- Kent, W. (1984), 'Fact-based data analysis and design', *Journal of Systems and Software* **4**(2-3), 99–121.
- Kerschberg, L., Baum, R. and Hung, J. (1990), KORTEx: An expert database system shell for a knowledge-based entity relationship model, *in* F. H. Lochovsky, ed., 'Entity-Relationship-Approach to Database Design and Querying', North-Holland, pp. 255–268.
- Kerschberg, L., Klug, A. C. and Tsichritzis, D. (1976), A taxonomy of data models, *in* 'Proceedings of the 2nd International Conference on Systems for Large Data Bases', North Holland and IFIP, pp. 43–64.
- Kerschberg, L. and Weishar, D. J. (2000), 'Conceptual models and architectures for advanced information systems', *Applied Intelligence* **13**, 149–164.
- Khatri, V., Ram, S. and Snodgrass, R. T. (2006), 'On augmenting database design-support environments to capture the geo-spatio-temporal data semantics', *Information Systems* **31**(2), 98–133.
- Kim, W. (1990), 'Object-oriented databases: Definition and research directions', *IEEE Transactions on Knowledge and Data Engineering* **2**(3), 327–341.
- Kim, W. and Seo, J. (1991), 'Classifying schematic and data heterogeneity in multidatabase systems', *Computer* **24**(12), 12–18.
- Kim, Y.-G. and March, S. T. (1995), 'Comparing data modeling formalisms', *Communications of the ACM* **38**(6), 103–115.
- King, R. and McLeod, D. (1985), 'A database design methodology and tool for information systems', *ACM Transactions on Office Information Systems* **3**(1), 2–21.

- Kline, N. (1993), 'An update of the temporal database bibliography', *ACM SIGMOD Record* **22**(4), 66–80.
- Klopprogge, M. R. (1981), TERM: An approach to include the time dimension in the entity-relationship model, *in* P. P. Chen, ed., 'Proceedings of the 2nd International Conference on the Entity-Relationship Approach (ER'81)', North-Holland, pp. 477–512.
- Klopprogge, M. R. and Lockeman, P. C. (1983), Modeling information preserving databases: Consequences of the concept of time, *in* 'Proceedings of the 9th International Conference on Very Large Data Bases', North-Holland, pp. 399–416.
- Klyne, G. and Carroll, J. J. (2004), 'Resource description framework (RDF): Concepts and abstract syntax', (<http://www.w3.org/TR/rdf-concepts/#section-data-model>), accessed November 10, 2008.
- Kroenke, D. M. and Gray, C. D. (2006), 'Toward a next generation data modeling facility: Neither the entity-relationship model nor UML meet the need', *Journal of Information Systems Education* **17**(1), 29–37.
- Krogstie, J., Opdahl, A. L. and Brinkkemper, S. (2007), *Conceptual Modelling in Information Systems Engineering*, Springer.
- Kunii, H. S. (1987), DBMS with graph data model for knowledge handling, *in* 'Proceedings of the 1987 Fall Joint Computer Conference on Exploring Technology: Today and Tomorrow.', IEEE Computer Society Press, pp. 138–142.
- La-Ongsri, S., Roddick, J. F. and de Vries, D. (2008), 'Accommodating mesodata into conceptual modelling methodologies', *Information and Software Technology* **50**(5), 424–435.
- Lacroix, M. and Pirotte, A. (1976), 'Generalized joins', *SIGMOD Record* **8**(3), 14–15.
- Laender, A. H. and Flynn, D. J. (1993), A semantic comparison of the modelling capabilities of the ER and NIAM models, *in* R. A. Elmasri, V. Kouramajian and B. Thalheim, eds, 'Proceedings of the 12th International Conference on the Entity-Relationship Approach', Vol. 823 of *Lecture Notes in Computer Science*, Springer, Arlington, Texas, USA, pp. 242–256.
- Lai, V. S., Kuiboer, J.-P. and Guynes, J. L. (1994), 'Temporal databases: Model design and commercialization prospects', *DATA BASE* **25**(3), 6–18.

- Lassila, O. and McGuinness, D. L. (2001), The role of frame-based representation on the semantic web, Technical Report KSL-01-02, Knowledge Systems Laboratory, Stanford University.
- Lechtenbörger, J. and Vossen, G. (2003), ‘Multidimensional normal forms for data warehouse design’, *Information Systems* **28**(5), 415–434.
- Leelawatananon, T. and Chittayasothorn, S. (2004), The ORM model as a knowledge representation for e-tutorial systems, in ‘Proceedings of the 6th International Conference on Enterprise Information Systems, ICEIS 2004’, Porto, Portugal, pp. 479–484.
- Lenat, D. B. (1995), ‘CYC: A large-scale investment in knowledge infrastructure’, *Communications of the ACM* **38**(11).
- Leung, C. M. R. and Nijssen, G. M. (1987), ‘From a NIAM conceptual schema into the optimal SQL relational database schema’, *The Australian Computer Journal* **19**(2), 69–75.
- Leung, C. M. R. and Nijssen, G. M. (1988), ‘Relational database design using the NIAM conceptual schema’, *Information Systems* **13**(2), 219–227.
- Levene, M. and Loizou, G. (1995), ‘A graph-based data model and its ramifications’, *IEEE Transactions on Knowledge and Data Engineering* **7**(5), 809–823.
- Longley, P., Goodchild, M. F., Maguire, D. J. and Rhind, D. W. (2005), *Geographic Information Systems and Science*, 2nd edn, John Wiley and Sons.
- Lósio, B. F., Salgado, A. C. and Galvão, L. d. R. (2003), Conceptual modeling of XML schemas, in ‘Proceedings of the 5th ACM International Workshop on Web Information and Data Management’, ACM, pp. 102–105.
- Lu, Y., Yang, Q. and Liu, H. (2006), XML conceptual modeling with XUML, in ‘Proceedings of the 28th International Conference on Software Engineering (ICSE’06)’, pp. 973–976.
- Luján-Mora, S., Trujillo, J. and Song, I.-Y. (2006), ‘A UML profile for multidimensional modeling in data warehouses’, *Data & Knowledge Engineering* **59**(3), 725–769.
- Lukovic, I., Ristic, S. and Mogin, P. (2003), On the formal specification of database schema constraints, in ‘Proceedings of the 1st Serbian-Hungarian

- Joint Symposium on Intelligent Systems (SISY 2003)', Subotica, Serbia and Montenegro, pp. 125–136.
- Ma, Z. (2005), *Fuzzy Database Modeling with XML*, Springer, New York.
- Ma, Z. (2006), *Fuzzy Database Modeling of Imprecise and Uncertain Engineering Information*, Springer.
- Ma, Z. M., Ma, W. Y. and Zhang, W. (2000), An extended conceptual model for fuzzy data modeling, *in* 'Proceedings of the 1st International Conference on Web Information Systems Engineering (WISE'00)', Vol. 2, IEEE Computer Society, pp. 75–80.
- Ma, Z. M. and Yan, L. (2008), 'A literature overview of fuzzy database models', *Journal of Information Science and Engineering* **24**, 189–202.
- Ma, Z. M., Zhang, W. J., Ma, W. Y. and Chen, G. Q. (2001), 'Conceptual design of fuzzy object-oriented databases using extended entity-relationship model', *International Journal of Intelligent Systems* **16**(6), 697–711.
- MacFarlane, A., McCann, J. and Liddell, H. (1996), A common data model for meta-data in interoperable environments, *in* 'Proceedings of the 2nd International Baltic Workshop on Databases and Information Systems', Vol. 1, Institute of Cybernetics, Tallinn, Estonia, pp. 75–86.
- Maier, D. (1983), *The Theory of Relational Databases*, Computer Science Press, Rockville.
- Malinowski, E. and Zimányi, E. (2006), 'Hierarchies in a multidimensional model: From conceptual modeling to logical representation', *Data & Knowledge Engineering* **59**(2), 348–377.
- Malinowski, E. and Zimanyi, E. (2008), *Advanced Data Warehouse Design: From Conventional to Spatial and Temporal Applications*, 1st edn, Springer.
- Mani, M. (2004), EReX: A conceptual model for XML, *in* 'Database and XML Technologies', Vol. 3186 of *Lecture Notes in Computer Science*, Springer, pp. 128–142.
- Martyn, T. (2004), 'Reconsidering multi-dimensional schemas', *ACM SIGMOD Record* **33**(1), 83–88.

- McAllister, A. (1998), ‘Complete rules for n-ary relationship cardinality constraints’, *Data & Knowledge Engineering* **27**(3), 255–288.
- McBrien, P., Seltveit, A. H. and Wangler, B. (1992), An entity-relationship model extended to describe historical information, *in* ‘International Conference on Information Systems and Management of Data’, pp. 244–260.
- McGee, W. C. (1976), ‘On user criteria for data model evaluation’, *ACM Transactions on Database Systems* **1**(4), 370–387.
- McGuinness, D. L. (2003), Ontologies come of age, *in* D. Fensel, J. Hendler, H. Lieberman and W. Wahlster, eds, ‘Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential’, MIT Press, pp. 171–195.
- McKenzie, E. (1986), ‘Bibliography: Temporal databases’, *ACM SIGMOD Record* **15**(4), 40–52.
- Meersman, R. (1996), An essay on the role and evolution of data(base) semantics, Technical Report STAR-1996-01, STAR Lab, Department of Computer Science, Vrije Universiteit Brussel.
- Meersman, R. (1999), The use of lexicons and other computer-linguistic tools in semantics, design and cooperation of database systems, *in* ‘Proceedings of Conference on on Cooperative Database Systems (CODAS99)’, Springer, pp. 1–14.
- Mena, E. and Illarramendi, A. (2001), *Ontology-Based Query Processing for Global Information Systems*, The Kluwer international series in engineering and computer Science, Kluwer Academic Publishers Group.
- Mindswap (2008), ‘National Cancer Institute thesaurus’, (<http://www.mindswap.org/2003/CancerOntology>), accessed January 9, 2008.
- Motro, A. (1988), ‘VAGUE: A user interface to relational databases that permits vague queries’, *ACM Transactions on Database Systems* **6**(3), 187–214.
- Moyne, J. R., Teorey, T. J. and Leo C. McAfee, J. (1991), ‘Time sequence ordering extensions to the entity-relationship model and their application to the automated manufacturing process’, *Data & Knowledge Engineering* **6**, 421–443.
- Muller, R. J. (1999), *Database Design for Smarties: Using UML for Data Modeling*, Morgan Kaufmann.

- Murthy, S., Delcambre, L. and Maier, D. (2006), Explicitly representing superimposed information in a conceptual model, *in* D. W. Embley, A. Olivé and S. Ram, eds, ‘Conceptual Modeling — ER 2006’, Vol. 4215 of *Lecture Notes in Computer Science*, Springer, pp. 126–139.
- Mylopoulos, J. (1992), Conceptual modeling and Telos, *in* P. Loucopoulos and R. Zicari, eds, ‘Conceptual Modelling, Databases and CASE: An Integrated View of Informations Systems Development’, McGraw Hill.
- Mylopoulos, J. (2004), ‘Data semantics revisited: Databases and the semantic web’, (<http://aitrc.kaist.ac.kr/dasfaa04/doc/JM-DASFAA.pdf>), accessed May 8, 2008.
- Narasimhalu, A. (1988), A data model for object-oriented databases with temporal attributes and relationships, Technical report, National University of Singapore.
- Navathe, S. B. (1992), ‘Evolution of data modeling for databases’, *Communications of the ACM* **35**(9), 112–123.
- Navathe, S. and Pillalamarri, M. (1988), Towards making the ER approach object-oriented, *in* C. Batini, ed., ‘Proceedings of the 7th International Conference on the Entity-Relationship Approach’, North-Holland, Rome, Italy.
- Necip, C. B. and Freytag, J.-C. (2003), Ontology based query processing in database management systems, *in* ‘On the Move to Meaningful Internet Systems 2003: CoopIS, DOA and ODBASE’, Vol. 2888 of *Lecture Notes in Computer Science*, Springer, pp. 839–857.
- Necip, C. B. and Freytag, J.-C. (2005), Semantic query transformation using ontologies, *in* ‘Proceedings of the 9th International Database Applications and Engineering Symposium (IDEAS’05)’, Montreal, Canada.
- Nečaský, M. (2006), Conceptual modeling for XML: A survey, *in* ‘Proceedings of the Annual International Workshop on Databases, Texts, Specifications and Objects (DATESO 2006)’, Vol. 176, pp. 40–53.
- Nečaský, M. (2007), XSEM — a conceptual model for XML, *in* J. F. Roddick and A. Hinze, eds, ‘Proceedings of the 4th Asia-Pacific Conference on Conceptual Modelling (APCCM 2007)’, Vol. 67 of *Conferences in Research and Practice in Information Technology*, Ballarat, Victoria, Australia, pp. 37–48.

- Nijssen, G. M. (1976), A gross architecture for the next generation database management systems, *in* G. M. Nijssen, ed., ‘Proceedings of the IFIP Working Conference on Modelling in Data Base Management Systems’, North-Holland, Freudenstadt, Germany, pp. 1–24.
- Nijssen, G. M. (1977), Current issues in conceptual schema concepts, *in* G. Nijssen, ed., ‘Proceedings of the IFIP Working Conference on Modelling in Data Base Management Systems’, North-Holland, Nice, France, pp. 31–66.
- Nijssen, G. M. (1985), On experience with large-scale teaching and use of fact-based conceptual schemas in industry and university, *in* T. B. Steel Jr and R. Meersman, eds, ‘Proceedings of the IFIP WG 2.6 Working Conference on Data Semantics (DS-1)’, North-Holland, Hasselt, Belgium, pp. 189–204.
- Nijssen, G. M. and Halpin, T. A. (1989), *Conceptual Schema and Relational Database Design: A Fact Oriented Approach*, Prentice Hall.
- Niles, I. and Pease, A. (2001), Towards a standard upper ontology, *in* ‘Proceedings of Formal Ontology in Information Systems (FOIS 2001)’, Maine, USA, pp. 2–9.
- Noy, N. F. and McGuinness, D. L. (2001), Ontology development 101: A guide to creating your first ontology, Technical Report KSL-01-05, SMI-2001-0880, Stanford Knowledge Systems Laboratory and Stanford Medical Informatics.
- Obrst, L. (2003), Ontologies for semantically interoperable systems, *in* ‘Proceedings of the 12th International Conference on Information and Knowledge Management’, ACM, New Orleans, Louisiana, USA, pp. 366–369.
- Oh, Y.-C. and Navathe, S. B. (1995), SEER: Security enhanced entity-relationship model for secure relational databases, *in* ‘OOER’95: Object-Oriented and Entity-Relationship Modeling’, Vol. 1021 of *Lecture Notes in Computer Science*, pp. 170–180.
- Ohsuga, S., Kangassalo, H., Jaakkola, H., Hori, K. and Yonezaki, N. (1992), *Information Modelling and Knowledge Bases*, IOS Press.
- OMG (2005), ‘Unified modeling language specification, version 1.4.2’, (<http://www.omg.org/docs/formal/05-04-01.pdf>), accessed November 21, 2005.

- OMG (2008), ‘Introduction to OMG’s unified modeling languages (UML)’, (http://www.omg.org/gettingstarted/what_is_uml.htm), accessed June 18, 2008.
- Ott, T. and Swiaczny, F. (2001), *Time-Integrative Geographic Information Systems: Management and Analysis of Spatio-Temporal Data*, 1st edn, Springer.
- Ou, Y. (1998), ‘On mapping between UML and entity-relationship model’, *The Unified Modeling Language: Technical aspects and applications* pp. 45–57.
- OwlSeek.com (2008), ‘What is ontology?’, (<http://www.owlseek.com/what-is.html>), accessed January 10, 2008.
- Özsoyoğlu, G. and Snodgrass, R. T. (1995), ‘Temporal and real-time databases: A survey’, *IEEE Transactions on Knowledge and Data Engineering* **7**(4), 513–532.
- Parent, C., Spaccapietra, S. and Zimányi, E. (1999), Spatio-temporal conceptual models: Data structures + space + time, *in* ‘Proceedings of the 7th ACM International Symposium on Advances in Geographic Information Systems’, ACM, Kansas City, Missouri, USA, pp. 26–33.
- Parent, C., Spaccapietra, S. and Zimányi, E. (2006a), *Conceptual Modeling for Traditional and Spatio-Temporal Applications, The MADS Approach*, Springer.
- Parent, C., Spaccapietra, S. and Zimányi, E. (2006b), ‘The MurMur project: Modeling and querying multi-representation spatio-temporal databases’, *Information Systems* **31**(8), 733–769.
- Parné, R. C. (2002), ‘From ternary relationship to relational tables: A case against common beliefs’, *SIGMOD Record* **31**(2), 46–49.
- Patig, S. (2006), ‘Evolution of entity-relationship modelling’, *Data & Knowledge Engineering* **56**(2), 122–138.
- Peckham, J. and Maryanski, F. (1988), ‘Semantics data models’, *ACM Computing Surveys* **20**(3), 153–189.
- Pelekis, N., Theodoulidis, B., Kopanakis, L. and Theodoridis, Y. (2004), ‘Literature review of spatio-temporal database models’, *The Knowledge Engineering Review* **19**(3), 235–274.

- Pernul, G., Winiwarter, W. and Tjoa, A. M. (1993), The entity-relationship model for multilevel security, *in* R. A. Elmasri, V. Kouramajian and B. Thalheim, eds, 'Proceedings of the 12th International Conference on the Entity-Relationship Approach', Vol. 823 of *Lecture Notes in Computer Science*, Springer-Verlag, Arlington, Texas, USA, pp. 166–177.
- Pierson, E. J. and Cruz, N. d. (2005), Using object role modeling for effective in-house decision support systems, *in* 'On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops', Vol. 3762 of *Lecture Notes in Computer Science*, Springer, Agia Napa, Cyprus, pp. 636–645.
- Pornphol, P. and Chittayasothorn, S. (2004), A temporal relational and object relational database design technique, *in* 'Proceedings of the 2004 IEEE SoutheastCon Conference', pp. 54–59.
- Potter, W. and Kerschberg, L. (1988), 'A unified approach to modeling knowledge and data', *Data and Knowledge (DS-2)* pp. 265–292.
- Price, R., Ramamohanarao, K. and Srinivasan, B. (1999), Spatiotemporal extensions to Unified Modeling Language, *in* 'Proceedings of the 10th International Workshop on Database and Expert Systems Applications', IEEE Computer Society, Florence, Italy, pp. 460–461.
- Price, R., Tryfona, N. and Jensen, C. S. (2000), 'Extended spatiotemporal UML: Motivations, requirements and constructs', *Journal of Database Management* **11**(4), 14–27.
- Proper, H. A., Hoppenbrouwers, S. and van der Weide, T. P. (2005), A fact-oriented approach to activity modeling, *in* 'On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops', Vol. 3762 of *Lecture Notes in Computer Science*, Springer, Agia Napa, Cyprus, pp. 666–675.
- Protégé (2008), 'The protégé ontology editor and knowledge acquisition system', (<http://protege.stanford.edu>), accessed January 15, 2008.
- Psaila, G. (2000), ERX: A conceptual model for XML documents, *in* 'Proceedings of the 2000 ACM symposium on Applied Computing', Vol. 2, ACM, pp. 898–903.
- Puntheeranurak, S. and Chittayasothorn, S. (2002), An extended NIAM conceptual schema model for object databases, *in* 'Proceedings of the 24th International Conference on Information Technology Interfaces', pp. 57–61.

- Purao, S. and Storey, V. C. (2005), 'A multi-layered ontology for comparing relationship semantics in conceptual models of databases', *Applied Ontology* **1**(1), 117–139.
- Ram, S., Snodgrass, R. T., Khatri, V. and Hwang, Y. (2001), DISTIL: A design support environment for conceptual modeling of spatio-temporal requirements, *in* 'Proceedings of the 20th International Conference on Conceptual Modeling, ER 2001', Vol. 2224 of *Lecture Notes in Computer Science*, Springer, pp. 70–83.
- Raymond, D., Tompa, F. and Wood, D. (1996), 'From data representation to data model: Meta-semantic issues in the evolution of SGML', *Computer Standards & Interfaces* **18**(1), 25–36.
- Reiter, R. (1978), On closed world databases, *in* H. Gallaire and J. Minker, eds, 'Logic and Databases', Plenum Press, New York, pp. 55–76. Reprinted in *Artificial Intelligence and Databases*. J. Mylopoulos and M.L. Brodie (eds.), Morgan Kaufmann, 248–258.
- Rice, S. P., Roddick, J. F. and de Vries, D. (2006), Defining and implementing domains with multiple types using mesodata modelling techniques, *in* M. Stumppner, S. Hartmann and Y. Kiyoki, eds, 'Proceedings of the 3rd Asia-Pacific Conference on Conceptual Modelling', Vol. 53 of *Conferences in Research and Practice in Information Technology*, ACS, Hobart, Australia, pp. 85–93.
- Rizzi, S., Abelló, A., Lechtenböcker, J. and Trujillo, J. (2006), Research in data warehousing modeling and design: Dead or alive?, *in* 'Proceedings of the 9th ACM international workshop on Data warehousing and OLAP', Arlington, Virginia, USA, pp. 3–10.
- Rochfeld, A. and Negros, P. (1992), 'Relationship of relationships and other inter-relationship links in E-R model', *Data & Knowledge Engineering* **9**(2), 205–221.
- Rochfeld, A. and Tardieu, H. (1983), 'MERISE: An information system design and development methodology', *Information and Management* **6**, 143–159.
- Roddick, J. F. (1995), 'A survey of schema versioning issues for database systems', *Information and Software Technology* **37**(7), 383–393.
- Roddick, J. F., Ceglar, A. and de Vries, D. (2007), Towards active conceptual modelling for sudden events, *in* J. Grundy, S. Hartmann, A. H. Laender, L. Maciaszek and J. F. Roddick, eds, 'Proceedings of the 26th International Conference on Conceptual Modeling (ER 2007) (Posters)', Vol. 83 of *Conferences*

- in Research and Practice in Information Technology*, ACS, Auckland, New Zealand, pp. 203–208.
- Roddick, J. F., Ceglar, A., de Vries, D. and La-Ongsri, S. (2008), Postponing schema definition: Low Instance-to-Entity Ratio (LIteER) modelling, *in* P. P. Chen and L. Y. Wong, eds, ‘Active Conceptual Modeling of Learning’, Vol. 4512 of *Lecture Notes in Computer Science*, Springer, pp. 206–216.
- Roddick, J. F., Craske, N. G. and Richards, T. J. (1994), A taxonomy for schema versioning based on the relational and entity-relationship models, *in* R. A. Elmasri, V. Kouramajian and B. Thalheim, eds, ‘Entity-Relationship Approach — ER’ 93’, Vol. 823 of *Lecture Notes in Computer Science*, Springer, pp. 137–148.
- Roddick, J. F., Craske, N. G. and Richards, T. J. (1996), ‘Handling discovered structure in database systems’, *IEEE Transactions on Knowledge and Data Engineering* **8**, 227–240.
- Roddick, J. F. and de Vries, D. (2006), Reduce, reuse, recycle: Practical approaches to schema integration, evolution and versioning, *in* F. Frandi, ed., ‘Proceedings of the 4th International Workshop on Evolution and Change in Data Management’, Vol. 4231, Springer, Tuscon, Arizona, pp. 209–216.
- Roddick, J. F., Hoel, E., Egenhofer, M. J., Papadias, D. and Salzberg, B. (2004), ‘Spatial, temporal and spatio-temporal databases — hot issues and directions for PhD research’, *ACM SIGMOD Record* **33**(2).
- Roddick, J. F., Hornsby, K. and Spiliopoulou, M. (2001), An updated bibliography of temporal, spatial, and spatio-temporal data mining research, *in* ‘Temporal, Spatial, and Spatio-Temporal Data Mining: First International Workshop, TSDM 2000’, Vol. 2007 of *Lecture Notes in Artificial Intelligence*, Springer, Lyon, France.
- Roddick, J. F. and Patrick, J. D. (1992), ‘Temporal semantics in information systems — a survey’, *Information Systems* **17**(3), 249–267.
- Roddick, J. F. and Spiliopoulou, M. (2002), ‘A survey of temporal knowledge discovery paradigms and methods’, *IEEE Transactions on Knowledge and Data Engineering* **14**(4), 750–767.
- Rolland, C. and Cauvet, C. (1992), Trends and perspectives in conceptual modelling, *in* P. Loucopoulos and R. Zicari, eds, ‘Conceptual Modelling, Databases

- and CASE: an Integrated View of Information Systems Development.’, John Wiley.
- Roth, M. A., Korth, H. F. and Silberschatz, A. (1989), ‘Null values in nested relational databases’, *Acta Informatica* **26**(7), 615–642.
- Routledge, N., Bird, L. and Goodchild, A. (2002), UML and XML schema, *in* ‘Proceedings of the 13th Australasian Database Conference (ADC2002)’, Vol. 5 of *Conferences in Research and Practice in Information Technology*, ACS, Melbourne, Australia, pp. 157–166.
- Saiedian, H. (1997), ‘An evaluation of extended entity-relationship model’, *Information and Software Technology* **39**(7), 449–462.
- Sapia, C., Blaschka, M., Höfling, G. and Dinter, B. (1998), Extending the E/R model for the multidimensional paradigm, *in* Y. Kambayashi, D. Lee, E. P. Lim, M. Mohania and Y. Masunaga, eds, ‘Proceedings of the ER Workshop on Data Warehousing and Data Mining’, Vol. 1552 of *Lecture Notes in Computer Science*, Springer, Singapore, pp. 105–116.
- Schrefl, M. (1991), Behavior modeling by stepwise refining behavior diagrams, *in* H. Kangassalo, ed., ‘Entity-Relationship Approach: The Core of Conceptual Modelling’, North-Holland, pp. 119–134.
- Sengupta, A. and Mohan, S. (2003), Formal and conceptual models for XML structures — the past, present, and future, Technical Report 137-1, Indiana University.
- Sengupta, A., Mohan, S. and Doshi, R. (2003), XER — extensible entity relationship modeling, *in* ‘Proceedings of the XML 2003 Conference’, Philadelphia, Pennsylvania, USA, pp. 140–154.
- Sengupta, A. and Wilde, E. (2006), The case for conceptual modeling for XML, Technical Report 242, Computer Engineering and Networks Laboratory, ETH Zürich.
- Senko, M. E. (1975), ‘Information systems: Records, relations, sets, entities and things’, *Information Systems* **1**(1), 3–13.
- Senko, M. E. (1976), NIAM as a detailed example of the ANSI SPARC architecture, *in* G. M. Nijssen, ed., ‘Modelling in Data Base Management Systems’, North-Holland, pp. 73–94.

- Shah, D. and Slaughter, S. (2003), Transforming UML class diagrams into relational data models, *in* ‘UML and the unified process’, Idea Group, pp. 217–236.
- Shekhar, S. and Chawla, S. (2002), *Spatial Databases: A Tour*, Prentice Hall.
- Shekhar, S., Vatsavai, R. R., Chawla, S. and Burk, T. E. (1999), Spatial pictogram enhanced conceptual data models and their translation to logical data models, *in* P. Agouris and A. Stefanidis, eds, ‘The International Workshop on Integrated Spatial Databases, Digital Images and GIS’, Vol. 1737 of *Lecture Notes in Computer Science*, Springer, Portland, Maine, USA, pp. 77–104.
- Sheth, A., Meersman, R. and Navathe, S. B. (1995), Data semantics: What, where and how, *in* ‘Proceedings of the 6th IFIP working Conference on Data Semantics (DS-6)’, pp. 601–610.
- Sheth, A., Thacker, S. and Patel, S. (2003), ‘Complex relationships and knowledge discovery support in the InfoQuilt system’, *The International Journal on Very Large Data Bases* **12**(1), 2–27.
- Shimazu, K., Momma, A. and Furukawa, K. (2003), DAISY, an RER model based interface for RDB to ILP, *in* I.-Y. Song, S. W. Liddle, T. W. Ling and P. Scheuermann, eds, ‘Conceptual Modeling — ER 2003’, Vol. 2813 of *Lecture Notes in Computer Science*, Springer, pp. 390–404.
- Shipman, D. W. (1981), ‘The functional data model and the data language DAPLEX’, *ACM Transactions on Database Systems* **6**(1), 140–173.
- Shoshani, A. and Wong, H. K. T. (1985), ‘Statistical and scientific database issues’, *IEEE Transactions on Software Engineering* **11**, 1040–1047.
- Silberschatz, A., Korth, H. F. and Sudarshan, S. (1996), ‘Data models’, *ACM Computing Surveys* **28**(1), 105–108.
- Silberschatz, A., Korth, H. F. and Sudarshan, S. (2005), *Database System Concepts*, 5th edn, McGraw-Hill.
- Simsion, G. C. and Witt, G. C. (2005), *Data Modeling Essentials*, 3rd edn, Morgan Kaufmann.
- Smith, B. (2003), Ontology, *in* L. Floridi, ed., ‘The Blackwell Guide to the Philosophy of Computing and Information’, Wiley-Blackwell, pp. 155–166.

- Smith, B. and Welty, C. (2001), Ontology: towards a new synthesis, in 'Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS 2001)', ACM Press, pp. 3–9.
- Snodgrass, R. T. (1987), 'The temporal query language TQuel', *ACM Transactions on Database Systems* **12**(2), 247–298.
- Snodgrass, R. T. (2000), *Developing Time-Oriented Database Applications in SQL*, Morgan Kaufmann, San Francisco, California.
- Socket, G. H., Burns, L. M., Malhotra, A. and Whang, K.-Y. (1993), 'GRAQULA: A graphical query language for entity-relationship or relational databases', *Data & Knowledge Engineering* **11**(2), 171–202.
- Song, I.-Y. and Forbes, E. A. (1991), Schema conversion rules between EER and NIAM, in 'Proceedings of the 10th International Conference on Entity-Relationship Approach', San Mateo, California.
- Song, I.-Y. and Jones, T. H. (1993), Analysis of binary relationships within ternary relationships in ER modeling, in R. Elmasri, V. Kouramajian and B. Thalheim, eds, 'International Conference on the Entity-Relationship Approach', Vol. 823 of *Lecture Notes in Computer Science*, Springer, Arlington, Texas, USA, pp. 271–282.
- Soo, M. D. (1991), 'Bibliography on temporal databases', *ACM SIGMOD Record* **20**(1), 14–23.
- Sowa, J. F. (2005), The challenge of knowledge soup, in J. Ramadas and S. Chunnawala, eds, 'Research Trends in Science, Technology and Mathematics Education', Homi Bhabha Centre, Mumbai, pp. 55–90.
- Spaccapietra, S. (2008), 'Ontologies', (<http://lbd.epfl.ch/e/teaching/SlidesST/Ontologies.pdf>), accessed January 12, 2008.
- Spaccapietra, S., March, S. T. and Kambayashi, Y. (2002), Foreword, in S. Spaccapietra, S. T. March and Y. Kambayashi, eds, 'Conceptual Modeling — ER 2002', Vol. 2503 of *Lecture Notes in Computer Science*, Springer, Tampere, Finland.
- Spaccapietra, S. and Parent, C. (1992), ERC+: An object-based entity relationship approach, in P. Loucopoulos and R. Zicari, eds, 'Conceptual Modelling, Databases and CASE: An Integrated View of Information Systems Development', John Wiley, pp. 69–86.

- Spaccapietra, S., Parent, C., Vangenot, C. and Cullot, N. (2004), On using conceptual modeling for ontologies, *in* ‘Proceedings of the Web Information Systems Workshops (WISE 2004 workshops)’, Lecture Notes in Computer Science, Springer, pp. 22–33.
- Spaccapietra, S., Parent, C. and Zimányi, E. (2008), Spatio-temporal and multi-representation modeling: A contribution to active conceptual modeling, *in* P. P. Chen and L. Y. Wong, eds, ‘Active Conceptual Modeling of Learning’, Vol. 4512 of *Lecture Notes in Computer Science*, Springer, pp. 197–205.
- Spencer, J. (2001), *The Strange Logic of Random Graphs*, 1st edn, Springer.
- Spyns, P., Meersman, R. and Jarrar, M. (2002), ‘Data modelling versus ontology engineering’, *ACM SIGMOD Record* **31**(4), 12–17.
- Stam, R. B. and Snodgrass, R. T. (1988), ‘A bibliography on temporal databases’, *Data Engineering Bulletin* **11**(4), 53–61.
- Stevens, R., Goble, C. A. and Bechhofer, S. (2000), ‘Ontology-based knowledge representation for bioinformatics’, *Briefings in Bioinformatics* **1**(4), 398–414.
- Stonebraker, M., Brown, P. and Moore, D. (1999), *Object Relational DBMSs: Tracking the Next Grave Wave*, 2nd edn, Morgan Kaufmann.
- Stonebraker, M. and Hellerstein, J. M. (2005), What goes around comes around, *in* ‘Readings in Database Systems’, 4th edn, Morgan Kaufmann.
- Stonebraker, M., Rowe, L. A., Lindsay, B. G., Gray, J., Carey, M. J., Brodie, M. L., Bernstein, P. A. and Beech, D. (1990), ‘Third-generation database system manifesto’, *ACM SIGMOD Record* **19**(3), 31–44.
- Storey, V. C. (1991), ‘Relational database design based on the entity-relationship model’, *Data & Knowledge Engineering* **7**, 47–83.
- Storey, V. C. (1993), ‘Understanding semantic relationships’, *The International Journal on Very Large Data Bases* **2**(4), 455–488.
- Storey, V. C. (2005), ‘Comparing relationships in conceptual modeling: Mapping to semantic classifications’, *IEEE Transactions on Knowledge and Data Engineering* **17**(11), 1478–1489.

- Storey, V. C., Goldstein, R. C., Chiang, R. H. L. and Dey, D. (1994), A common-sense reasoning facility based on the entity-relationship model, *in* R. A. Elmasri, V. Kouramajian and B. Thalheim, eds, 'Entity-Relationship Approach — ER '93', Vol. 823 of *Lecture Notes in Computer Science*, Springer, pp. 218–241.
- Studer, R., Benjamins, V. R. and Fensel, D. (1998), 'Knowledge engineering: Principles and methods', *Data & Knowledge Engineering* **25**(1-2), 161–197.
- Sugumaran, V. and Storey, V. C. (2002), 'Ontologies for conceptual modeling: Their creation, use, and management', *Data & Knowledge Engineering* **42**, 251–271.
- Sugumaran, V. and Storey, V. C. (2006), 'The role of domain ontologies in database design: An ontology management and conceptual modeling environment', *ACM Transactions on Database Systems* **31**(3), 1064–1094.
- Sure, Y. (2003), Methodology, tools and case studies for ontology based knowledge management, PhD thesis, University of Karlsruhe.
- Tauzovich, B. (1991), Towards temporal extensions to the entity-relationship model, *in* T. J. Teorey, ed., 'Proceedings of the 10th International Conference on the Entity-Relationship Approach (ER'91)', ER Institute, San Mateo, California, USA, pp. 163–179.
- Teorey, T. J. (1990), *Database Modeling and Design: The Entity-Relationship Approach*, Morgan Kaufmann.
- Teorey, T. J., Lightstone, S. and Nadeau, T. (2006), *Database Modeling & Design: Logical Design*, 4 edn, Morgan Kaufmann, San Francisco.
- Teorey, T. J., Yang, D. and Fry, J. P. (1986), 'A logical design methodology for relational databases using the extended entity-relationship model', *Computing Surveys* **18**(2), 197–222.
- ter Hofstede, A. H. M., Lippe, E. and van der Weide, T. P. (1995), A categorical framework for conceptual data modeling: Definition, application, and implementation, Technical Report CSI-R9512, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands.
- Thalheim, B. (1990), Extending the entity-relationship model for a high-level, theory-based database design, *in* J. W. Schmidt and A. A. Stogny, eds, 'Next

- Generation Information System Technology: First International East/West Data Base Workshop', Vol. 504 of *Lecture Notes in Computer Science*, Springer, Kiev, USSR, pp. 161–184.
- Thalheim, B. (2000), *Entity-Relationship Modeling: Foundations of Database Technology*, Springer, Berlin.
- Theodoulidis, C. I. and Loucopoulos, P. (1991), 'The time dimension in conceptual modeling', *Information Systems* **16**(3), 273–300.
- Theodoulidis, C. I., Loucopoulos, P. and Wangler, B. (1991a), 'A conceptual modelling formalism for temporal database applications', *Information Systems* **16**(4), 401–416.
- Theodoulidis, C. I., Loucopoulos, P. and Wangler, B. (1991b), The entity-relationship time model and the conceptual rule language, in 'Proceedings of the 10th International Conference on Entity-Relationship Approach (ER'91)', ER Institute, pp. 181–204.
- Tryfona, N., Busborg, F. and Christiansen, J. G. B. (1999), starER: A conceptual model for data warehouse design, in 'Proceedings of ACM 2nd International Workshop on Data Warehousing and OLAP (DOLAP)', pp. 3–8.
- Tryfona, N. and Jensen, C. S. (1999), 'Conceptual data modeling for spatiotemporal applications', *Geoinformatica* **3**(3), 245–268.
- Tsichritzis, D. and Klug, A. C. (1978), 'The ANSI/X3/SPARC DBMS framework report of the study group on database management systems', *Information Systems* **3**(3), 173–191.
- Tsotras, V. J. and Kumar, A. (1996), 'Temporal database bibliography update', *ACM SIGMOD Record* **25**(1), 41–51.
- Tu, S. Y. and Wang, R. Y. (1993), Modeling data quality and context through extension of the ER model, in 'Workshop on information technology and systems (WITS' 93)'.
- Ullrich, H., Purao, S. and Storey, V. C. (2000), An ontology for classifying the semantics of relationships in database design, in 'Proceedings of the 5th International Conference on Applications of Natural Language to Information Systems-Revised Papers (NLDB'00)', Springer-Verlag, London, UK.

- Valo, A., Hyvönen, A. and Komulainen, V. (2005), A tool for collaborative ontology development for the semantic web, *in* 'Proceedings of International Conference on Dublin Core and Metadata Applications (DC-2005)', Madrid, Spain, pp. 209–212.
- Vassiliadis, P. and Sellis, T. (1999), 'A survey of logical models for OLAP databases', *SIGMOD Record* **28**(4), 64–69.
- Vassiliou, Y. (1979), Null values in data base management: A denotational semantics approach, *in* 'Proceedings of the ACM SIGMOD International Conference on the Management of Data', ACM Press, Boston, Massachusetts, pp. 162–169.
- Velez, F. (1985), LAMBDA: An entity-relationship based query language for the retrieval of structured documents, *in* P. P. Chen, ed., 'Entity-Relationship Approach: The Use of ER Concept in Knowledge Representation', North-Holland, pp. 82–89.
- Verheijen, G. M. A. and van Bekkum, J. (1982), NIAM: An information analysis method, *in* 'Proceedings of the IFIP WG 8.1 Working Conference on Comparative Review of Information Systems Design Methodologies', North Holland, Netherlands, pp. 537–590.
- Vert, G., Stock, M. and Morris, A. (2002), 'Extending ERD modeling notation to fuzzy management of GIS data files', *Data & Knowledge Engineering* **40**(2), 163–179.
- Vrdoljak, B., Banek, M. and Rizzi, S. (2003), Designing web warehouses from XML schemas, *in* 'International conference on data warehousing and knowledge discovery: DaWak 2003', Vol. 2737, pp. 89–98.
- W3C (2008), 'SPARQL Query Language for RDF', (<http://www.w3.org/TR/rdf-sparql-query>), accessed November 18, 2008.
- Wand, Y., Storey, V. C. and Weber, R. (1999), 'An ontological analysis of the relationship construct in conceptual modeling', *ACM Transactions on Database Systems* **24**(4), 494–528.
- Wang, X., Zhou, X. and Lu, S. (2000), Spatiotemporal data modeling and management: A survey, *in* 'Proceedings of the 36th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS-Asia'00)', pp. 202–211.

- Weber, R. and Zhang, Y. (1996), 'An analytical evaluation of NIAM's grammar for conceptual schema diagrams', *Information Systems* **6**, 147–170.
- Winslett, M. (2004), 'Peter Chen speaks out on paths to fame, the roots of the ER model in human language, the ER model in software engineering, the need for ER databases, and more', *SIGMOD Record* **33**(1), 110–118.
- Woods, W. A. (1988), What's in a link: Foundations for semantics networks, *in* A. Collins and E. E. Smith, eds, 'Readings in Cognitive Science', Morgan Kaufmann, San Mateo.
- WrongDiagnosis.com (2008), 'Types of immune disorders', (<http://www.wrongdiagnosis.com/i/immune/subtypes.htm>), accessed January 9, 2008.
- Wu, Y., Jajodia, S. and Wan, X. S. (1998), Temporal database bibliography update, *in* O. Etzion, S. Jajodia and S. Sripada, eds, 'Temporal Databases: Research and Practice', Vol. 1399 of *Lecture Notes in Computer Science*, Springer, pp. 338–366.
- Yazici, A., George, R., Buckles, B. P. and Petry, F. E. (1992), A survey of conceptual and logical data models for uncertainty management, *in* 'Fuzzy Logic for the Management of Uncertainty', John Wiley & Sons, Inc., pp. 607–643.
- Yue, K.-B. (1991), 'A more general model for handling missing information in relational databases using a 3-valued logic', *SIGMOD Record* **20**(3), 43–49.
- Yuliana, O. Y. and Chittayasothorn, S. (2005), XML schema re-engineering using a conceptual schema approach, *in* 'International Symposium on Information Systems: Coding and Computing (ITCC 2005)', Vol. 1, IEEE Computer Society, pp. 255–260.
- Zaniolo, C. (1984), 'Database relations with null values', *Journal of Computer and System Sciences* **28**(1), 142–166.
- Zhang, Z.-Q. and Mendelzon, A. O. (1983), A graphical query language for entity-relationship databases, *in* 'Proceedings of the 3rd International Conference on the Entity-Relationship Approach', pp. 441–448.
- Zimanyi, E., Parent, C., Spaccapietra, S. and Pirrotte, A. (1997), TERC+: A temporal conceptual model, *in* 'Proceedings of the International Symposium on Digital Media Information Base (DMIB'97)', Nara, Japan.

- Zvieli, A. and Chen, P. P. (1986), Entity-relationship modeling and fuzzy databases, *in* 'Proceedings of the 2nd International Conference on Data Engineering', Los Angeles, California, USA, pp. 320–327.