AN EXTENSIBLE FRAMEWORK FOR GENERATING ONTOLOGY FROM VARIOUS DATA MODELS

by

Khalid Albarrak
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

_____	Dr. Edgar H. Sibley, Dissertation Director
_____	Dr. Alexander Brodsky, Committee Member
_____	Dr. David A. Schum, Committee Member
_____	Dr. Kathryn B Laskey, Committee Member
_____	Dr. Mihai Boicu, Committee Member
_____	Dr. Stephen Nash, Senior Associate Dean
_____	Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering
Date:_____	Summer Semester 2013 George Mason University Fairfax, VA

An Extensible Framework for Generating Ontology from Various Data Models

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Khalid Albarrak
Master of Science in Computer Science
California State University, 2000

Director: Edgar H. Sibley, University Professor & Eminent Scholar
Department of Computer Science

Summer Semester 2013
George Mason University
Fairfax, VA

DEDICATION

To my parents, wife and two sons, Riyadh and Faisal. They all have been very supportive, encouraging, and most of all, exceptionally patient throughout the program.

ACKNOWLEDGEMENTS

I would like to express my profound gratitude and appreciation to my advisor, Professor Edgar Sibley, for his patience and invaluable and continuous support throughout the dissertation. Prof. Sibley's continued encouragement and guidance is unparalleled. I would also like to thank SACM for the partial sponsorship.

TABLE OF CONTENTS

Page

LIST OF TABLES VIII

LIST OF FIGURES IX

ABSTRACT X

CHAPTER 1: INTRODUCTION 1

1.1. BACKGROUND 2
1.2. MOTIVATION 6
1.3. PROBLEM STATEMENT 8
1.4. THE RESEARCH CHALLENGE 10
1.5. SUMMARY OF CONTRIBUTIONS 12
1.5.1. Major Contributions 12
1.5.2. Additional Contributions 16
1.6. THESIS STATEMENT 20
1.7. ORGANIZATION OF THE DISSERTATION 21

CHAPTER 2: RELATED WORK 22

2.1. INTRODUCTION 22
2.2. BACKGROUND 23
2.3. THE COMPARISON FRAMEWORK 27
2.4. STATE-OF-THE-ART IN GENERATING ONTOLOGY FROM DATABASE MODELS 30
2.4.1. Stojanovic et al. Approach 31
2.4.2. Buccella et al. Approach 32
2.4.3. Astrova Approach(es) 32
2.4.4. Man Li et al. Approach 34
2.4.5. Relational.OWL Tool 35
2.4.6. RDB2ONT Tool 35
2.4.7. DB2OWL Tool 36
2.4.8. DataMaster Plug-In 37
2.4.9. Yan and Changrui Method 38
2.4.10. Xu and Li Approach 38
2.4.11. Automatic Ontology Generator Tool 39
2.4.12. Lubyte and Tessaris Framework 39
2.4.13. Changjun Hu et al. Method 40
2.4.14. RDBToOnto Tool 41
2.4.15. OWLFROMDB Tool 42

2.4.16. RDOL Approach *43*
2.5. SUMMARY 44
CHAPTER 3: AN EXTENSIBLE FRAMEWORK FOR GENERATING ONTOLOGY MODELS FROM DATA MODELS 47
3.1. INTRODUCTION 47
3.2. DM2ONT ARCHITECTURE 48
3.2.1. DM2ONT Controller *49*
3.2.2. The Source Collector *50*
3.2.3. The Ontology Converter *51*
3.2.4. The Ontology Generator *52*
3.3. DM2ONT FOR RDB/ORDB AND OWL MODELS 52
3.3.1. The RDB/ORDB Collector Component *54*
3.3.2. The OWL Converter Component *68*
3.4. EXAMPLES 72
3.4.1. Example I (RDB Schema - Simple) *72*
3.4.2. Example II (RDB Schema – Bridge Table) *73*
3.4.3. Example III (ORDB Schema) *74*
3.4.4. Example IV (Sparse-Column Values) *77*
3.5. SUMMARY 79
CHAPTER 4: CANDIDATE SYMMETRIC AND CANDIDATE TRANSITIVE BINARY RELATIONS 80
4.1. INTRODUCTION 80
4.2. EXAMPLES OF SYMMETRIC AND TRANSITIVE BINARY RELATIONS 81
4.3. BASIC DEFINITIONS 82
4.4. MOTIVATION 83
4.5. ASSUMPTIONS 84
4.6. IDENTIFYING CANDIDATE SYMMETRIC AND CANDIDATE TRANSITIVE BINARY RELATIONS 85
4.6.1. Formal Definitions *85*
4.6.2. Methods to Identify Candidate Symmetric and Candidate Transitive Binary Relations *101*
4.7. SUMMARY 138
CHAPTER 5: HEURISTIC METHODOLOGY TO MEASURE THE RELATIVE EXPLICITNESS OF ONTOLOGY MODELS 139
5.1. INTRODUCTION 139
5.2. RELATED WORK 141
5.3. DEFINITION OF AN ONTOLOGY MODEL 143
5.4. METHODOLOGY TO MEASURE EXPLICITNESS 146
5.4.1. Cleansing Phase *147*
5.4.2. Matching Phase *147*
5.4.3. Explicitness Computation Phase *149*
5.5. CASE STUDIES 155

5.6. SUMMARY 159
CHAPTER 6: VALIDATION AND RESULTS OF TESTING 160
6.1. INTRODUCTION 160
6.2. BACKGROUND 160
6.3. IMPLEMENTATION OF THE DM2ONT PROTOTYPE 162
6.4. VALIDATION OF DM2ONT USING VARIOUS DATABASES INSTANCES 164
6.5. SYNTACTIC EXAMINATION OF THE GENERATED ONTOLOGY MODEL 167
6.6. FUNCTIONAL VERIFICATION OF DM2ONT 167
6.7. SEMANTIC VALIDATION USING DOMAIN REQUIREMENTS AND COMPARISON WITH AN EXISTING APPROACH 167
6.8. EXPLICITNESS MEASUREMENT AGAINST EXISTING APPROACHES 171
6.9. SUMMARY 176
CHAPTER 7: CONCLUSION AND FUTURE RESEARCH 179
7.1. CONCLUSION 179
7.2. FUTURE RESEARCH 180
APPENDIX A: RESEARCH PUBLICATIONS 182
APPENDIX B: SYMMETRY/TRANSITIVITY - SAMPLE RELATIONS 183
B.1 SAMPLE RELATIONS 183
B.1.1 Pattern 1: *183*
B.1.2 Pattern 2: *184*
B.1.3 Pattern 3: *186*
APPENDIX C: SAMPLE DATABASE SCRIPTS USED IN VALIDATION 187
1. CASE-STUDY ONE: 187
1.1. DDL Script for IBM DB2 Sample RDB Instance: *187*
1.2. DML Script for IBM DB2 Sample RDB Instance: *193*
2. CASE-STUDY TWO: 202
2.1. DDL Script for MS SQL Server Sample RDB Instance: *202*
2.2. DML Script for MS SQL Server Sample RDB Instance: *207*
APPENDIX D: SETTINGS IN DM2ONT AND DATAMASTER 220
1. DM2ONT PROPERTY FILE: 220
2. DATAMASTER SETTINGS: 221
APPENDIX E: DOMAIN REQUIREMENTS AND RECALL/PRECISION 222
1. CASE-STUDY ONE: 222
2. CASE-STUDY TWO: 228
APPENDIX F: EXPLICITNESS MEASUREMENT METHODOLOGY 238
1. CASE-STUDY ONE: 238
2. CASE-STUDY TWO: 239
BIBLIOGRAPHY 243
BIOGRAPHY 255

LIST OF TABLES

Table Page

Table 1: Dissertation Outline.....21
Table 2. General properties for the reviewed methods.....45
Table 3. Type of constructs handled.....46
Table 4. Type of Information Inferred. 46
Table 5. Examples of confidence ratio threshold for different cardinalities. 63
Table 6. Distribution Ratio (DR) for sparse-column values.....66
Table 7: Sample data instances for Person table 77
Table 8: Examples of binary relations along with their properties and cardinality.....82
Table 9: Abstract representation of domain ontology generated by DM2ONT (om_1) 156
Table 10: Abstract representation of domain ontology from DataMaster (om_2) 157
Table 11: Weights assigned to the different types of ontology construct. 157
Table 12: Summary of explicitness calculations for ontology fragment in case-study 1.....158
Table 13 - Characteristics of the sample RDB instances included with major DBMS(s).....165
Table 14 – Various metrics for the semantic validation experiment.....171
Table 15 - Explicitness measurement in case-study 1 (IBM DB2 sample RDB).....174
Table 16 - Explicitness measurement in case-study 2 (MS SQL Server Sample RDB) 175

LIST OF FIGURES

Figure Page

Figure 1: Semi-automated ontology-based system integration. …….7
Figure 2: Vertical and horizontal translations. ……. 25
Figure 3: DM2ONT Framework Architecture..…….49
Figure 4: DM2ONT for translation from RDB/ORDB to OWL …..…….53
Figure 5: OWL Representation for Example I. ……..……73
Figure 6: OWL representation for Example II. ……..……75
Figure 7: OWL representation for Example III. ……..……..76
Figure 8: OWL representation for Example IV. ……..……..78
Figure 9: Identifying candidate symmetric and transitive binary relations - Overall Process ……..……..102
Figure 10: Methodology to measure explicitness between two Ontology models …….146
Figure 11: Earlier validation experiment. ……..……..161
Figure 12: DM2ONT high-level architecture (extensibility view). 163
Figure 13: Semantic validation experiment. 169
Figure 14: Measuring Explicitness between ontology models 172

ABSTRACT

AN EXTENSIBLE FRAMEWORK FOR GENERATING ONTOLOGY FROM VARIOUS DATA MODELS

Khalid Albarrak, Ph.D.

George Mason University, 2013

Dissertation Director: Edgar H. Sibley

In the Information Technology field, Ontology is concerned with the use of formal representation to describe concepts and relationships in a domain of knowledge. Using ontologies, organizations can facilitate processes such as integrating heterogeneous systems, assessing data quality, validating business rules, and discovering hidden facts. Ontology engineering, however, is not a trivial process. Developing ontologies is highly dependent on the availability and knowledge of ontology modelers and domain experts. Moreover, the development process is often lengthy and error-prone.

In this dissertation, I developed an extensible framework for generating ontologies from data models. For this dissertation, the framework is limited to generating ontology from two types of data models: the Relational Database (RDB) and Object-Relational Database (ORDB) models. The framework, however, is extensible to support the generation of ontologies from other types of data models (e.g. XML). The derived ontology is expressed in the OWL Web Ontology Language, a W3C recommendation.

For RDB and ORDB models, my framework extracts information about these models from the metadata maintained by the Database Management System (DBMS), and from the data instances in certain cases. The extracted metadata includes the integrity constraints that are typically maintained by a DBMS (e.g. primary/foreign keys, not-null and unique constraints). In order to obtain more semantics from a data model implementation, the framework also examines data instances to discover some of the semantic gaps found in the metadata. Once extracted, the metadata and data instances are then analyzed to identify classes and their properties, discover explicit and implicit relationships between classes (including potential class hierarchies), and identify restrictions related to properties and relationships. This analysis is based on heuristic database modeling techniques. The analyzed data model is then translated automatically into an OWL ontology that can be reviewed and/or augmented further with more semantics by ontology modelers based on input from domain experts.

The proposed framework has been validated by implementing it as a prototype, and by examining the ontologies it generates from a syntactic and semantic perspective. For the semantic examination, domain requirements were used to compute the recall and precision for the ontologies generated by my framework and that of a similar tool. Moreover, the relative amount of terminological content (which I call the relative *explicitness*) of these ontologies was measured as well using a methodology that I developed in my research. The results showed the ability of my framework to generate ontologies that are closely aligned with the domain.

CHAPTER 1: Introduction

Semantic Computing attempts to address challenges in information integration and information finding through the use of technologies that can derive and utilize the semantics (i.e. meaning) of the information being exchanged [81]. These technologies rely on the use of ontologies to express the semantics of information [76]. Developing ontologies and assessing their content, however, are not trivial tasks. While state-of-the-art ontology engineering includes methods for semi-automating the generation of ontologies from one type of data model or another, current research lacks a unified and extensible framework for generating ontologies that represent as much content about the domain they describe as possible. Furthermore, current ontology evaluation methods do not provide a formal measure for determining the relative amount of content between two ontology models or between an ontology model and a reference ontology (i.e. gold standard); in this context, an ontology model is said to be *more explicit* about the domain than another model if the former contains more relevant[1] axioms. In my research, I not only developed a unified and extensible framework for generating ontologies from different types of data models, but I also devised a heuristic methodology for measuring the relative explicitness of one ontology model in comparison to another.

[1] Generally speaking, relevance of axioms is established using a reference ontology or domain requirements, and by assigning weights to the different types of axioms; more details in Chapter 5.

In the following sections, I provide a brief background on the research area and discuss the motivation behind this research. Next, I state the problem addressed by this research, describe the research challenges, and present my contributions. This is followed by a formulation of the thesis statement and the validation methods undertaken. Lastly, I conclude this chapter by outlining the remainder of my dissertation.

1.1. Background

One of the challenges when searching for and integrating information is attributed to the absence of semantics when exchanging information. By its very nature, information artifacts typically convey different meaning to different people and systems. This issue becomes evident when the information being exchanged is meant to be processed by computer systems without any human intervention.

In natural languages, words and terminologies typically have more than one meaning (e.g. homonym). For instance, in English, the word “chair” can mean a seat, president, professorship, etc. [69]. In general, people are able to recognize the meaning behind words based on the context in which they are used. On the other hand, abstract information (i.e. data) can be misinterpreted by both people and computer systems. For example, a value representing the price of an item may not denote the currency, or may not indicate whether it is the price for acquiring, producing, or selling an item. Without the meaning clearly and explicitly stated, critical errors can occur.

To enable computer systems to process information effectively, the semantics of the information must be represented in a clear and formal manner. Recent advances in

knowledge representation languages and tools hold the potential for addressing the semantics issues encountered in information finding and integration [77][81]. These languages not only allow augmenting information with semantics in a formal manner, but they also enable tools to reason about information and infer new knowledge.

In the field of semantic computing, the semantics of information are addressed using ontologies. The term *Ontology* originated from the Philosophy discipline where it is concerned with the study of existence. In the Information Technology (IT) field, a commonly used definition for *Ontology* -- coined by Gruber [43] -- is "an explicit specification of a conceptualization". By *explicit*, it means the specification of concepts in a domain of knowledge, and the relationships between these concepts, is described in a formal and unambiguous manner. By *specification of conceptualization*, it means identifying the concepts of interest in a domain of knowledge, the properties of these concepts, and the relationships between these concepts. It is worth noting here that while Gruber's definition is widely used, researchers within the IT and other scientific communities have different understanding of what constitute an ontology [44][74].

Depending on the purpose for which they are used, ontologies range from a *controlled vocabulary* (enumeration of unordered terminologies), *taxonomy* (tree-like hierarchies of terminologies), *thesaurus* (graph-like association among terminologies), to *formal ontology* (defines classes, properties, relationships, and possibly axioms) [74][22]. Moreover, different interpretations exist for the term Ontology and whether it refers to metadata only (i.e. T-Box) or both metadata and data instances (i.e. T-Box and A-box). In

my research, I use the terms *ontology* and *ontology model* interchangeably to refer to a *formal ontology* that contains metadata only.

A domain ontology is a formal ontology that describes a specific domain of knowledge. A domain of knowledge is a subject area such as the Human Resources (HR) function of an enterprise, the student admissions function of an academic institution, or the patient care function of a health establishment. Concepts within subject areas vary. In the HR area for instance, an ontology may contain an Employee concept, a Department concept, and a Project concept, among others. An ontology also captures the properties that are typically found within each concept and the relationships between these concepts. Using the same HR example, an Employee concept may have properties or attributes such as Employee ID Number and Employee Name, and relationships to other concepts, such as Department to denote the area in which an employee works, and Project to indicate in which work stream an employee is assigned.

In order to capture the specification of concepts within a domain of knowledge in an explicit manner, an expressive and formal language is necessary. Over the years, many different ontology languages have been proposed and used. These include DAML (DARPA Agent Markup Language) [29], OIL (Ontology Inference Layer) [36], RDF (Resource Description Framework) and RDFS (RDF Schema) [8], and, more recently OWL Web Ontology Language [8][89].

OWL is not only a prominent ontology language and an international standard, but it is also viewed as a key enabler of the W3C Semantic Web vision. The W3C described this vision as “The idea of having data on the Web defined and linked in a way

that it can be used by machines – not just for display purposes – but for automation, integration and reuse of data across various applications, and thus fully harness the power of information semantics" [91]. This vision highlights the fact that data on the Web needs to be structured, not only for people to understand and process, but also for computer programs to process. These computer programs are often referred to as Intelligent Agents. An intelligent agent is an autonomous software component that is designed to monitor an environment and take actions in a proactive and/or reactive manner to achieve its design objectives [94]. For an intelligent agent to be able to process data efficiently and effectively, the data has to be augmented with clear and formal semantics. This is typically achieved using ontology.

Over the past few decades, organizations have invested a great deal of resources to create and maintain data models that support one or more business applications. Although different types of data models exist today -- such as the Relational, Object-Relational, Object-Oriented, and Hierarchical data models -- the main principles in these models remain the same. Generally speaking, a data model captures both the main data elements in a subject area and the characteristics of these data elements. Depending on the type of data model, similar data elements are grouped into data constructs such as entity sets, which is the case in the Relational model [30][72], or classes, which is the case in Object-Relational and Object-Oriented models [60][72]. Moreover, a data model captures the relationships between the data constructs. For example, a relational database model in the HR subject area may contain Employee, Department and Project entity sets with relationship sets similar to those in the ontology example discussed earlier.

1.2. Motivation

There are various semantic computing solutions in which ontology models can have a significant role. Using such models, organizations can address the common and recurring business challenges they typically face such as:

- *Integration of Heterogonous Systems*: With an ontology model describing each system, integration of these systems can be semi-automated based on the semantic mapping and alignment between these models as shown in Figure 1. Currently, integration between systems is performed either manually, based on input from domain experts, or semi-automatically, based on syntactic mapping (i.e. table/class names, or attribute names and domains). With a global/shared ontology (ONT-G in Figure 1), new applications can be developed to leverage data from heterogonous systems by using a common vocabulary. These new applications can rely on the mapping and alignment rules, which are stored in the Mapping & Alignment Repository component, to perform the translation between the common vocabulary and the local vocabulary. Note that Figure 1 shows a design-time view, where the Local Ontology is generated only once (i.e. when a system is to be integrated into the environment). The use of ontology for system and database integration is the subject of various research projects (e.g. see [92] for a survey on Ontology-based integration approaches).

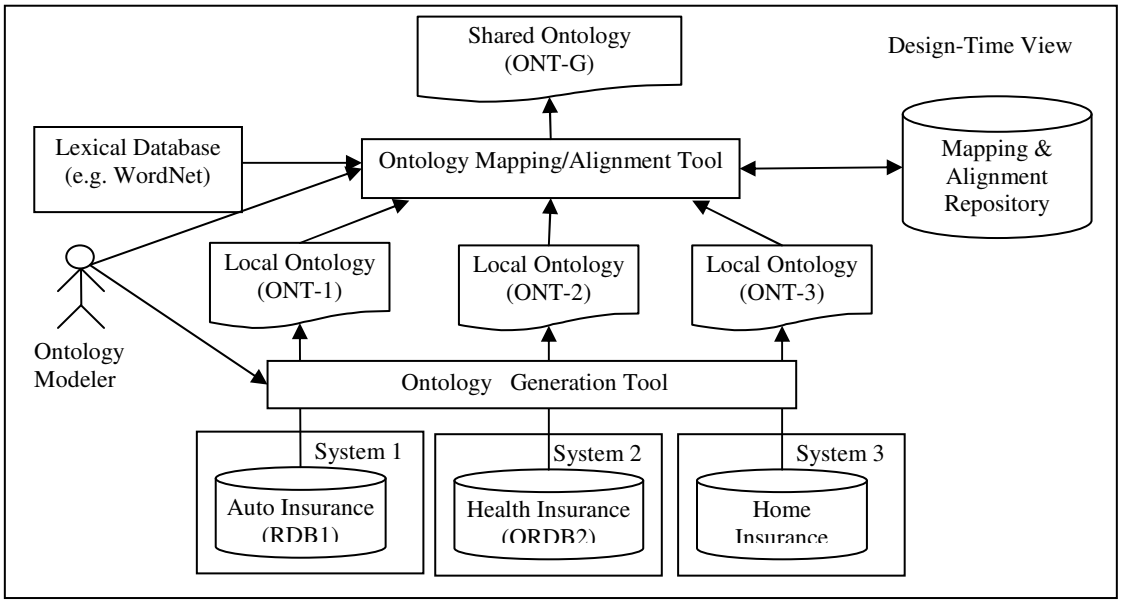


Figure 1: Semi-automated ontology-based system integration

- *Assessment of Data Quality and Validation of Business Rules*: Using ontology and a reasoning engine (a.k.a. Semantic Reasoner and Rule Engine), organizations can use open-standards technologies to asses the quality of their data and validate their business rules. At present, many organizations use proprietary technologies for such purposes.
- *Discovery of New Facts or Knowledge*: Using ontology and an inference engine, organizations can infer or derive new facts from existing and known facts. Inference engines have been in use in Artificial Intelligence (AI) research and applications for many years. An application of such technology can discover hidden and indirect relationships between individuals, objects, etc.

With semantic computing solutions relying on the use of ontologies to provide explicit and formal specification for the domain of knowledge, an acceleration of the development of ontologies can reduce the deployment time for such solutions. This can be achieved by reusing existing assets (e.g. data models), and by automating the process of generating ontologies from these assets rather than developing one from the start.

Lastly, with the relatively recent development and standardization of the ORDB (e.g. SQL:1999 and SQL:2003) [60], organizations that take advantage of ORDB facilities can rely on a single and unified framework to generate ontology from ORDB and RDB models.

1.3. Problem Statement

Developing a new ontology model is not a trivial task. Identifying all of the concepts, properties, relationships, and restrictions in a domain of knowledge requires expertise in the subject area, knowledge in ontology modeling, and skills in an ontology language or ontology editing tool. During the ontology development phase, ontology modelers work closely with domain experts to gain insight into the subject area. Organizations, however, tend to avoid engaging their domain experts in long running endeavors, as they are also expected to perform their primary or day-to-day responsibilities.

Furthermore, over the past few decades, organizations have invested significant resources to create data models that support one or more business applications. Since some of the knowledge about the subject area is already encoded in these data model instances, a practical, inexpensive, and less error-prone approach would be to derive this

knowledge from existing data models. With data models and ontologies capturing similar information about the domain they model but differing in the way they encode it and the level of details they include (e.g. detailed cardinality, symmetry in binary relations, etc.), an effective approach for developing ontology models is to translate existing data models into an equivalent ontology model. The generated ontology can then serve as a rudimentary model for ontology modelers, who can further augment it with more semantics based on consultation with domain experts. In other words, instead of expecting that a domain expert explains the entire subject area, an ontology modeler can use the generated ontology to ask specific and targeted questions about the subject area (e.g. cardinality of specific relationship, restriction on certain property, etc.). Using this approach, the involvement of domain experts can be limited to activities related to refining and validating the generated ontology.

In addition, each of the existing methods for generating ontologies from data models focuses on one data model or another (either RDB or XML). My survey of the existing approaches also revealed that they merely translate constructs found in the source data models into their equivalent ontology construct, without analyzing many aspects of the source data model to recover hidden or lost semantics. To the best of my knowledge, there is not one current approach that generates ontology from ORDB models or from more than one type of data model. Although ORDB and Object-Oriented models have an ontological-like structure (e.g. classes, data members/fields, relationships etc.), a translation method is still needed to convert this ontological-like structure into a valid ontology representation such as OWL, RDF and others. Moreover, none of the existing

approaches performs comprehensive translation from RDB to ontology models. For instance, existing approaches do not identify potential transitive and symmetric relations, discover different types of generalization and specialization relationships (i.e. IS-A), determine the cardinality of relationships, detects sparse-column values, or identify database constraints such as unique and not-null based on both metadata and data analysis. While these characteristics and others can be set manually by ontology modelers, automating their recovery will allow generating a richer ontology and reduce the involvement of domain experts.

Finally, most of the ontology evaluation techniques tend to be subjective and lack any formal measurement for assessing the amount of terminological content (*relative explicitness*) of one ontology model in comparison to another. Given two ontology models – whether they were developed independently for the same domain or different versions of the same ontology – there is often a need to measure which of the two models has more terminological content or is more explicit about the domain. Developing a methodology that accounts for the different types of ontology constructs and computes a formal measurement can instill confidence in the ontology models being evaluated.

1.4. The Research Challenge

The ontology engineering process has been the subject of extensive research, as demonstrated in Chapter 2 of this dissertation. To identify the proper techniques and the gaps in current approaches, I conducted a thorough review of the literature.

Additionally, in order to generate ontology models from different types of data models, one has to attain comprehensive knowledge in Ontology, RDB, and ORDB. This includes a thorough understanding of the semantics of constructs found within these models (i.e. the meta-model), and the modeling guidelines and patterns for each. Moreover, advanced expertise is needed in the query language (i.e. SQL) and the various client interfaces (or Application Programming Interface -- API) available for different DBMS to enable the efficient retrieval of RDB and ORDB information. Although standards exist for SQL and certain client interfaces (e.g. JDBC), deviations from these standards by DBMS vendors have introduced a level of complexity when developing the prototype for the proposed framework. In addition to addressing the RDB and ORDB as source data models, and to allow for extensibility, the framework was designed to take into account other types of source data models.

Lastly, devising a heuristic methodology to measure the relative explicitness between two ontology models is a considerable challenge. Developing this methodology required attaining a comprehensive understanding of the ontology meta-model. Such comprehension was essential for developing a methodology that accounts for various ontology constructs and computes a meaningful measure. Other challenges include employing an abstract ontology representation (to provide the flexibility to compare models that use different ontology representations), cleansing the models to be compared (to eliminate irrelevant axioms), and mapping and aligning constructs across models.

1.5. Summary of Contributions

The prime objective of this research is to develop an *extensible* framework for generating highly *explicit* ontologies (i.e. containing as much terminological content about the domain they describe as possible) from various types of data models. This framework is termed *Data Models to Ontologies (DM2ONT).* Extensibility in DM2ONT allows for the translation from various types of data models into different ontology representations. The target ontology representations (e.g. OWL, RDF) supported by DM2ONT however are limited to the power of DM2ONT internal object representation as described in Section 3.2. Contributions of this research are classified into major contributions and additional contributions as discussed below:

1.5.1. Major Contributions

1. *Methods to generate ontologies that represent substantial terminological content about the domains they describe:* In this dissertation, I developed methods to analyze various aspects of RDB data models to enable recovering the semantics embedded in their implementations. This included retrieving and analyzing both metadata and data instances. The recovered semantics were expressed – as suggestions – in the ontology model generated by DM2ONT. Specifically, the following semantics have been recovered from the source data model and expressed in the generated ontology:
    1.1. *Transitive Binary Relations:* Unlike RDB models, OWL ontologies allow modelers to identify transitive binary relations between concepts. In my research, I identified the types of RDB binary relations that have the *potential*

for being transitive based on design guidelines/patterns and data analysis. The method used to identify such relations is based on heuristic data modeling techniques. Once identified, DM2ONT designated these binary relations as *candidate transitive* in the generated ontology model. Furthermore, since data in databases represent the state of an enterprise at a given point in time (i.e. a snapshot of the system of record), a confidence ratio/factor was calculated and assigned to findings that are based on data analysis. This confidence ratio was computed using a formula I developed, which takes into account the total number of rows supporting the findings. In this dissertation, I formally defined the method to identify candidate transitive relations. Moreover, the case studies used in validating DM2ONT contained three transitive binary relations, all of which were successfully identified by DM2ONT. In this experiment also, DM2ONT did not erroneously identify non-transitive relations as transitive. Although the result of this experiment did not yield any false-negatives or false-positives, it is well understood that this result can not be generalized. Chapter 4 contains more details about this method.

1.2. *Symmetric Binary Relations:* Unlike RDB models, an OWL ontology allows modelers to recognize symmetric binary relations between concepts. In order to discover symmetric binary relations in the source database, I identified various RDB design guidelines/patterns that are used for such relations. Upon identifying these relations, DM2ONT performed data analysis in certain cases to confirm the pattern. Once both tests are passed (i.e. conformed the design and data pattern),

DM2ONT marked these relations as *candidate symmetric* in the generated ontology. Moreover, DM2ONT calculated and reported a confidence ratio for these findings as they are also based on data analysis. Similar to the transitivity method, the dissertation also included formal definitions for the method to identify candidate symmetric relations and the algorithms associated with this method. More details about this method can be found in Chapter 4.

1.3. *Relationship Cardinalities:* Ontology models such as OWL allow modelers to assert the minimum, maximum, or exact cardinality as a non-negative integer when establishing relationships between classes. By setting the cardinality, a modeler asserts that an individual/instance from one class can be related to at-least (lower-bound), at-most (upper-bound), or exactly a specific number of individual(s) from another class. By analyzing data instances in the source database, DM2ONT was able to *suggest* the minimum, maximum, and exact cardinality, and set them in the generated ontology model. Similar to other findings that are based on the analysis of data instances, a confidence ratio was computed and assigned to such findings for the modeler to review. From a validation standpoint, the case studies used in validating this method included numerous relationships with upper and lower cardinalities. DM2ONT was able to correctly retrieve almost all cardinalities (true-positives) and did not identify any incorrectly (i.e. no false-positives); the few that were not retrieved (false-negatives) are attributed to low row counts, which resulted in a low confidence ratio. It is worth noting that the results obtained in this experiment can not be

generalized to say that this method will never return false-positives. Furthermore, my claim in this method covers only the portion pertaining to generating ontology constructs for relationship cardinalities. Section 3.3.1.5 contains more details about this method.

1.4. *Sparse-Column Values:* Some ontology models representations (e.g. OWL) allow modelers to specify restrictions on the values that a data-type property can take. In RDB, restrictions on column values are enforced either using database constraints (e.g. using SQL *Check* constraint) or using application-level programming logic. Through the analysis of RDB data instances, DM2ONT was able to identify such columns and set them in the generated ontology. A confidence ratio was computed for and assigned to restrictions that are set based on data analysis. This method was also validated in the case studies that were used in evaluating DM2ONT. The database instances used in these case studies included several columns with sparse values, and all were correctly identified by DM2ONT. On the other hand, the experiment showed few columns incorrectly identified as having sparse values. A careful review of the data in these columns however revealed that they had unrealistic data values (e.g. the start-date column in the project table was identified as sparse because the table had 20 rows with two distinct values in the start-date). Similar to the previous methods, the results here can not be generalized. Further, my claim here includes only the portion pertaining to generating ontology constructs for sparse columns. Details about this method can be found in Section 3.3.1.5.

1.5.2. Additional Contributions

2. *An extensible framework for generating ontologies from different types of data models:* In this research, I developed an extensible framework capable of generating different ontology representations (target) from various types of data models (source). DM2ONT accomplishes extensibility by 1) employing an internal intermediary object representation that is source/target agnostic (i.e. not specific to a source data model or a target ontology representations) and 2) isolating the logic for analyzing the source model and converting to the target models into separate software components. It is worth noting here that the target ontology representations supported by DM2ONT are limited to the power of DM2ONT's internal object representation as described in Section 3.2. Another extensible framework, namely RDBToOnto [24], used an RDB-based intermediary representation and generates OWL-only representation. Although not addressed by this research, DM2ONT can be extended to generate ontologies from sources such as XML and others. Such an extension involves developing a Source-Collector component that is specific for the type of source being addressed to populate the internal intermediary object representation of DM2ONT. Currently, DM2ONT only focuses on two source models: RDB and ORDB; other source models (e.g. XML) should be considered for future work. Using DM2ONT, organizations with diverse models can rely on a single and unified system to generate ontologies from the models they maintain instead of using fragmented tools. Furthermore, instead of constructing ontologies manually and requiring the continuous involvement

of domain expert, organizations can utilize DM2ONT to automate and facilitate the generation of ontologies and thus, minimize the involvement of domain experts.

3. *Methods to generate an Ontology model from an ORDB data model:* In DM2ONT, I developed methods for translating ORDB models (SQL:1999 [7]) into ontology models. This translation is based on the metadata and data instances found in the DBMS where the ORDB models are implemented. The ORDB constructs considered by DM2ONT include different forms of User-Defined Types (UDTs), Array-Type columns, and Reference-Type columns. In order to retrieve ORDB metadata and data instances, I explored different DBMS client interfaces (e.g. ODBC, JDBC, proprietary clients) to identify the appropriate mechanism for retrieving such information. Also, given the fact that RDB and ORDB models are interrelated (i.e. RDB constructs can refer to ORDB constructs and vice versa), DM2ONT combined the translation logic for these two models into a single Source-Collector component; namely the *RDB/ORDB Collector* (section 3.3.1). Unlike another existing method [26], which treats ORDB as RDB models composed of tables and columns, DM2ONT focused on the main constructs found in ORDB models (i.e. UDTs, and Array and Reference columns). With this contribution, organizations using ORDB models can rely on DM2ONT to automatically generate ontologies from ORDB models and thus, facilitate the deployment of their semantic-based solutions.
4. *Comprehensive survey of methods that generate ontologies from RDB:* In this research, I conducted a comprehensive survey of the current methods in the literature for translating RDB into ontology models. This survey, which was published in [4],

covers eighteen different methods. In order to evaluate these methods in a consistent manner, I developed a framework that allowed comparing these methods across approximately twenty different criteria. This comparison framework, which was used in Chapter 2 (*Related Work*) and in [4], enabled a side-by-side comparison of the different methods for translating RDB into ontology models.

5. *A heuristic methodology to measure the relative amount of terminological content (relative explicitness) of ontology models:* In this research, I devised a heuristic methodology for measuring the relative explicitness of one ontology model in comparison to another. This methodology provides a formal measure that can assist in determining which ontology model is more explicit than another. This is achieved by computing a weighted sum of the number of concepts, properties, relationships and restrictions that are found in either ontology. Moreover, the methodology employs an abstract ontology representation to enable comparing ontology models that use different representations. To avoid accounting for extraneous and erroneous ontology constructs in the models being compared, the methodology included a phase that eliminates invalid and redundant constructs. This methodology allows comparing two domain ontology models with each other or a domain ontology model with a reference ontology model. This methodology can aid modelers in assessing the content of the ontologies being evaluated and *quantifying* their level of explicitness. For instance, given two ontology models that were developed independently for the same domain or two different versions of the same ontology, this methodology can be used to measure and quantify which of these two models has more terminological

content or is more explicit about the domain. To the best of my knowledge, there is not a method for quantifying the explicitness between two correct ontology models.

6. *A Comprehensive Approach:* I developed an approach that allows the generation of an explicit ontology model by translating various RBD and ORDB constructs and by recovering hidden semantics in the source models. This was accomplished by translating *most* of the RDB/ORDB constructs that have corresponding ontology constructs. This includes table and UDT schema information, array-type and reference-type columns, primary and foreign keys, and unique and not-null constraints. Furthermore, DM2ONT recovered different types of IS-A relationships, dissolved certain binary many-to-many relationships, determined cardinality of relationships, detected spares-column values, and identified potential symmetric and transitive binary relations. As discussed in the *Related Work* chapter, other approaches perform only a subset of these features. Devising a comprehensive approach allowed DM2ONT to generate ontologies with greater terminological content (i.e., that are more explicit) than those generated by similar approaches. To the best of my knowledge, existing approaches do not recover *all* of these semantics. The *Validation and Results* chapter shows a comparison between DM2ONT and another approach. The results obtained demonstrated the ability of DM2ONT to recover more semantics and generate ontologies that are more explicit than others. Using these methods, DM2ONT will enable organizations to recover semantics embedded in their databases and thus, reduce the involvement of domain experts.

As presented in Chapter 2, there have been previous attempts to develop methods and tools that translate RDB models into ontology models. In my dissertation, however, I addressed several gaps and shortcomings that were found in existing approaches.

1.6. Thesis Statement

My thesis is two-fold:

- An extensible framework can be developed to automate the generation of highly-explicit ontologies from different types of data models, and
- A heuristic methodology can be devised to be effective in measuring the relative explicitness of an ontology model in comparison to another.

My thesis was validated by:

- Developing a software prototype for the proposed framework (DM2ONT),
- Running the software prototype against at least two RDB and ORDB implementations from various sources (e.g. sample databases that are packaged with commercial DBMS offerings) in order to validate the ability of DM2ONT to handle different types of databases, data constructs, data types, integrity constraints, and data instances,
- Examining the syntax of the generated ontology using two OWL parsers/validators,
- Verifying the generated ontology models manually to confirm that all RDB/ORDB constructs were properly translated to their equivalent ontology constructs according to the predefined translation rules,

- Proving the effectiveness of DM2ONT by comparing it to similar approaches. This is achieved using domain requirements, and by computing the recall and precision for the ontology models generated by DM2ONT and another method, and finally
- Measuring the relative explicitness for the ontology models generated by DM2ONT in comparison to that generated by a similar method.

1.7. Organization of the Dissertation

The following table outlines the remainder of this dissertation:

Table 1: Dissertation Outline

Chapter	Chapter Description
1	**Introduction**: This chapter.
2	**Related Work**: This chapter provides a survey of the existing approaches for translating different types of data models into ontology models.
3	**An Extensible Framework for Generating Ontology Models from Data Models:** This chapter presents the main components of the DM2ONT framework, and the rules for translating various RDB/ORDB constructs into their equivalent ontology constructs.
4	**Candidate Symmetric and Candidate Transitive Binary Relations:** This chapter discusses symmetric and transitive binary relations in RDB, and the methods DM2ONT use to identify them.
5	**Heuristic Methodology to Measure the Relative Explicitness of Ontology Models:** This chapter reviews existing ontology evaluation methods, and describes the methodology I devised for measuring the relative explicitness of one ontology model in comparison to another.
6	**Validation and Results:** In this chapter, I discuss the various experiments attempted for validating this research, and present the results I obtained from these experiments. This chapter includes methods that were attempted and proved to be unfeasible and those that were completed successfully.
7	**Conclusion and Future Research.** In this chapter, I present the conclusion of my research, and future directions for this research.

CHAPTER 2: Related Work

2.1. Introduction

The use of ontology in semantic computing has led researchers to explore methods that aid in developing new ontologies. Developing a new ontology for a domain of knowledge is a lengthy process that involves identifying concepts, properties, relationships and restriction in a specific domain of knowledge, and validating the resultant model with domain experts. Such an effort requires expertise in different areas (e.g. domain knowledge, ontology modeling, and an ontology language or editing tool). Moreover, developing an ontology can take several months, and may require continuous involvement of domain experts.

Since some of this knowledge has already been encoded in computer applications and databases, some researchers have concluded that it might be more practical to derive this knowledge from existing assets (e.g. data models). As a result, several methods have been developed for generating ontologies from existing data models. These range from primitive methods that merely translate constructs to advanced methods that infer knowledge beyond what is explicitly encoded.

This chapter provides a comprehensive survey of the methods available to (semi-)automatically generate an ontology model from a data model. Although there are

different types of data models in use today, the translation methods to date have generally focused on relational data models as the prime source of the information; I therefore concentrated on this when comparing the various reported methods of translation from data models to ontology models. These methods generated models using different ontology representations. The richness (i.e. amount of terminological content) of these ontologies differed widely also. In my survey, I developed a comparison framework that allowed evaluating the methods in a consistent manner. This framework was then used to compare eighteen different reported methods, which, to the best of my knowledge, cover all the available methods to date for generating ontology models from data models.

2.2. Background

The rapid growth in the use of databases has led researchers to investigate ways of extracting semantic information about their subject area. This information is embedded in the data model as implemented by the database, and can be obtained by retrieving and analyzing the metadata and data instances that are maintained by the database. Once extracted, the semantic information can be represented using different models including logical data models (e.g. relational schemas), conceptual data models (e.g. Entity-Relationship (ER) models) or ontology models. These models can then be used when maintaining or enhancing existing systems, developing new systems, or integrating several systems. However, since these models represent constructs and constraints differently, a non-trivial translation process is needed to obtain one model from another.

From a conceptual standpoint, translation between models can be classified into one of two types:

- Translation between different abstraction levels within a model type, or
- Translation between different types of models.

The first of these occurs when the source and target models share the same underlying theory (e.g. both models are based on the concepts of relational data modeling). This process is often termed reverse-engineering and/or forward-engineering. For example, in the relational data model, forward engineering is performed to obtain a logical data model from a conceptual data model (e.g. relational schema from ER model), while reverse-engineering is performed to obtain a conceptual data model from a logical data model (e.g. an ER model from a relational schema). Such a translation can be viewed as *Vertical*.

On the other hand, the second type of translation takes place when, for example, translating between relational and hierarchical types of models, or between relational and ontology types of models. This type of translation can be viewed as *Horizontal*. Figure 2 illustrates the two translation types; where the y-axis denotes the different abstraction levels within a particular model (e.g. Model 'A' is represented at two different abstraction levels: Model 1 and Model 2), while the x-axis denotes different types of models (e.g. Model 2 is based on Model A theory while Model 3 is based on model B theory).

Over the past few decades, several approaches have been proposed for performing vertical translation within the Relational Database (RDB) model (e.g. [6,21,27,49]). In

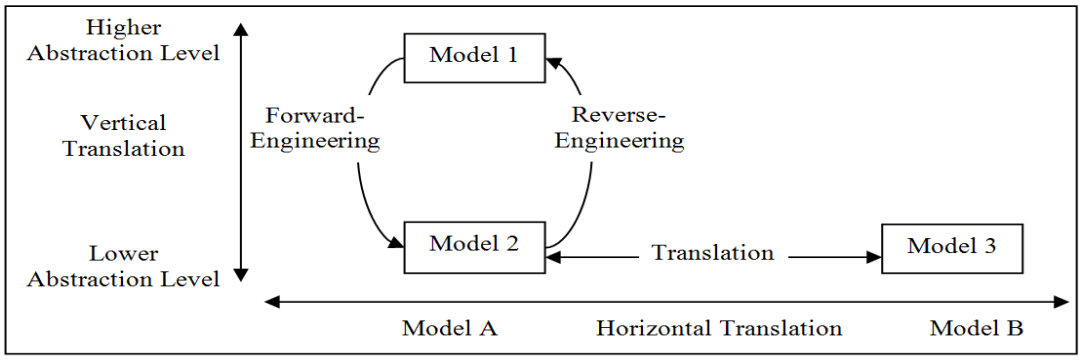


Figure 2: Vertical and horizontal translations

Chiang et al. [27], the authors proposed a semi-automatic and heuristic-based approach to extract an Extended ER (EER) model from an existing RDB by analyzing the RDB metadata and data instances. An EER was defined in [6,27] as an ER model that included aggregation, generalization, and specialization. On the other hand, Alhajj in [6] proposed a semi-automatic and heuristic-based approach with thorough algorithms for extracting the EER from an existing legacy RDB by analyzing only the data instances. While approaches [6,27] focused on reverse-engineering an existing RDB into a higher-abstraction level (i.e. from a logical to a conceptual data model), other approaches [21,49], which were embedded in commercial software products, exist for reverse-engineering and forward-engineering RDB models based on analyzing their metadata and data instances.

Additional approaches exist also for horizontal translations. With ontology gaining momentum in recent years, a number of approaches were proposed for translating different types of models into ontology models and *vice versa*. While some of these approaches translated from sources such as plain-text [93] and XML [14,37,83,86] to

ontology models, others translated from RDB based data models to ontology models [9-11,18,24,26,28,41,46,47,52,54,55,57,62,65,80,84,97,98]. On the other hand, other approaches exist as well for translating from an ontology model to a relational data model [82,85,87]. However, since the focus in this research is on the translation into ontology models (i.e. considering the ontology model as a target), I limited the discussion here to approaches that generate ontology models as a result of the translation.

In [93], an approach was proposed and implemented for translating plain-text data into OWL ontology. The authors used both domain ontology and HTML pages that were converted into plain-text as input into their system, and they claimed to have achieved approximately 90% precision in extracting OWL concepts, properties, and individuals. In [37], Ferdinand et al. proposed and implemented two independent translation procedures for translating XML into ontology: one that translated XML data instances into an RDF model and another that translated XML schema into an OWL ontology model. Bohring and Auer in [14] proposed and implemented an XSLT based framework for translating XML schemas and XML data instances into OWL ontology models and OWL ontology individuals respectively. Bohring and Auer's approach differed from Ferdinand et al. approach in the following aspects: a) XML data instances were converted into OWL individuals rather than RDF, b) XML schema can be generate from XML data instances when an XML schema is not present, and c) XSLT-based transformation was used to generate OWL instead of Java. In [86], Tsinaraki and Christodoulakis proposed and implemented an XSLT-based model that generated both OWL ontology from XML Schema and mapping between the generated OWL model and the XML schema; this

mapping could be used later to convert OWL individuals that were inferred from or added to the knowledge base back into valid XML data instances. An approach that translate XML Schemas into RDF Schema (RDFS) model was proposed by Thuy et al. in [83]. In their approach, an RDFS model can be generated also from XML data when an XML Schema is not present, though human intervention is required in this case.

Unlike the translation from plain-text and XML to ontology, translations from RDB models to ontology models have been widely investigated. However, a closer look into the current methods revealed several shortcomings. The methods found ranged from theoretical (i.e. never implemented by the authors) to concrete and established (i.e. implemented and tested by the authors). Moreover, some of the methods were primitive as they merely translate every table into an ontology class and every attribute into an ontology property. These methods did not make any attempt to translate database constraints into ontology restrictions, discover hidden IS-A relationships, or appropriately handle certain types of many-to-many relationships. On the other hand, some proposed more sophisticated methods in an attempt to generate a richer ontology. Furthermore, my survey showed a significant overlap between the methods.

2.3. The Comparison Framework

In order to compare the various translation methods, I developed a framework that can be used to address their main characteristics in a consistent manner. This framework has six different dimensions, each of which consists of one or more related elements:

1. ***The context of the method***, which addresses whether the approach serves as a component within a broader scope (as part of a larger system) or is a stand-alone tool.
2. ***Implementation of the method,*** which states whether the method was implemented and tested by the authors.
3. ***The type of models,*** which consists of three elements:
    a. The source model type (e.g. relational, object-relational, XML),
    b. The formalism of the target model (e.g. OWL, RDF, Frame-Logic.), and
    c. Whether the generated model uses an ontology meta-model. Generally speaking, using a meta-model leads to generating an ontology model that describes and mirrors the *source model* rather than the *subject area* served by the source model.
4. ***The source of the information***, which may include the DBMS metadata, Data Definition Language (DDL), data instances, XML schema, database-driven HTML pages, or a combination of them.
5. ***The type of constructs***, which describes the constructs considered when performing the translation, including table schema information (e.g. table and column names and data types), primary and foreign keys, and unique and non-null constraints.
6. ***The type of information inferred,*** which includes the following elements:
    a. *IS-A relationships:* Identifies the types of IS-A relationships inferred by the method. In the RDB model, these relationships can be modeled in one of the following ways:
        i. By linking children tables to the parent table via foreign-keys, which we will refer to as *IS-A type 1*,

ii. By embedding attributes of the parent entity-set into the children tables and thus not creating a parent table; we will refer to this as *IS-A Type 2*,

iii. By using a discriminating attribute in the parent table to distinguish between children tuples and thus, not creating the children tables; we will refer to this as *IS-A Type 3*,

b. *Certain many-to-many relationships:* States whether the method being reviewed translates binary many-to-many relationship-sets that do not have descriptive attributes into ontology relationships (e.g. Object-Properties in OWL) rather than ontology classes. In RDB, these relationship-sets are converted into tables. In ontology however, these types of relationship-sets can be created as direct relationships between classes.

c. *Sparse-column values:* Indicates whether the method identifies columns that contain sparse values (e.g. gender, letter grade for courses).

d. *Relationship cardinalities*: Reports whether the method identifies minimum and maximum cardinalities for relationships. In both RDB and ontology, relationships may have a specified minimum and maximum cardinality on each side of the relationship. In the RDB model, the cardinality may be 0, 1, or unbound. In ontology models however, the cardinality may be any arbitrary non-negative integer or unbound.

e. *Symmetric relations*: Indicates whether the method identifies symmetric relations (e.g. spouse, sibling).

f. *Transitive relations*: Indicates whether the method identifies transitive relations (e.g. manages, part-of).

g. *Unique and Not-Null attr*ibutes: Rather than relying on unique and not-null constraints being defined in the metadata only, a method can perform data analysis to infer these properties.

Since only two of the reviewed methods were available for hands-on evaluation [24,65], my review could only be based on the published literature for the other methods. I therefore had to make an assumption in order to compare these methods: the fact that a method neglected to mention information related to any of the dimensions or elements was deemed sufficient to consider it as missing or overlooked. For instance, if the literature for a particular method showed examples of SQL DDL statements and did not mention that data instances were analyzed, I had to consider that the method only used SQL DDL statements as a source for the translation. Furthermore, for the type of constructs considered by any approach, my review assumed that a specific type of construct was addressed only when it was translated to an ontology axiom correctly. For example, if a primary key constraint was not translated to axioms that stated both totality (not null) and uniqueness, I did not consider it addressed; if either axiom was set correctly, it was considered as partially addressed.

2.4. State-of-the-Art in Generating Ontology from Database Models

In this section, a comprehensive review is provided for eighteen different methods pertaining to the translation from a Relational Database (RDB) and Object-Relational

Database (ORDB) models to an ontology model. Although other database models exist such as the Object-Oriented Database (OODB) and hierarchical database, there does not seem to be any approach in the literature that addresses these models. Lastly, the review below lists the approaches by the date when it was published (in chronological order). In cases where the same author(s) has/have published more than one approach, these approaches will be grouped together under the same section/title.

2.4.1. Stojanovic et al. Approach

In Stojanovic et al. approach [80], the authors focus on enabling database-driven Web pages for the Semantic Web. The authors proposed and implemented an approach that creates a Frame-Logic (F-Logic) Ontology model from the SQL Data Definition Language (DDL) script that was used to create the RDB behind the database-driven Web pages. The approach uses a set of 10 mapping rules to map tables and attributes in the DDL script into F-Logic concepts, properties, and axioms. These rules identify and eliminate tables that were created as a result of many-to-many relationships, and can recognize IS-A relationships between child and parent tables when both exist and are related through a foreign-key relationship (ISA Type 1). The metadata considered by this approach is table schema information, primary and foreign keys, and not-null and unique constraints. Furthermore, the approach claims to handle relationships' cardinality such as one-to-one through the inclusion dependency maintained by the referential integrity information in the DDL scripts; i.e. it assumes that a one-to-one relationship is modeled using two foreign keys, one in each table that participate in the relationship to reference

the other table. However, such modeling technique is rare; database text books [30,72] and database design tools [21,49] model one-to-one relationships using only one foreign key. This type of cardinality is typically obtained only when the attribute designated as a foreign key is defined with unique constraint, or by analyzing data.

2.4.2. Buccella et al. Approach

Buccella el al [18] propose a federation system, which is based on the hybrid Ontology approach for integration [92], to address the need for integrating data from different sources. In the hybrid Ontology approach for integration, each source system is described by a local/source Ontology model, which is in turn mapped to a global Ontology model. In [18], the authors describe in details the procedure used to generate the local Ontology model for an RDB system. The approach takes SQL DDL script as an input and generates an OWL Ontology model. The metadata considered by the proposed approach is table schema information, primary keys (partially addressed; covering totality only) and foreign keys, and not-null constraints. Moreover, the approach identifies and eliminates tables that were created as a result of many-to-many relationships. Although the authors did not discuss whether IS-A relationships or sparse-columns are inferred, they did clearly state that relationship cardinalities are left to the domain experts (i.e. manually set). Lastly, the approach described in [18] does not appear to be implemented.

2.4.3. Astrova Approach(es)

In [9,10,11], Astrova proposed three different approaches to translate RDB models into Ontology models. These approaches differ in the source and target models, source of

information, and type of information considered; therefore, each approach will be discussed separately.

In [9], Astrova proposed a stand-alone approach to translate an RDB model into a Frame-Logic Ontology model. The source of information used in [9] is SQL DDL statements and data instances. The metadata considered in [9] is table schema information, primary and foreign keys, and unique and not-null constraints. Using the metadata and by performing correlation analysis, this approach identifies and eliminates tables that are created as a result of many-to-many relationships, and discovers IS-A relationships between child and parent tables when both exist and are linked via a foreign-key relationship (IS-A Type 1). Although the approach was not implemented, it provided examples on how SQL DDL scripts are translated into Frame-Logic Ontology models.

The second stand-alone approach that was proposed by Astrova in [10] is targeted toward database-driven Web pages. In [10], the approach analyzes HTML forms that are based on RDB models and generates a Frame-Logic Ontology model. Given the limited type of information available in HTML forms, only table schema information and not-null constraints are extracted. Moreover, Astrova in [10] did not mention whether any type of information is inferred, or whether the approach is implemented.

Lastly, a third and more recent approach was proposed by Astrova in [11]. Although the approach has not been implemented as well, it shows examples on how SQL DDL statements for an RDB model are translated into an OWL Ontology model. This approach proposes a set of rules to identify and handle the different type of

relationships that exist between tables such as binary many-to-many and IS-A relationships in a manner similar to that in Astrova's first approach. The metadata considered in [11] is table schema information, primary and foreign keys, and not-null and unique constraints.

2.4.4. Man Li et al. Approach

From the perspective of translating an RDB model into an Ontology model, Man Li et al. in [54,55] proposed an identical approach. In [55] however, the authors extended the approach by refining the generated Ontology model using an external lexical knowledge repository such as WordNet. In the translation approach, Man Li et al. proposed and implemented a set of twelve rules to translate an RDB model into an OWL Ontology model based on the metadata and data instances in the database. The metadata addressed by this approach includes table schema information, primary keys (partially addressed; covering totality), foreign keys, and not-null constraints. The proposed approach also identifies IS-A relationships (IS-A Type 1) and attempts to dissolve many-to-many relationships. However, there are several shortcomings in the proposed approach. First, a many-to-many relationship, even those with descriptive columns, are dissolved and replaced by two object properties (per Rule 5). Second, an n-ary relationship is decomposed into multiple OWL object-properties without creating an OWL class that corresponds to the n-ary relationship (per Rule 6). Third, an object property that corresponds to a foreign key is set to minCardinaly=1, which will enforce totality (per Rule 9). Fourth, a data property that corresponds to a column with the UNIQUE

constraint is set to maxCardinality=1, which denotes atomicity rather than uniqueness (per Rule 11).

2.4.5. Relational.OWL Tool

Laborda and Conard in [52] propose an approach that aims to ease the integration between diverse relational databases through the use of Ontology models. In Relational.OWL, the authors proposed an OWL meta-model that can describe the characteristics of RDB models, and implemented an approach for translating an RDB model to an OWL model that is based on their proposed meta-model. The approach uses the metadata maintained in the DBMS. The type of information considered by Relational.OWL is table schema information, and primary and foreign keys. Because the approach uses its proposed OWL meta-model to describe the RDB model, the generated OWL model mirrors the existing RDB rather than the subject area supported by the RDB model. For instances, the generated model will not dissolve binary many-to-many relationships that do not have descriptive columns or discover IS-A relationships. Furthermore, Relational.OWL does not handle unique and not-null constraints, relationship cardinality, or sparse-column values. Lastly, the Ontology meta-model proposed in Relational.OWL is used by DataMaster [65] (discussed in Section 2.4.8).

2.4.6. RDB2ONT Tool

Trinh et al. in [84] proposed a framework to address the semantic interoperability between relational databases in large-scale environment using the hybrid Ontology approach for integration [92]. The proposed framework uses the RDB2ONT tool, which

is described in detail in [84], to generate an OWL Ontology model from an RDB model. Similar to Relational.OWL in [52], RDB2ONT proposed and used an OWL meta-model to describe RDB models, which leads to generating an OWL Ontology model that describes the RDB rather than the subject area served by an RDB model. The RDB2ONT tool uses the metadata that is maintained in the DBMS to retrieve table schema information, primary and foreign keys, and not-null constraints. Because RDB2ONT is intended to describe the RDB system, the Ontology model generated by the RDB2ONT neither dissolves many-to-many relationships nor identifies IS-A relationships. Lastly, a prototype of RDB2ONT was implemented and discussed in [84].

2.4.7. DB2OWL Tool

In Cullot et al. [28] and Ghawi & Cullot [41], the authors proposed and implemented a tool, DB2OWL, for translating an RDB model to an OWL Ontology model. The proposed DB2OWL tool is part of a framework that addresses semantic interoperability using the hybrid Ontology approach for integration [92]. In DB2OWL, the metadata maintained in the DBMS is used to classify tables into one of three types: tables created as a result of many-to-many relationships, tables that participate in IS-A relationship (ISA Type 1), and tables that are neither in the first or second type. The RDB metadata retrieved by DB2OWL include table schema information, and primary and foreign keys information to determine the relationship between tables. The authors however did not mention whether the generated OWL will have restrictions on the properties that map to the RDB columns defined as primary key, unique, or not-null. Lastly, the DB2OWL tool

appears to be implemented for specific DBMS implementations (i.e. uses Oracle and MySQL system catalog tables/views).

2.4.8. DataMaster Plug-In

Nyulas et al. [65] developed a plug-in for Protégé[(2)] that allows importing an existing RDB model implementation into either Protégé-OWL or Protégé-Frames projects. Using Protégé, these projects can later be exported into different Ontology languages including OWL and RDFS. DataMaster also incorporates the Relational.OWL meta-model [52], which provides users of the plug-in with the option of generating OWL Ontology models that mirror the RDB model being translated. In DataMaster, information about the RDB model is obtained from the metadata maintained in the DBMS. Based on the experiment we performed, we found that it can handle only table schema information as well as primary and foreign keys information for the purpose of determining relationships between tables; in other words, the primary key columns will not be translated into OWL restrictions that state totality and uniqueness. Although the plug-in is publicly accessible (i.e. open-source) and intuitive, the generated Ontology model is very limited since it lacks the support for translating primary keys, and unique and not null constraints into Ontology axioms. The plug-in also does not infer any type of information (e.g. IS-A, certain binary many to many, etc.).

[2] Protégé is a free and open-source Ontology Editor from Stanford University (http://protege.stanford.edu/)

2.4.9. Yan and Changrui Method

In Yan and Changrui [98], the authors proposed a method for generating a domain Ontology model from an RDB model. The authors however did not specify which Ontology formalism is used (e.g. OWL, RDF(S), etc.), nor did they clearly state what was the source of information for the translation (e.g. RDB metadata, DDL, data instances). The proposed method first extracts and analyzes the RDB structure to generate an EER. The generated EER is then translated into a domain Ontology. The information considered for translation by this approach is table schema information, and primary and foreign keys information to determine the relationships between tables. Although this method clearly stated that primary keys will be omitted from the generated Ontology, it did not discuss whether unique and not-null constraints are handled. This method also did not mention whether it is capable of inferring any other type of information. For relationship cardinalities however, the authors suggest either performing data instances' analysis, or consulting a domain-expert. Lastly, it is unclear if the proposed method was implemented.

2.4.10. Xu and Li Approach

In Xu and Li [97], an approach was proposed for translating XML data to OWL Ontology model using an RDB model as an intermediary model. The proposed approach first translates XML data to an Entity-Relationship (ER) model. The generated ER model is then translated into an OWL Ontology model. The approach also uses an OWL meta-model for RDB models, which has implications on the generated Ontology similar to that

found in Relational.OWL and RDB2ONT. Given the lack of semantic information in XML data related to identifying primary key, not null and unique constraints, it is unclear how such information can be obtained automatically. Although the examples given in [97] show an OWL ontology fragment with the primary key, not null, and relationships cardinality axioms set, the authors admit that such information can not be obtained automatically and requires the input of a domain expert. Lastly, the authors did not mention whether the approach was implemented.

2.4.11. Automatic Ontology Generator Tool

Mukhopadhyay et al. [62] proposed and implemented an interactive tool that can assist in constructing an RDF model from an existing RDB model. Using a graphical user interface, the tool prompts the user to select from a list the tables to be translated into RDF. The tool also expects the user to establish IS-A relationships between tables manually. The authors however did not discuss the source of information for translation (e.g. metadata, DDL, data instances, etc.) or the type of information considered (e.g. primary/foreign keys, unique and not-null constraints). Furthermore, the authors did not mention whether the tool can infer any type of information.

2.4.12. Lubyte and Tessaris Framework

Lubyte and Tessaris [57] presented a framework for generating an Ontology model from an RDB model and for creating views in the DBMS to allow data access using the generated Ontology model. The framework uses heuristic methods to reverse engineer an existing RDB model into an Entity-Relationship (ER) model. The derived ER model is

then used to construct a DLR-DB Ontology model; DLR-DB is a variant of DLR-Lite. The framework relies on extracting information from the DBMS metadata (i.e. table schema information, primary and foreign keys, constraints on attributes, and dependencies between tables) to generate the ER. Furthermore, in order generate the ER model, the framework classifies tables in the RDB into one of four types: base relation, specific relation (IS-A child relation), relationship relation, and ambiguous relations that require intervention from the domain expert. Using this classification, the framework is able to recognize IS-A relationships between child and parent tables when both exist and are related through a foreign-key relationship (IS-A Type 1). The authors however did not discuss other type of information that can be inferred. Lastly, it is not clear whether this approach has been implemented.

2.4.13. Changjun Hu et al. Method

Changjun Hu et al. in [47] proposed and implemented a method for translating an RDB model into an OWL Ontology model based on the metadata maintained in the DBMS. The method uses three classification rules to categorize the tables in the RDB model. Based on this classification, the authors propose six mapping rules to translate the RDB tables and attributes to the appropriate OWL constructs (i.e. OWL classes and datatype/object properties). These mapping rules dissolve binary many-to-many relationships that do not have descriptive columns into two OWL object properties. The proposed method takes into account metadata related to table schema information, and primary/foreign keys information to determine the relationship between tables. However,

the method does not translate attributes that are defined as primary keys to OWL data properties. Moreover, the authors did not mention whether RDB constraints such as unique and not-null are obtained from the RDB model and set in the generated OWL model. Other shortcoming of this method include the handling of IS-A relationships and tables that are split into multiple tables for optimization purposes (Rule 2.1); such tables are either mapped into OWL IS-A hierarchy classes or merged into an OWL class respectively. Specifically, the proposed rule for handling these types of tables does not mention how either option will be taken by the method; typically this is determined either by performing data inclusion analysis, or based on input from a domain-expert. This method also does not discover sparse-columns or relationship cardinalities. Lastly, the method seems to be implemented for a specific DBMS implementation (i.e. Oracle) since it uses system catalog tables/views to retrieve metadata instead of using a generic JDBC/ODBC metadata API.

2.4.14. RDBToOnto Tool

Cerbah in [24,25] proposed and implemented a tool for translating RDB models into OWL ontology models. The proposed tool, termed RDBToOnto, is part of a large project that aims at facilitating the transition of existing legacy applications to ontologies (TAO[3]). RDBToOnto uses database metadata and data instances to generate an ontology model. Specifically, the metadata is used to generate ontology classes and properties (both data and object properties) while data instances are used to infer class hierarchies.

[3] http://www.tao-project.eu/researchanddevelopment/demosanddownloads/RDBToOnto.html

Based on the experiment I performed, I found that RDBToOnto uses the RDB metadata to retrieve only table schema information and primary/foreign keys information; the latter is used to determine relationships between tables only. In RDBToOnto, class hierarchies (IS-A Type 3) are inferred by using one of two methods: 1) lexical clues in column names (e.g. column with *category* or *type* in the column name), or 2) data mining (e.g. data diversity and entropy). The experiment I conducted also showed that RDBToOnto does not a) set OWL restrictions for RDB columns that are defined as primary key, unique, or not-null, b) dissolve certain binary many-to-many relationships, or c) infer relationship cardinality, ISA Type 1 and 2, sparse-column, or symmetric/transitive binary relations. Lastly, RDBToOnto supports only specific types of sources (i.e. Microsoft Access and Excel, MySQL, and Oracle), though the tool is designed to be extensible; developers can implement a database reader to translate from database-like sources (e.g. DB2, Microsoft SQL Server, XML) by populating RDBToOnto internal representation, which mirrors the relational database model. Extending RDBToOnto to translate from non-structured sources such as Text corpus or web sources is currently not supported. Furthermore, RDBToOnto does not document whether it can be extended to translate into other types of ontology representations (e.g. RDFS).

2.4.15. OWLFROMDB Tool

He-ping et al. in [46] proposed and implemented a tool, OWLFROMDB, to translate an RDB model into an OWL Ontology model. Similar to some of the other approaches, the tool first reverse-engineers the existing RDB model into an ER model, then translates the

ER model into an OWL Ontology model. OWLFROMDB uses the metadata maintained by the DBMS to obtain both table schema information and primary/foreign keys information; the latter is used only to determine the relationships between tables. OWLFROMDB recognizes IS-A relationships between child and parent tables when both exist and are related through a foreign-key relationship (IS-A Type 1). Lastly, the authors in [46] did not discuss whether unique and not-null constraints are handled, or whether any other type of information is inferred.

2.4.16. RDOL Approach

Chen et al. in [26] proposed a method for semi-automating the generation of OWL ontologies from various types of data sources. The method, which they termed Rule Driven Ontology Learning (RDOL), uses sets of translation rules, each of which is designed for translating from a specific type of data source. In [26], the authors presented a set of rules for translating from ORDB type of instances. The proposed rules however treat ORBD instances as RDB without any regard to the main constructs found in ORDB (e.g. User-Defined Types, Arrays, etc.); these rules view an ORDB instance as a collection of relations, where each relation is composed of set of atomic attributes. To conduct the translation, RDOL uses the metadata and data instances maintained in the DBMS. Using this information, RDOL is able to identify table schema information, foreign keys, and IS-A relationships (Type 1). Although the method proposes rules to handle certain binary many-to-many tables (Rule 5) and transitive relations (Rule 7), these rules appear to be faulty. Specifically, Rule 5 fails to exclude binary many-to-many

relationships that contain descriptive columns, and Rule 7 declares IS-A relationships as transitive, which is already implied by OWL semantics for the subclass construct.

2.5. Summary

Given the attention to ontology and its use in a range of applications in recent years, many researchers have explored automating the development of ontology models by reusing and inferring information from existing data models. In this chapter, I conducted a thorough review of the literature that compared eighteen different methods for translating RDB into ontology models. To compare these methods, I developed a framework that allows evaluating them in a consistent manner. This framework was then used in here to compare the different methods found in this area.

While some of these approaches [46,62,98] provided insufficient information pertaining to the dimensions presented in my comparison framework, another approach [97] provided enough information to classify it as primitive. An approach was considered primitive if it does not translate any RDB constraints. Apart from the aforementioned approaches, the remaining methods made more effort to extract additional semantics from the source model [9,10,18,24,26,28,41,47,52,54,55,57,80,84]. Furthermore, out of the eighteen methods reviewed, only ten clearly stated that their method was implemented [24,28,46,47,52,54,62,65,80,84].

The review of these methods also shows an overlap between them in what was considered for translation (e.g. RDB constraints) and how it was translated. Nevertheless, most methods provided some contribution to the literature. A comparison to determine

which approach is best can be challenging given the fact that some of these methods either were part of a larger system, were proposed theoretically and were never implemented, represented the ontology model using different ontology language (e.g. OWL, RDF, F-Logic, etc.), or used an ontology meta-model (e.g. [52,84,97]), which leads to describing the RDB model being translated instead of the subject area served by the RDB model.

Table 2, 3 and 4 summarize the review along the different dimensions presented in the comparison framework discussed earlier.

Table 2. General properties for the reviewed methods.

Method Name	Stand-alone	Implemented	Type of Model			Source of Information
			Source	Target	Uses Meta-model	
Stojanovic et al	✓	✓	RDB	F-Logic		DDL
Buccella et al			RDB	OWL		DDL
Astrova – 1	✓		RDB	F-Logic		DDL and Data
Astrova – 2	✓		RDB	F-Logic		HTML Forms
Astrova – 3	✓		RDB	OWL		DDL
Man Li et al	✓	✓	RDB	OWL		Metadata & Data
Relational.owl	✓	✓	RDB	OWL	✓	Metadata
RDB2ONT		✓	RDB	OWL	✓	Metadata
DB2OWL		✓	RDB	OWL		Metadata
DataMaster	✓	✓	RDB	OWL &Frames	✓[4]	Metadata
Yan-Changrui	✓		RDB	Unknown	Unknown	Unknown
Xu-Li	✓		XML	OWL	✓	XML Data
Automatic Ontology Generator	✓	✓	RDB	RDF		Unknown
Lubyte/Tessaris	✓		RDB	DLR-DB		Metadata
Changjun Hu et al	✓	✓	RDB	OWL		Metadata
RDBToOnto		✓	RDB	OWL		Metadata & Data
OWLFROMDB	✓	✓	RDB	OWL		Metadata
RDOL	✓		RDB[5]	OWL		Metadata & Data

[4] DataMaster can optionally generate ontology models using Relational.owl meta-model.
[5] RDOL claims to translate ORDB but proposes rules that treat ORDB as an RDB.

Table 3. Type of constructs handled.

Method Name	Type of Constructs				
	Table Schema	Primary Key	Foreign Key	Unique	Not-Null
Stojanovic et al	✓	✓	✓	✓	✓
Buccella et al	✓	Partial	✓		✓
Astrova – 1	✓	✓	✓	✓	✓
Astrova – 2	✓				✓
Astrova – 3	✓	✓	✓	✓	✓
Man Li et al	✓	Partial	✓		
Relational.owl	✓	✓	✓		✓
RDB2ONT	✓	✓	✓		
DB2OWL	✓		✓		
DataMaster	✓		✓		
Yan-Changrui	✓		✓		
Xu-Li	✓				
Automatic Ontology Generator	✓				
Lubyte/Tessaris	✓		✓	✓	✓
Changjun Hu et al	✓		✓		
RDBToOnto	✓		✓		
OWLFROMDB	✓		✓		
RDOL	✓		✓		

Table 4. Type of Information Inferred.

Method Name	Type of Information Inferred						
	IS-A Types 1, 2, or 3	Certain many-to-many	Rel Card	Sparse Cols	Symmetric Rel	Transitive Rel	Unique & Not-Null
Stojanovic et al	Type 1	✓					
Buccella et al		✓					
Astrova – 1	Type 1	✓					
Astrova – 2							
Astrova – 3	Type 1	✓					
Man Li et al	Type 1						
Relational.owl							
RDB2ONT							
DB2OWL	Type 1	✓					
DataMaster							
Yan-Changrui							
Xu-Li							
Auto. Ont. Generator							
Lubyte/Tessaris	Type 1						
Changjun Hu et al		✓					
RDBToOnto	Type 3						
OWLFROMDB	Type 1	✓					
RDOL	Type 1						

CHAPTER 3: An Extensible Framework for Generating Ontology Models from Data Models

3.1. Introduction

In this chapter, I describe an extensible framework for translating various types of data models into different ontology representations. This framework is termed *Data Models to Ontologies (DM2ONT).* Given the pervasiveness of the RDB and ORDB models, DM2ONT initially addresses these two models. However, the framework is intended to be extensible: it will allow translation from other types of data models in future. Moreover, while DM2ONT allows the derived ontology to be expressed in different ontology representations, this research focuses on generating ontologies that are expressed in Web Ontology Language (OWL) [89].

In the following sections, I discuss DM2ONT from an extensibility perspective and present its main components. Next, I focus the discussion on DM2ONT as a method for translating *RDB* and *ORDB* data models into *OWL* ontology models. This chapter concludes with several examples to demonstrate how RDB and ORDB are translated into OWL ontologies.

3.2. DM2ONT Architecture

The DM2ONT framework is designed to allow the translation from various types of data models to different types of ontology representations. In order to allow for extensibility, DM2ONT employed an intermediate object representation and a componentized architecture. Using such mechanisms will enable software developers to plug in support for translating from other types of data models and to other types of ontology representations.

Unlike other frameworks (e.g. [24]), the intermediate object representation used in DM2ONT is model-neutral; i.e. DM2ONT does not assume the models to be an RDB source or OWL target. Furthermore, this intermediate object representation is provided as Java classes for developers to instantiate and populate with findings from a source model, and for DM2ONT to use downstream when converting into a target ontology. It is worth noting here that this intermediate object representation is intended for DM2ONT internal use only (i.e. not to be serialized externally as an ontology).

The intermediate object representation in DM2ONT allows for the creation of ontology classes, data-type properties, object properties (binary relations between instances of classes), and various types of restrictions on classes, data-type and object properties. Restrictions on classes include generalization/specialization type of relationships (with multiple inheritance). For data-type properties, restrictions include cardinalities (minimum, maximum and exact non-negative integers), uniqueness and restricted domain values. Lastly, restrictions on object properties include cardinalities (minimum, maximum and exact non-negative integers), transitivity and symmetry.

From an architectural standpoint, DM2ONT consists of four main components as depicted in Figure 3: *DM2ONT Controller, Source Collector, Ontology Converter*, and *Ontology Generator*. In this architecture, only the Source Collector and Ontology Converter components are model dependent; i.e. they need to be developed for each *type* of source/target models. The following subsections describe each component in details.

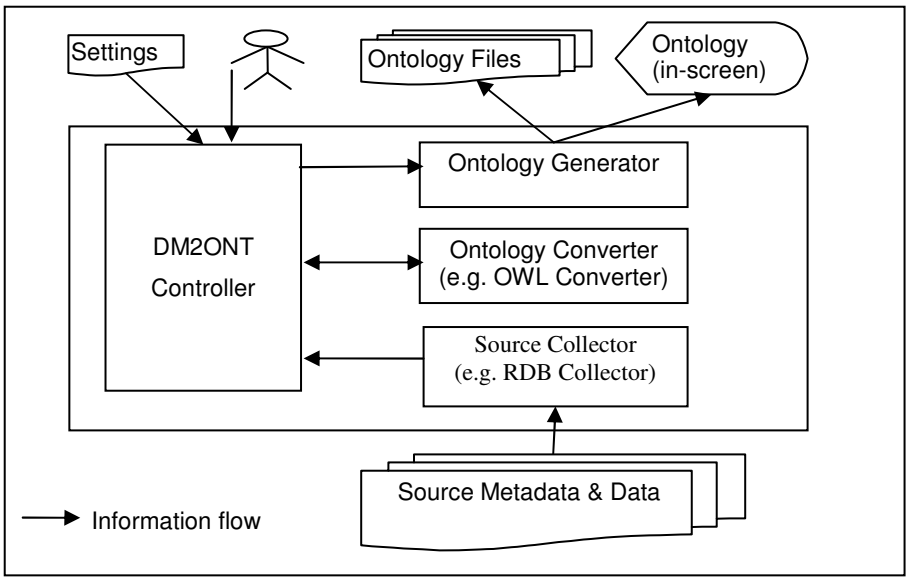


Figure 3. DM2ONT framework architecture.

3.2.1. DM2ONT Controller

The Controller component is responsible for orchestrating DM2ONT activities. These include extracting and analyzing the source data model, populating the intermediate object representation, and converting and generating the ontology model. The controller is initiated when a user launches DM2ONT. The user passes input parameters to

DM2ONT either inline or via a settings file. The parameters include source data model information (e.g. database/file name, user credentials), environment information (e.g. type of source/target models, thresholds), and target ontology information (e.g. output file name/path). The Controller parses the input parameters, and passes them to the Source Collector, Ontology Converter and Ontology Generator components for validation and processing. Since the framework is designed to interact with other types of models, each of which may require different input parameters, validation and processing of the input parameters are assigned to the Source Collector and Ontology Converter components. Upon successful validation of the input parameters, the Controller invokes the Source Collector component to obtain and analyze the source model. Once the extraction and analysis phase is completed and the intermediate object representation is populated, the result is forwarded to the Ontology Converter to perform the conversion from DM2ONT intermediate representation to the target ontology representation. When the ontology conversion phase concludes, the Controller passes the ontology data to the Ontology Generator to produce the ontology model as a file or an in-screen display.

3.2.2. The Source Collector

The Collector component provides DM2ONT with an abstraction layer over the specific characteristics found in each *type* of source data model. A type of source data model includes RDB, ORDB or hierarchal types of models. Different instances of a data model type are handled by a single source collector (e.g. RDB Source Collector handles different RDB instances). A collector component is responsible for extracting and

analyzing information about source data models and populating DM2ONT intermediate object representation.

The DM2ONT framework includes a base Java class for the Source Collector component. Supporting a new type of data model requires extending (i.e. sub-classing in Java) this base class. Furthermore, the base Java class contains polymorphic methods that new source collectors are expected to override. These methods are invoked by the Controller to initialize the Collector, validate input parameters, analyze the source model and populate the intermediate object representation of DM2ONT, and finally to terminate activities in the Collector.

3.2.3. The Ontology Converter

This component provides DM2ONT with an abstraction layer over the types of ontology representations that DM2ONT can translate into. To convert into a specific type of ontology representation, a new Converter component is needed (e.g. OWL Converter, RDFS Converter, etc.). The main responsibility of this component is to convert the DM2ONT intermediate object representation containing the findings from the source data model into the target ontology representation.

Similar to the Source Collector component, DM2ONT also provides a base Java class that needs to be extended to support a specific type of ontology representation. A new converter is expected to override the Java methods found in this base Java class. These methods are invoked by the Controller to initialize the Converter, validate the input

parameters, convert the intermediate object representation of DM2ONT into the target ontology representation, and finally to conclude the conversion phase.

3.2.4. The Ontology Generator

The Ontology Generator component produces the ontology representation in the form of an external file or an in-screen display. The Controller invokes methods in this component to validate user input parameters (e.g. output file name/path), and to output the ontology representation produced by the Ontology Converter.

3.3. DM2ONT for RDB/ORDB and OWL Models

Since this research focuses on translating from RDB and ORDB to OWL models, discussion in the following sections is limited to these types of models. Other data models (such as hierarchical or object oriented models) and ontology representations (e.g. RDFS), could be developed and integrated later.

Generally speaking, the RDB/ORDB Collector in DM2ONT extracts information about RDB and ORDB data models from the metadata maintained by the DBMS and from the data instances. The extracted metadata includes most of the integrity constraints that are typically maintained by a DBMS. To add more semantics about the data model, DM2ONT extracts data instances to fill some of the semantic gaps found in the metadata. The extracted metadata and data instances are then analyzed to identify ontology concepts, properties, and explicit relationships, discover bridge tables and implicit

relationships, and identify restrictions on properties and relationships. The analysis performed by DM2ONT is based on heuristic database modeling techniques.

The OWL Converter in DM2ONT, on the other hand, automatically translates the intermediate object representation into an OWL ontology model. This model can be revised by an ontology modeler based on feedback from domain experts. This derived ontology model is generated in a way that describes the *subject area* of the data model instead of the data model itself.

Figure 4 depicts the architecture of DM2ONT as a method for translating RDB and ORDB into OWL models.

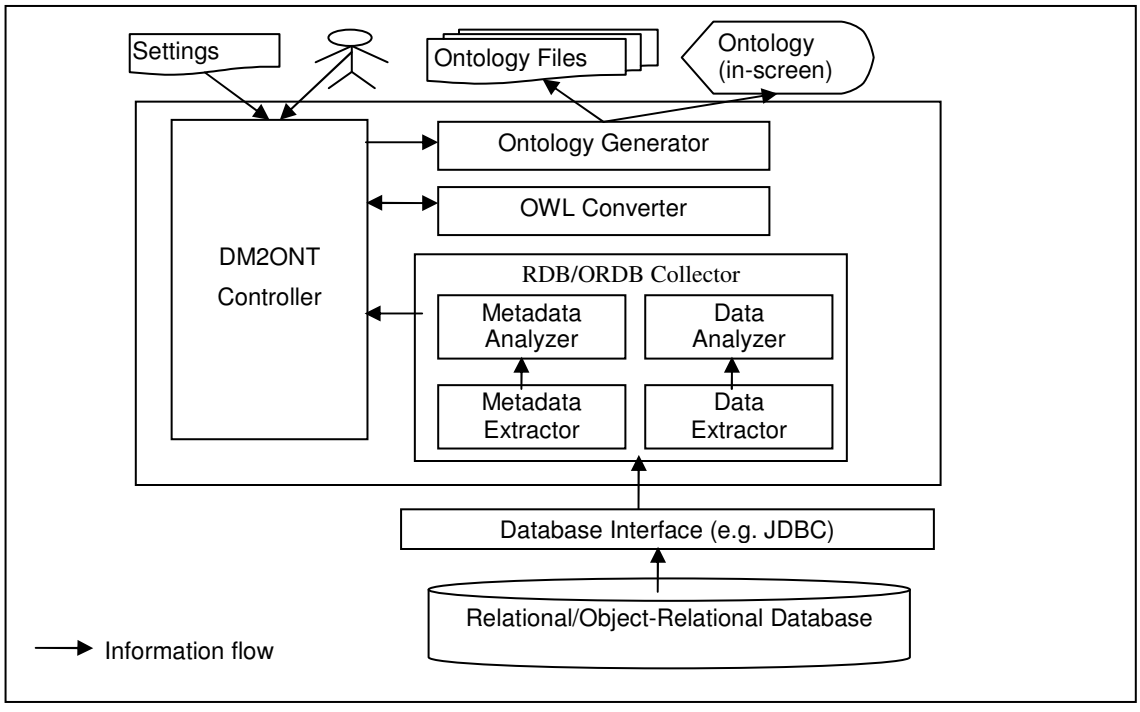


Figure 4: DM2ONT for translation from RDB/ORDB to OWL

3.3.1. The RDB/ORDB Collector Component

The RDB/ORDB Collector component consists of four main subcomponents: *Metadata Extractor*, *Metadata Analyzer*, *Data Extractor,* and *Data Analyzer.* As implied by their names, the *Metadata Extractor* and *Metadata Analyzer* focus on the RDB and ORDB metadata maintained in the DBMS in which these models are implemented. On the other hand, the *Data Extractor* and *Data Analyzer* focus on RDB and ORDB data instances that are contained in the DBMS in which these models are implemented. Before discussing these subcomponents, a preliminary definition of RDB is provided.

3.3.1.1. Preliminary RDB Definitions and Notation:

The underlying model of *relational databases* is the relational model where *relations* are the main constructs for representing data. A relational database is defined as a set of relations, each of which consists of a *relation schema* and a *relation instance*. In turn, a relation schema consists of a set of *attribute-domain* pairs, while a relation instance is a set of *tuples*. The set of relation schemas for all relations in a relational database is termed *relational database schema*. The set of relation instances in a relational database is termed *relational database instance* ([72], pp. 59-62). Here, I extend the formal definition of the relational model to add constructs that are typically used in DBMS and in RDB implementations, and are relevant to DM2ONT. Formally, let:

- *rdbs* be a relational database schema with a finite set of relation schemas:

  rdbs = $\{R_1, R_2, \ldots, R_n\}$, where *n* is the number of relation schemas in *rdbs*,

- R_i (where $1 \leq i \leq n$) be a relation schema with a finite set of Attribute-Domain pairs: $R_i = \{(A_{i1} : D_{i1}), (A_{i2} : D_{i2}), \ldots, (A_{im_i} : D_{im_i})\}$, where m_i is the number of attribute-domain pairs in R_i,
- r_i be a relation instance (or relation in short) that corresponds to (or over) R_i and has a finite set of tuples: $r_i = \{t_{i1}, t_{i2}, \ldots, t_{ix_i}\}$, where tuple t_{iz} ($1 \leq z \leq x_i$) is an element in $D_{i1} \times \ldots \times D_{im_i}$,
- attrib(R_i) be a function that returns the set of attributes in R_i (i.e. $\{A_{i1}, A_{i2}, \ldots, A_{im_i}\}$),
- pkey(R_i) be a function that returns the set of attributes that are defined as part of the primary key in R_i, thus pkey(R_i) $\subseteq$ attrib(R_i),
- fkey(R_i) be a function that returns the set of foreign keys that are defined in R_i: fkey(R_i) = $\{fk_{i1}, fk_{i2}, \ldots, fk_{ir_i}\}$, where fk_{ik} ($1 \leq k \leq r_i$) is a set of one or more attributes in R_i, thus $fk_{ik} \subseteq$ attrib(R_i), and attributes in fk_{ik} and fk_{im} (where $1 \leq m \leq r_i$) are disjoint[6] (i.e. $fk_{ik} \cap fk_{im} = \varnothing$),
- refpk(fk_{ik}) be a function returning the set of primary key attributes referenced by fk_{ik},
- "NULL" be a marker to indicate a missing value for a particular non primary-key attribute in a tuple,
- is_null($t_{iz}[A]$), where $t_{iz}[A]$ is the projection of tuple $t_{iz} \in r_i$ on attribute(s) $A \subseteq$ attrib(R_i), be a function that returns true if *any* component in $t_{iz}[A]$ has NULL marker, and false otherwise,
- is_not_null($t_{iz}[A]$), be the inverse of is_null($t_{iz}[A]$)), and

[6] For referential integrity reasons, renowned database author C.J. Date strongly advice against overlap of foreign keys. Date, C.J, "Relational Database Writings 1985-1989" pp. 153. Addison Wesley. 1990.

– R_1 and R_2 be relation schemas (R_1 and R_2 not necessarily distinct), r_1 and r_2 be relation instances over R_1 and R_2 respectively, pk_1 be the primary key of R_1 (i.e. pk_1=pkey(R_1)), and fk_2 be a foreign key in R_2 that references pk_1 (i.e. $fk_2 \in$ fkey(R_2) $\wedge$ refpk(fk_2) = pk_1). "For all time, each value of fk_2 in r_2 either is NULL or is identical to the value of pk_1 in some tuple in r_1" ([30], pp. 127].

In SQL and DBMS(s), different terms are used to refer to the constructs found in the relational model. Since DM2ONT deals with DBMS implementations rather than the theoretical foundation upon which DBMS(s) are built, the following sections uses the terms employed in DBMS implementations. Specifically, I use the terms table to refer to relation (e.g. table schema, table instance), column and data type to refer to attribute and domain respectively, and row or data instance to refer to tuple.

Furthermore, different terms are used when representing the database design using an Entity-Relationship (ER) model. As discussed in the *Related Work* chapter, widely-accepted guidelines exist for generating a relational database schema from an ER model [6,27,21,49]. Generally speaking, each entity set is mapped to a relation. For a relationship set (or relationship in short) and depending on its cardinality, the database designer can have a choice of mapping it either to a distinct relation or to attributes in one of the relations that corresponds to the entity-set involved in the relationship. In either case, ER relationships are maintained in relational databases using foreign keys.

From a relational database standpoint, a binary relation r_3 from the domain of primary keys D_{pk1} to D_{pk2} (i.e. $r_3 \subseteq D_{pk1} \times D_{pk2}$, where $D_{pk1} = \pi_{pk1}(r_1)$ and $D_{pk2} = \pi_{pk2}(r_2)$,

r_1 and r_2 are relations over schemas R_1 and R_2 respectively, pk_1 = pkey(R_1) and pk_2 = pkey(R_2), and r_1 and r_2 are not necessarily distinct) can have r_3 implemented either as a separate relation/table or as columns in either r_1 or r_2. Specifically, the database designer has the choice of implementing r_3 either as a distinct relation or as columns in r_1 if r_3 is 1:1 (One-to-One) or M:1 (Many-to-One). On the other hand, an r_3 that is N:M (Many-to-Many) can be implemented as a distinct relation only. In the former case, and if the designer elects to merge/embed the attributes/columns of r_3 into r_1, one can dynamically (at runtime) compute the binary relation r_3 using the following query:

$$r_3 = \pi_{pk_1,\, fk_1} (\sigma_{is\_not\_null\,(\,fk_1\,)} (r_1))$$

where r_1 is a relation instance over relation schema R_1, pk_1 is the primary key column(s) in R_1, fk_1 is a foreign key columns(s) in R_1, $pk_1 \cap fk_1 = \varnothing$, and r_3 columns are contained/embedded in r_1 as pk_1 and fk_1.

3.3.1.2. Metadata Extractor

The main responsibility of the *Metadata Extractor* is to retrieve metadata from the DBMS system catalog. Using the source data model information obtained as input parameters, the Metadata Extractor connects to the database using the Database Interface component (i.e. the database client software). In order to support different DBMS implementations (e.g. IBM DB2, Oracle, Microsoft SQL Server), and to be able to obtain ORDB metadata in addition to the RDB metadata, the RDB/ORDB Collector in DM2ONT uses JDBC 3.0 compliant drivers. Other database interfaces are either DBMS implementation specific, platform dependent, or lack support for the ORDB model.

DBMSs today maintain different types of metadata. Some of the metadata maintained by the DBMSs describe the data constructs and their relationships (tables, User-Defined Types, primary/foreign keys, etc.), while others describe storage aspects (table-spaces, page/extents size, etc.) and environment/security characteristics (locale, codepage, permissions, etc.). Since ontology models describe concepts in a subject area, the Metadata Extractor extracts only metadata that describes both the data constructs and their relationships. In order to obtain as much of the semantics as possible from the source data model, the Metadata Extractor retrieves:

- **Table Schema Information:** This includes table and column names, and column data types. The data types can be one of the DBMS built-in data types (e.g. Integer, Character) or User-Defined Types (UDT).
- **User-Defined Type (UDT) Schema Information:** A UDT, which is an ORDB construct, can be either a distinct, complex/structured, or array data type. A distinct data type is a user-defined atomic data type created to allow strong typing. A complex data type is a user-defined type with an internal structure composed of multiple attributes, each of which has a name and data type; moreover it can participate in an *IS-A* hierarchy. An array data type is discussed below. The UDT schema information for both distinct and complex data types includes the UDT name, and either the base-type for the distinct data types or the attribute names and data types for complex data types.
- **Array-Type Columns:** An array data type is a UDT that consists of an ordered set of elements of the same data type. The information obtained includes the column

name, the array cardinality (i.e. number of elements in the array), and the data type for the elements of the array.

- **Reference-Type Columns:** Information about these columns is used to identify how UDT(s) reference one another. The information obtained includes the reference column name and data type, and the referenced UDT.
- **Primary Key Constraints:** This includes the primary key constraint name and the column name(s) that makes up the primary key.
- **Foreign Key Constraints:** This includes the foreign key constraint name, the local column(s), and the referenced table and column(s).
- **Not-Null and Unique Constraints:** Information is obtained about these constraints for each column.

Upon successful retrieval of this information from the DBMS, the Extractor forwards this information to the Metadata Analyzer for further analysis.

3.3.1.3. Metadata Analyzer

The *Metadata Analyzer* performs analysis on the metadata; its main task involves identifying Bridge Tables and implicit relationships:

Identify Bridge Tables: Tables that were created as a result of many-to-many relationships between two entity sets and have no descriptive columns are generally created to overcome a known limitation in modeling binary many-to-many relationships in RDB(s). This limitation dictates creating tables to capture such relationships. In DM2ONT, these types of tables are referred to as *Bridge Tables*. A bridge table is

designated as such if the table has a composite (multi-attribute) primary key, the primary key attributes are defined as two foreign keys, and the table has no columns other than those in the primary key.

Definition 3.1: Let *rdbs* be a relational database schema. R_3 is declared as a *bridge table* if:

$$\{ R_3 \mid R_3 \in \text{rdbs} \wedge \exists R_1, R_2 ((R_1, R_2 \in \text{rdbs}) \wedge$$

$$(\exists pk_1, pk_2, pk_3 ((pk_1 = \text{pkey}(R_1)) \wedge (pk_2 = \text{pkey}(R_2)) \wedge (pk_3 = \text{pkey}(R_3)) \wedge$$

$$(\exists fk_1, fk_2 ((fk_1 \in \text{fkey}(R_3)) \wedge (fk_2 \in \text{fkey}(R_3)) \wedge (\{fk_1, fk_2\} = \text{fkey}(R_3)) \wedge$$

$$(fk_1 \cup fk_2 = pk3) \wedge (pk3 = \text{attrib}(R_3)) \wedge (\text{refpk}(fk_1) = pk_1) \wedge (\text{refpk}(fk_2) = pk_2)))))\}$$

In DM2ONT, a bridge table such as R_3 is not translated into an ontology class. Instead, such table is translated as a direct relationship (e.g. object property in OWL) between the ontology classes that corresponds to R_1 and R_2. Note that other types of tables that are created as a result of many-to-many relationships (e.g. n-ary relationships, binary relationships with descriptive columns) are translated into ontology classes.

Identify IS-A Relationships: A *potential* IS-A type of relationship is identified when there is a set of tables that either share a common primary key with foreign keys referencing one of the tables in the set of tables being examined, or share common attributes and a common primary key; these two types respectively corresponds to IS-A type-1 and type-2 that were introduced in Chapter 2. As an example of IS-A type 1, let R_1, R_2, and R_3 be relation schemas in the relational database schema *rdbs* with pk_1, pk_2, and pk_3 as the primary key for R_1, R_2, and R_3 respectively. A potential IS-A relationship is

declared if we have pk_1 defined in R_1 as a foreign key referencing pk_3, and pk_2 defined in R_2 as a foreign key referencing pk_3. In this case, R_3 is declared as a potential super-class in the generated ontology, and R_1 and R_2 as subclasses.

Definition 3.2: Let *rdbs* be a relational database schema. R_1, R_2 and R_3 are declared to have a potential IS-A type 1 relationship if:

$$\{ R_3 \mid R_3 \in \text{rdbs} \wedge \exists R_1, R_2 ((R_1, R_2 \in \text{rdbs}) \wedge$$

$$(\exists pk_1, pk_2, pk_3 ((pk_1 = \text{pkey}(R_1)) \wedge (pk_2 = \text{pkey}(R_2)) \wedge (pk_3 = \text{pkey}(R_3)) \wedge$$

$$(\exists fk_1, fk_2 ((fk_1 \in \text{fkey}(R_1)) \wedge (fk_2 \in \text{fkey}(R_2)) \wedge$$

$$(fk_1 = pk_1) \wedge (fk_2 = pk_2) \wedge (\text{refpk}(fk_1) = pk_3) \wedge (\text{refpk}(fk_2) = pk_3))))))\ \}$$

An example of IS-A type 2 is when we have two relation schemas R_1 and R_2 with pk_1 and pk_2 as the primary key for R_1 and R_2 respectively. If pk_1 and pk_2 are syntactically equivalent (i.e. share the same attribute names and domains), and the intersection of the attributes in R_1 and R_2 produces a set of attributes other than pk_1 and pk_2, a potential *IS-A* relationship is declared. In this case, R_1 and R_2, without the attributes in the intersection between them, are declared as subclasses and R_3, which consists of the attributes found in the intersection between R_1 and R_2, is declared a super-class.

Definition 3.3: Let *rdbs* be a relational database schema. R_1 and R_2 are declared to have a potential IS-A type-2 relationship if:

$$\{R_1, R_2 \mid R_1, R_2 \in \text{rdbs} \wedge$$

$$\exists pk_1, pk_2 ((pk_1 = \text{pkey}(R_1)) \wedge (pk_2 = \text{pkey}(R_2)) \wedge (pk_1 \notin \text{fkey}(R_1)) \wedge (pk_2 \notin \text{fkey}(R_2)) \wedge$$

$(| pk_1 | = | pk_2 |) \wedge (\forall x \in pk_1 , \exists y \in pk_2 (SYNTAC\_EQ^7(x , y))) \wedge$

$(\exists ca_1, ca_2 ((ca_1 \subseteq attrib(R_1)) \wedge (ca_2 \subseteq attrib(R_2)) \wedge (|ca_1| > 1) \wedge (|ca_1| = |ca_2|) \wedge$

$(ca_1 \cap pk_1 = \varnothing) \wedge (ca_2 \cap pk_2 = \varnothing) \wedge (\forall a \in ca_1, \exists b \in ca_2 (SYNTAC\_EQ^7(a , b)))))) \}$

3.3.1.4. Data Extractor

The main task of the *Data Extractor* is to issue SQL statements to retrieve information about data instances for all the tables in the data model being translated. The information retrieved includes the total number of data instances (i.e. number of rows) for each table, number of null values for each column, number of distinct values for each column, and data instances themselves in certain cases. This subcomponent retrieves information based on requests from the Data Analyzer subcomponent.

3.3.1.5. Data Analyzer

The *Data Analyzer* performs analysis on information about data instances in order to recommend – in the generated ontology -- the cardinality of relationships, sparse-column values, candidate transitive and symmetric binary relations, and not-null and unique columns (those without null/unique constraints). However, since data instances in an RDB implementation represent the state of an organization at a given point in time, a confidence ratio is derived from the data instances information, and provided to the ontology modeler to assist him/her when reviewing the generated ontology. The formula for deriving the confidence ratio is shown in Algorithm 1.

[7] SYNTAC_EQ is a Boolean function that returns true if two attributes are syntactically equivalent.

```
01 Method table-confid-ratio:
02 Input: tab (table instance) , confid_ratio_threshold (0 ≤ decimal value ≤ 1)
03 Output: table_confid_ratio ( 0 ≤ decimal value ≤ 1), acceptable_ratio (Boolean)
04 Begin-Steps
05 Let table_card = SELECT COUNT (*) FROM tab
06 If (table_card > 10 ) then
07   Let table_confid_ratio = 1 – ( 10 / table_card)
08 else
09   Let table_confid_ratio = 0
10 End_If
11 Let acceptable_ratio = table_confid_ratio >= confid_ratio_threshold
12 End-Steps
```

Algorithm 1: Confidence ratio formula

Table 5 shows the confidence ratio for tables with different number of rows (*table_card*) when the Confidence Ratio Threshold *(confid_ratio_threshold)* is set to 0.9.

Table 5. Examples of confidence ratio threshold for different cardinalities.

Number of Rows	Confidence Ratio	Accepted?
0	0	False
10	0	False
50	0.80	False
100	0.90	True
200	0.95	True
1,000	0.99	True

In addition to deriving the confidence ratio, the other main responsibilities of the Data Analyzer are to discover the cardinality of the relationships, detect sparse-column

values, identify candidate symmetric and transitive binary relations, and uncover not-null and unique columns:

Discover Relationship Cardinality: In ontology models, cardinality restrictions can be placed on a relationship between classes to state the minimum, maximum, or exact cardinality between instances of these classes. Some ontology representations (e.g. OWL) allow modelers to set cardinalities to arbitrary non-negative integer values. In relational databases however, cardinalities from Entity-Relationship (ER) models are typically generalized into one-to-many (1:M) to state the maximum cardinality. In order to obtain and recommend the cardinalities for each relationship in a relational database, and given this semantic gap, data retrieval and analysis is performed. This includes finding the minimum and maximum cardinality for all relationships and their inverses. The proposed cardinalities however are obtained only when the confidence ratio (in algorithm 1) is above a user-supplied threshold value.

Definition 3.4: Given relation schemas R_1 and R_2 in a relational database schema *rdbs* and integrity constraints pk_1 and pk_2 as the primary keys of R_1 and R_2 respectively (i.e. $pk_1 = \text{pkey}(R_1) \wedge pk_2 = \text{pkey}(R_2)$). For any relation instances r_1 over R_1 and r_2 over R_2, let D_{pk1} and D_{pk2} be the projection of pk_1 and pk_2 values respectively (i.e. $D_{pk1} = \pi_{\text{pk1}}(r_1) \wedge D_{pk2} = \pi_{\text{pk2}}(r_2)$), r_3 be a binary relation from D_{pk1} to D_{pk2} (i.e. $r_3 \subseteq D_{pk1} \times D_{pk2}$), and R_3 be the schema of r_3 such that $R_3 = \{(A_{pk1}:D_{pk1}), (A_{pk2}:D_{pk2})\}$. For R_3, the minimum cardinality denotes the minimum number of pk_2 values (D_{pk2} elements) a given pk_1 value (a D_{pk1} element) must pair with in any r_3, maximum cardinality denotes the maximum

number of pk_2 values a given pk_1 value can pair with in any r_3, the inverse minimum cardinality states the minimum number of pk_1 values a given pk_2 value must pair with in any r_3, and the inverse maximum cardinality states the maximum number of pk_1 values a given pk_2 value can pair with in any r_3.

The algorithm below computes the relationship cardinalities for the binary relation schema R_3 as defined in 3.4. This algorithm is given for illustration purposes, and in practice, more efficient algorithms may be implemented. Note that in the following algorithm, the binary relation r_3 can be given (i.e. r_3 exist in the relational database as a distinct relation) or computed as discussed in section 3.3.1.1 (i.e. r_3 columns are embedded in relation r_1):

- Compute r_{tmp1} : r_{tmp1} = Select Count (A_{pk1}) As card From r_3 Group By (A_{pk1})
- Set min_card (minimum cardinality) to:
  - 0 (zero): if "Select Count(Distinct A_{pk1}) From r_3" < "Select Count(pk_1) From r_1", or otherwise
  - x : x = Select Min (card) From r_{tmp1}
- Set max_card (maximum cardinality) to:
  - x : x = Select Max (card) From r_{tmp1}
- Compute r_{tmp2} : r_{tmp2} = Select Count (A_{pk2}) As card From r_3 Group By (A_{pk2})
- Set inv_min_card (inverse minimum cardinality) to:
  - 0 (zero): if "Select Count(Distinct A_{pk2}) From r_3" < "Select Count(pk_2) From r_2", or otherwise
  - x : x = Select Min (card) From r_{tmp2}
- Set inv_max_card (inverse maximum cardinality) to:
  - x : x = Select Max (card) From r_{tmp2}.

Detect Sparse-Column Values: A column is proposed as one that has sparse values if both the number of distinct values and the Sparse Confidence Ratio (SCR) for the column are within user-specified thresholds. The max_distinct threshold setting allows users to specify the maximum number of distinct values that a column can have in order to be considered as such (e.g. maximum of 10 distinct values). The min_SCR threshold setting enables users to set the minimum acceptable SCR for such columns. *SCR* is calculated as:

$$SCR = 1 - (DV / NNV)$$

where DV is the number of distinct values, and NNV is the number of non-null values. Furthermore, to allow ontology modelers to make a judgment, the sparse confidence ratio is reported for columns that DM2ONT propose as sparse columns. An example of a column with sparse values is a column that contains gender indicator (e.g. M or F), or course-grades (e.g. A, B, C).

Table 6 shows examples of the sparse confidence ratio for columns with different number of distinct values and different number of non-null values.

Table 6. Sparse Confidence Ratio (SCR) for sparse-column values.

Number of Non-Null Values	Number of Distinct Values	Sparse Confidence Ratio
50	50	0
50	10	0.80
50	5	0.90
100	10	0.90
100	5	0.95
1000	10	0.99
1000	5	0.995

Identify Candidate Transitive and Candidate Symmetric Binary Relations: In RDB, transitive and/or symmetric binary relations can not be annotated as such (i.e. SQL does not provide constructs to identify them). Using heuristic data modeling techniques and data instance analysis, such relations can be identified as *likely* to be transitive and/or symmetric. For example, based on metadata and data analysis, binary one-to-one and many-to-many relations may be identified as candidate symmetric (e.g. spouse-of and friends-with relations in Person's table). Examples of candidate transitive relations include binary one-to-one, one-to-many, and many-to-many (e.g. next-in-queue, manager-of for employee table, or composed-of for a bill-of-materials table). By analyzing data instances in these types of relations, candidate symmetric and transitive binary relations can be identified. Chapter 4 provides more details on the RDB design guidelines/patterns used with such relations and how DM2ONT identifies them.

Uncover Not-Null and Unique Columns: In addition to identifying not-null and unique columns based on database constraints (by the Metadata Extractor), DM2ONT analyzes data instances in each table in the database to uncover columns that do not contain null markers and columns that contain unique values. Such findings are proposed in the generated ontology if the confidence ratio for the table being analyzed is above a user-specified threshold.

Upon concluding the data analysis phase, information and control are returned to DM2ONT Controller.

3.3.2. The OWL Converter Component

This component converts DM2ONT internal object representation of the source data model into an OWL Full representation. For a source data model that neither contains primary key nor unique columns, the OWL Converter produces an OWL DL representation. This component converts the internal representation as follows:

- **Table Schema Information:** This is converted into OWL classes and properties in the ontology model. Specifically, and except for tables that were identified by the Metadata Analyzer as bridge, each table is converted into <owl:Class>. Furthermore, all non foreign-key, non reference-type, and non UDT columns are mapped to <owl:DatatypeProperty>. These columns, which use the DBMS built-in data types, are mapped to their equivalent built-in XML Schema data types using <rdf:range> in their respective <owl:DatatypeProperty> element. Columns that are defined as foreign-key, reference-type, or UDT are converted into <owl:ObjectProperty> with the <rdf:range> set to the referenced table or UDT.
- **User-Defined Type (UDT) Schema Information:** This information is converted in a manner similar to the table schema information. In particular, a UDT is converted into OWL classes <owl:Class>. Attributes within a UDT are converted into <owl:DatatypeProperty> if these attributes have a type that is neither reference-type nor UDT. An attributes that is defined as reference-type is handled as a reference-type column (see below). An attribute that is defined as UDT is handled recursively. In both cases, reference-type and UDT attributes are converted into <owl:ObjectProperty> in the class where they are defined.

- **Array-Type Columns:** An Array-Type column is converted into an <owl:DatatypeProperty> in the class that maps to the table or UDT where the Array-Type column is defined. Furthermore, the array cardinality (i.e. number of elements in the array) is used to set the <owl:maxCardinality> restriction for the property that maps to the Array-Type column.
- **Reference-Type Columns:** A column that is defined as Reference-Type is converted into <owl:ObjectProperty> with the <owl:FunctionalProperty> restriction set to indicate that this property can have one value at most.
- **Primary Key Constraints:** A single-column primary key is translated to <owl:InverseFunctionalProperty> (i.e. unique) [89] and <owl:minCardinality=1> (i.e. not-null) restrictions in the OWL data property created for the primary key column. For a multi-column primary key, this information is set as an <rdfs:comment> in the class that corresponds to the table where this key is defined.
- **Foreign Key Constraints:** Information about foreign keys is used to identify relationships between classes in the ontology model. Such columns are converted into either two <owl:ObjectProperty> or an <rdfs:Subclass> depending on the result obtained from the metadata analysis phase (e.g. table relationship or *IS-A* relationship.) For non-IS-A relationships, two <owl:ObjectProperty> are created between the classes that participate in the relationship being addressed to capture the two-way nature of a relationship. Specifically, an <owl:ObjectProperty> is created in the class that corresponds to the table that has the foreign key(s), with an <owl:FunctionalProperty> restriction to indicate that this property can have one

value at most; we refer to this relationship as a *child relationship*. Another <owl:ObjectProperty> is created in the class that corresponds to the table that is being referenced by the foreign key, with an <owl:inverseOf> construct to highlight the relationship between these two <owl:ObjectProperty>; we refer to this relationship as a *parent relationship*. Cardinality restrictions for *child* and *parent relationships* are set based on data analysis as described below.

- **Not-Null Columns:** Information about columns that are identified as Not-Null based on database constraints or data analysis are translated into <owl:minCrdinality=1> restriction in the data properties that map to these column.
- **Unique Columns:** Columns that are identified as unique (based on constraints or data analysis) are translated into <owl:InverseFunctionalProperty> in the data properties that map to these columns.
- **Relationships Cardinalities:** Depending on the relationship cardinality analysis performed by the Data Analyzer, different cardinality-related constructs might be set in each of the two <owl:ObjectProperty> that corresponds to the child and/or parent relationship.
- **Sparse-Column Values:** Columns identified as having sparse values are translated into <owl:DatatypeProperty>. The <rdf:range> for this property is set to <owl:DataRange> with the column's distinct values specified using the basic list constructs (i.e. <rdf:first>, <rdf:rest> and <rdf:nil>). To assist the ontology modeler when reviewing the generated ontology model, the sparse confidence ratio is

reported using <rdfs:comment>. Example IV in Section 3.4 below demonstrates how the translation is conducted for such columns.

- **Transitive and Symmetric Binary Relations:** A binary relation that is identified as a candidate transitive and/or candidate symmetric is translated into <owl:ObjectProperty> with <rdf:type rdf:resource="&owl;TransitiveProperty"/> and/or <rdf:type rdf:resource="&owl;SymmetricProperty"/> constructs respectively. Furthermore, the confidence ratio is provided as an <rdfs:comment> to assist the modeler when reviewing the generated ontology model.

Upon successfully converting DM2ONT internal object representation of the source data model into an OWL representation, the OWL Converter returns the OWL representation to the Controller.

3.4. Examples

We now provide four examples to show how DM2ONT translates RDB and ORDB models into OWL ontology models. The first example simply highlights how an RDB table with primary key and not-null constraints is translated into an OWL class. The second example shows how DM2ONT handles bridge Tables. The third example shows an ORDB schema and how UDT(s) are translated. Finally, the fourth example shows how DM2ONT translates sparse-column values.

3.4.1. Example I (RDB Schema - Simple)

Consider the following table (using pseudo syntax):

- TABLE HR.Employee (id CHAR(10) PRIMARY KEY, name CHAR(30) NOT NULL),

This HR.Employee table has two columns: the 'id' column, which is defined as *primary key*, and the 'name' column, which is defined as a *Not-Null*. Figure 5 shows the OWL equivalent description of this table.

A column that is defined as a primary key is treated as though it was defined with both not-null and unique constraints. In the OWL representation in Figure 5, lines 7 12 and line 23 apply the not-null and the unique constraints on the 'id' property using the <owl:minCardinality> and <owl:InverseFunctionalProperties> restrictions. Similarly, lines 13 18 show how not-null constraint on the 'name' property is handled. Lines 24 and 29 apply the <owl:FunctionalProperty> restrictions on the 'id' and 'name' properties, which indicate that these properties can have one value at most. Since database columns

that are not UDT-based can have atomic values only, these columns will always be defined in the generated OWL model with the <owl:FunctionalProperty> restriction.

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <rdf:RDF … >
03 ...
04 <owl:Class rdf:ID=”HR.Employee”>
05  <rdfs:subClassOf rdf:resource=”&owl;Thing”/>
06  <rdfs:comment>HR.Employee table</rdfs:comment>
07  <rdfs:subClassOf>
08   <owl:Restriction>
09    <owl:onProperty rdf:resource="#HR.Employee.id"/>
10    <owl:minCardinality rdf:datatype="&xsd;int"> 1  </owl:minCardinality>
11   </owl:Restriction>
12  </rdfs:subClassOf>
13 <rdfs:subClassOf>
14   <owl:Restriction>
15    <owl:onProperty rdf:resource="#HR.Employee.name"/>
16    <owl:minCardinality rdf:datatype="&xsd;int"> 1 </owl:minCardinality>
17   </owl:Restriction>
18  </rdfs:subClassOf>
19 </owl:Class>
20 <owl:DatatypeProperty rdf:ID="HR.Employee.id">
21  <rdfs:domain rdf:resource="#HR.Employee" />
22  <rdfs:range  rdf:resource="&xsd;string"/>
23  <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
24  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
25 </owl:DatatypeProperty>
26 <owl:DatatypeProperty rdf:ID="HR.Employee.name">
27  <rdfs:domain rdf:resource="#HR.Employee" />
28  <rdfs:range  rdf:resource="&xsd;string"/>
29  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
30 </owl:DatatypeProperty>
31 ...
```

Figure 5: OWL Representation for Example I.

3.4.2. Example II (RDB Schema – Bridge Table)

Consider the following table (using pseudo syntax):

- TABLE HR.Employee (id CHAR(10) PRIMARY KEY, name CHAR(30) NOT NULL),
- TABLE HR.Project (p-id CHAR(8) PRIMARY KEY, p-name CHAR(30))
- TABLE HR.Assigned-To (id CHAR(10), p-id CHAR(8), PRIMARY KEY (id, p-id), FOREIGN KEY id REFERENCES HR.Employee, FOREIGN KEY p-id REFERENCES HR.Project)

As shown, the Assigned-To table shows a many-to-many binary relation between the Employee and Project tables. Figure 6 shows the corresponding OWL output.

For clarity and to eliminate repetition, the OWL representation shown in Figure 6 omits the HR.Employee properties and restrictions imposed on these properties (lines 5 and 7) since they are identical to those in Figure 5. Since the Assigned-To table was created to overcome an RDB limitation related to modeling binary many-to-many relations (see *Identify Bridge Tables* in 3.3.1.3), the OWL representation shown in Figure 6 does not create an <owl:Class> that corresponds to the Assigned-To table. Instead, two <owl:ObjectProperty> properties are created to model the relationship between the HR.Employee and HR.Project class as seen in lines 8-12 and lines 28-32 respectively.

3.4.3. Example III (ORDB Schema)

Consider the following table (using pseudo syntax):

- TYPE HR.Address AS (line_1 CHAR(30), city CHAR(30)).
- TYPE HR.US_Address UNDER HR.Address AS (state CHAR(2), zip-code CHAR(10)).

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <rdf:RDF … >
03 ...
04 <owl:Class rdf:ID="HR.Employee">
05   ...
06 </owl:Class>
07 ...
08 <owl:ObjectProperty rdf:ID="HR.Employee.has-project">
09  <rdfs:domain rdf:resource="#HR.Employee"/>
10  <rdfs:range rdf:resource="#HR.Project"/>
11  </owl:ObjectProperty>
12 <owl:Class rdf:ID="HR.Project">
13  <rdfs:subClassOf rdf:resource="owl;Thing"/>
14  <rdfs:comment>HR.Project table.</rdfs:comment>
15   … <!-- minCardinality for p-id datatype property -->
16 </owl:Class>
17 <owl:DatatypeProperty rdf:ID="HR.Project.p-id">
18  <rdfs:domain rdf:resource="#HR.Project" />
19  <rdfs:range  rdf:resource="&xsd;string"/>
20  <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
21  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
22 </owl:DatatypeProperty>
23 <owl:DatatypeProperty rdf:ID="HR.Project.p-name">
24  <rdfs:domain rdf:resource="#HR.Project" />
25  <rdfs:range  rdf:resource="&xsd;string"/>
26  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
27 </owl:DatatypeProperty>
28 <owl:ObjectProperty rdf:ID="HR.Project.has-employee">
29  <rdfs:domain rdf:resource="#HR.Project"/>
30  <rdfs:range rdf:resource="#HR.Employee"/>
31  <owl:inverseOf rdf:resource="#HR.Employee.has-project"/>
32 </owl:ObjectProperty>
33 ...
```

Figure 6: OWL representation for Example II

This example shows an ORDB schema with two UDT(s): HR.Address and HR.US_Address. The schema also shows the UDT HR.Address as a super-class and UDT HR.US_Address as a subclass, as indicated by the ORDB keyword 'UNDER' in the HR.US_Address UDT definition. Figure 7 illustrates the corresponding OWL representation for this ORDB schema. Since HR.US_Address is defined as a subclass of

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <rdf:RDF … >
03 ...
04 <owl:Class rdf:ID=”Person”>
05  <rdfs:subClassOf rdf:resource=”&owl;Thing”/>
06  <rdfs:comment>Person table</rdfs:comment>
07 </owl:Class>
08 <owl:DatatypeProperty rdf:ID="Person.id">
09  <rdfs:domain rdf:resource="#Person" />
10  <rdfs:range  rdf:resource="&xsd;string"/>
11 <rdf:type rdf:resource="&owl;FunctionalProperty"/>
12 </owl:DatatypeProperty>
13 <owl:DatatypeProperty rdf:ID="Person.name">
14  <rdfs:domain rdf:resource="#Person" />
15  <rdfs:range  rdf:resource="&xsd;string"/>
16  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
17 </owl:DatatypeProperty>
18 <owl:DatatypeProperty rdf:ID="Person.gender">
19  <rdfs:domain rdf:resource="#Person" />
20  <rdfs:comment>Sparse Confidence Ratio = 0.98 </rdfs:comment>
21  <rdfs:range>
22   <owl:DataRange>
23    <owl:oneOf>
24     <rdf:List>
25      <rdf:first rdf:datatype="&xsd;string">Male</rdf:first>
26      <rdf:rest>
27       <rdf:List>
28        <rdf:first rdf:datatype="&xsd;string">Female</rdf:first>
29        <rdf:rest rdf:resource="&rdf;nil" />
30       </rdf:List>
31      </rdf:rest>
32     </rdf:List>
33    </owl:oneOf>
34     </owl:DataRange>
35   </rdfs:range>
36  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
37 </owl:DatatypeProperty>
38 ...
```

Figure 7. OWL representation for Example III.

HR.Address, line 19 uses the <rdfs:subClassOf> construct to denote the relationship between these two classes.

3.4.4. Example IV (Sparse-Column Values)

Consider the following table definition and data:

- TABLE Person (id CHAR(10), name CHAR(30), gender CHAR(6))
- Sample data instances (i.e. rows) for Person table as shown in Table 7

Table 7: Sample data instances for Person table

Row #	id	Name	Gender
1	A01	Edgar S.	Male
2	A02	Khalid A.	Male
3	A03	Amirah. A	Female
…			…
100	A100	Reem A.	Female

To avoid repetition, the data instances above shows only the first three rows and the last row (as indicated by Row #) in a table with 100 rows. Now, assume that the gender column contains only one of two possible values: Male, Female; although the gender column can contain Null, we consider Null as "value missing" and thus, treat it as non-value. Through data analysis, namely the *Detect Sparse-Column Values* method, the gender column in the Person table is designated as a sparse-column with a Sparse Confidence Ratio (SCR) of 0.98; with the "Number of Distinct Values" and "Number of Non-Null Values" set to 2 and 100 respectively. For this example, we are assuming the threshold parameters were set to max_distinct=10 and min_SCR=0.9.

Figure 8 shows how sparse-column values are represented in OWL (lines 21-35) and how SCR is reported (line 20). Based on the SCR value, an ontology modeler can now accept or reject the definition for the *gender* property.

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <rdf:RDF … >
03 ...
04 <owl:Class rdf:ID="HR.Address">
05  <rdfs:subClassOf rdf:resource="owl;Thing"/>
06  <rdfs:comment>HR.Address UDT.</rdfs:comment>
07 </owl:Class>
08 <owl:DatatypeProperty rdf:ID="HR.Address.line_1">
09  <rdfs:domain rdf:resource="#HR.Address" />
10  <rdfs:range  rdf:resource="&xsd;string"/>
11  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
12 </owl:DatatypeProperty>
13 <owl:DatatypeProperty rdf:ID="HR.Address.city">
14  <rdfs:domain rdf:resource="#HR.Address" />
15  <rdfs:range  rdf:resource="&xsd;string"/>
16  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
17 </owl:DatatypeProperty>
18 <owl:Class rdf:ID="HR.US_Address">
19  <rdfs:subClassOf rdf:resource="#HR.Address"/>
20  <rdfs:comment>HR.US_Address UDT.</rdfs:comment>
21 </owl:Class>
22 <owl:DatatypeProperty rdf:ID="HR.US_Address.state">
23  <rdfs:domain rdf:resource="#HR.US_Address" />
24  <rdfs:range  rdf:resource="&xsd;string"/>
25  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
26 </owl:DatatypeProperty>
27 <owl:DatatypeProperty rdf:ID="HR.US_Address.zip-code">
28  <rdfs:domain rdf:resource="#HR.US_Address" />
29  <rdfs:range  rdf:resource="&xsd;string"/>
30  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
31 </owl:DatatypeProperty>
32 ...
```

Figure 8. OWL representation for Example IV.

3.5. Summary

The proliferation of information across various organizations increases the demand for technologies that can facilitate the integration and sharing of information. One technology that can address this need is semantic computing, which relies on the use of ontology models to provide an explicit and formal description for the information being exchanged. With data models such as the RDB and ORDB models sharing conceptions similar to those found in ontology models, and with the pervasiveness of the RDB and ORDB models in organizations today, my approach reuses information already captured in the RDB and ORDB models by automating the translation of these data models into an ontology model. Providing a framework to automate the translation of both data models into OWL ontology models can be of great benefit to organizations deploying semantic computing based solutions.

CHAPTER 4: Candidate Symmetric and Candidate Transitive Binary Relations

4.1. Introduction

In the real world, similar distinguished relationships between pairs of objects can be captured using a binary relation. In its generic form, a pair (or tuple) in such relation states that one object is associated with another in a unidirectional manner (e.g. object *x* depends on object *y*). Depending on the semantics of the relationship, a binary relation can have properties that allow inferring additional associations between objects without having to explicitly state them. Some of the common properties include transitivity and symmetry [32][79]. These properties are important because they can have implications on the manner we interpret and process tuples in these relations.

In information systems, handling of these properties is often delegated to the (knowledgeable-) user, application logic, or database layer. While the database layer is the most logical place to handle these properties, current database standards (e.g. SQL) and DBMS implementations do not provide direct and elegant methods to address them. Furthermore, database modeling representations, such as Entity-Relationship (ER), do not provide the constructs (or grammar) to model binary relations that are characterized as transitive or symmetric [30][72][73]. Attentive database modelers and developers

therefore rely on design guidelines and development methods when encountering such relations [13]. Failing to address these properties properly can lead to storing duplicate information, which can result in unnecessary storage cost and data inconsistency.

Unlike data modeling representations, ontology languages such as OWL provide the constructs for annotating binary relations as symmetric and/or transitive [88]. Identifying such properties in an ontology can lead to a model that is closely aligned with the subject area it describes and thus, eliminate misinterpretation among stakeholders.

This chapter describes the methods DM2ONT employ when identifying *candidate symmetric* and *candidate transitive* binary relations. These methods are based on heuristic database modeling techniques and data analysis and are therefore considered as *suggestions*, hence the use of the term *candidate symmetric* and *candidate transitive* binary relations. The aim here is to assist ontology modelers by providing them with a small set of binary relations to review for symmetry and/or transitivity with the domain experts rather than having them review each and every binary relation in the domain.

4.2. Examples of Symmetric and Transitive Binary Relations

There are plenty of examples in the real world where binary relations (e.g. r on domain D or more formally $r \subseteq D \times D$) can be symmetric, transitive, both symmetric and transitive, or neither. Table 8 shows examples of these binary relations along with their usual properties (i.e. symmetric, transitive, etc.) and cardinalities. Appendix B contains sample relations (relation schemas and instances).

Table 8: Examples of binary relations along with their properties and cardinality.

Name of the Binary Relation	*Properties of the Binary Relation*	*Cardinality of the Binary Relation*
Married-to (Person)	Symmetric & Non-Transitive	One-to-One (1:1)
Next (Queued items)	Transitive & Non-Symmetric	One-to-One (1:1)
Manages (Staff)	Transitive & Non-Symmetric	One-to-Many (1:M)
Knows (Person) , Follows (Twitter)	Non-Symmetric & Non-Transitive	Many-to-Many (N:M)
Borders (Territory)	Symmetric & Non-Transitive	Many-to-Many (N:M)
Composed-Of (Product), Dependency (Tasks)	Non-Symmetric & Transitive	Many-to-Many (N:M)
Siblings (Person), Live-with (Person)	Symmetric & Transitive	Many-to-Many (N:M)

4.3. Basic Definitions

This chapter relies on the RDB definitions introduced in an earlier section, namely *3.3.1.1 Preliminary RDB Definitions and Notation*. In this section, symmetry and transitivity are defined as properties of a binary relation *r* on domain *D* (i.e. $r \subseteq D \times D$).

Definition 4.1: A binary relation *r* on domain *D* is symmetric if:

$$\forall x, y \in D, (x, y) \in r \Rightarrow (y, x) \in r.$$

Definition 4.2: A binary relation *r* on domain *D* is transitive if:

$$\forall x, y, z \in D, ((x, y) \in r) \wedge ((y, z) \in r) \Rightarrow (x, z) \in r.$$

Generally speaking, binary relations can be represented using a vertex-edge graph (or graph in short) from the graph theory. In this context, the components in each tuple/pair are represented using vertices with an edge connecting them. In other words, the tuple (x , y) in a binary relation r can be represented as a graph with two vertices, namely x and y, and an edge connecting these two vertices.

4.4. Motivation

Ontologies describe subject areas in an explicit and formal manner with the objective of facilitating activities such as information discovery and integration (Motivation section in Chapter 1). In identifying candidate symmetric and candidate transitive binary relations in relational databases, one can expect the ontologies generated from these databases with these constructs to be more explicit about and aligned with the subject-area. Such an ontology can aid in inferring facts about a subject-area beyond those that are clearly stated in a knowledge-base. These inferred facts can be crucial to solving business problems or identifying business opportunities.

In the homeland security domain, a typical scenario involves an agent who is trying to identify all the people who live with a suspect who is under investigation. Using an ontology that identifies the binary relation (e.g. object property in OWL) *lives-with* as both transitive and symmetric, and a knowledge-base that contains instances stating that *Person-1* lives with *Person-2* and *Person-2* lives with *Suspect-1*, an application can use symmetry to infer that *Suspect-1* lives-with *Person-2* and *Person-2* lives-with *Person-1*, and use transitivity to infer that *Suspect-1* lives with *Person-1.*

4.5. Assumptions

Generally speaking, the methods used in DM2ONT to identify *candidate symmetric* and *candidate transitive* binary relations expect the source database to conform to the following two conventions (these are formally defined in section 4.6):

1. The database has been created using common design guidelines/patterns for symmetric and/or transitive binary relations [13][31][32], and
2. The database did not store tuples that are implied by symmetry or transitivity.

Following these conventions in databases not only aids in avoiding data inconsistency, but also reduces storage cost. Data consistency and storage cost are affected by storing redundant information such as that implied by symmetric or transitive binary relations; e.g. storing (y, x) given (x, y) in a symmetric binary relation or (x, z) given (x, y) and (y, z) in a transitive binary relation. Storing such data can lead to data inconsistency, which is caused by deleting one tuple and not the other. To overcome this problem, database modelers and developers rely on both modeling and development techniques to allow users to retrieve tuples even when they are not explicitly stored in the database. These techniques range from using views and stored procedures in the database tier to developing business logic in the application tier. However, since analyzing programming logic to discover symmetry and transitivity is infeasible -- due to the wide spectrum of programming languages and paradigms in use nowadays, I opted for retrieving and analyzing data in databases, which can be accomplished using a standardized language (i.e. SQL).

It is worth noting though that in some cases, organizations may opt for duplicating information, thus incurring storage cost, in order to gain better performance. The rules given here for identifying *candidate symmetric* and *candidate transitive* relations would not be applicable to such cases.

4.6. Identifying Candidate Symmetric and Candidate Transitive Binary Relations

DM2ONT identifies *candidate symmetric* and *candidate transitive* binary relations using heuristic data modeling and data analysis. This section formally defines the general assumptions discussed in Section 4.5 and the methods used in DM2ONT to identify *candidate symmetric* and *candidate transitive* binary relations in relational databases.

4.6.1. Formal Definitions

Definition 4.3: Given a binary relation *r* on domain *D* and property *P*, we say that *r* is *minimal* w.r.t. *P* if the following holds:

$(\neg \exists\, t \in r)\ (t \in (r - \{t\})^{+}_{P})$, where $(r - \{t\})^{+}_{P}$ is the *P*-closure of *r* without tuple *t*.

This definition states that the binary relation *r* is said to be minimal if we can not find a tuple *t* in relation *r* when the *P*-closure (e.g., symmetric or transitive closure) of *r* without *t* will yield *t*. In other words, a relation *r* is considered minimal w.r.t property *P* if *r* does not include *any* tuple that is implied by *P* (e.g. symmetry and transitivity).

As discussed in section 3.3.1.1, and for optimization and maintenance reasons, a database designer might choose to merge a binary relation -- one that maps to a binary relationship set in the ER model -- with one of the relations/tables that are involved in the relationship (i.e. its domain or co-domain) if the binary relation is one-to-one (1:1) or many-to-one (M:1). Nevertheless, these binary relations can still be computed at runtime using a simple query: $r_1 = \pi_{pk_2, fk_2} (\sigma_{is\_not\_null(fk_2)} (r_2))$,

where r_2 is a relation instance over relation schema R_2, pk_2 is the primary key column(s) in R_2, fk_2 is a foreign key columns(s) in R_2, $pk_2 \cap fk_2 = \varnothing$, and r_1 columns are contained/embedded in r_2 as pk_2 and fk_2.

The following section introduces the formal definitions for the common structural/schema patterns used to represent symmetric and transitive binary relations in RDB. These are followed by the definitions for the *Symmetric Encoding* and *Transitive Encoding*.

Definition 4.4 - Pattern 1 (P_1): Given a relation schema R_2, a relation instance r_2 over R_2, and integrity constraints pk_2 as the primary key of R_2 (i.e. pk_2 = pkey(R_2)) and fk_2 as a foreign key in R_2 with reference to pk_2 (i.e. $fk_2 \in$ fkey(R_2) and refpk(fk_2) = pk_2). Let D_{pk2} be the projection of pk_2 values (i.e. $D_{pk2} = \pi_{pk_2}(r_2)$), r_1 be a binary relation on D_{pk2} (i.e. $r_1 \subseteq D_{pk2} \times D_{pk2}$), and R_1 be the schema of r_1 (i.e. R_1={(A_{1a}:D_{pk2}), (A_{1b}:D_{pk2})}). We say r_1 conforms to Pattern 1 (P_1 in short) if the following holds:

- $r_1 = \pi_{pk_2, fk_2} (\sigma_{is\_not\_null\ (fk_2)} (r_2))$.

We note Pattern 1 as a structure $P_1 = (R_2, r_2, R_1, r_1, IC_1)$, where $IC_1 = (pk_2, fk_2)$ is a tuple with integrity constraints relevant to P_1. Examples of P_1 binary relations include Married-to (Person), Next (Queued Items), and Manage (Staff). This pattern is used with 1:1 or M:1 binary relations. Appendix B contains samples of these relations (both schemas and instances).

Definition 4.5 - Pattern 2 (P_2): Given relation schemas R_1 and R_2, relation instances r_1 over R_1 and r_2 over R_2, and integrity constraints pk_1 as the primary key of R_1 (i.e. pk_1 = pkey(R_1)), pk_2 as the primary key of R_2 (i.e. pk_2 = pkey(R_2)), and fk_1 and fk_2 as foreign keys in R_1 with reference to pk_2 (i.e. $fk_1, fk_2 \in$ fkey(R_1) and refpk(fk_1) = refpk(fk_2) = pk_2). Let D_{pk2} be the projection of pk_2 values (i.e. $D_{pk2} = \pi_{pk_2}(r_2)$), and A_{1a} and A_{1b} be sets of attributes corresponding to fk_1 and fk_2 respectively (i.e. $A_{1a} = fk_1$, $A_{1b} = fk_2$). We say r_1 is a binary relation (on D_{pk2}) that conforms to Pattern 2 (P_2 in short) if the following holds:

i) $(\{fk_1 , fk_2\} = fkey(R_1)) \wedge (fk_1 \cup fk_2 = attrib(R_1))$, and

ii) $fk_1 \cup fk_2 = pk_1$.

We note Pattern 2 as a structure: $P_2 = (R_2, r_2, R_1, r_1, IC_2)$, where $IC_2 = (pk_1, pk_2, fk_1, fk_2)$ is a tuple with integrity constraints relevant to P_2. Examples of P_2 binary relations include Follows (Twitter), Boarders (Territory), Composed-of (Products), and Siblings (Person). This pattern is used mostly with N:M binary relations but can also be used as an alternative to P_1 when property 'ii' in Definition 4.5 is adjusted. Appendix B contains samples of such relations.

Definition 4.6 - Pattern 3 (P_3): Given relation schemas R_2 and R_3, relation instances r_2 over R_2 and r_3 over R_3, and integrity constraints pk_2 as the primary key of R_2 (i.e. pk_2 = pkey(R_2)), pk_3 as the primary key of R_3 (i.e. pk_3 = pkey(R_3)), and fk_2 as a foreign key in R_2 with reference to pk_3 (i.e. $fk_2 \in$ fkey(R_2) and refpk(fk_2) = pk_3). Let D_{pk2} and D_{pk3} be the projections of pk_2 and pk_3 values respectively, r_1 be a binary relation from D_{pk2} to D_{pk3} (i.e. $r_1 \subseteq D_{pk2} \times D_{pk3}$), and R_1 be the schema of r_1 (i.e. R_1= {(A_{1a}:D_{pk2}), (A_{1b}:D_{pk3})}). We say r_1 conforms to Pattern 3 (P_3 in short) if the following holds:

i) $fk_2 \cap pk_2 = \varnothing$,

ii) attrib(R_3) – $pk_3 = \varnothing$, and

iii) $r_1 = \pi_{pk_2, fk_2} (\sigma_{is\_not\_null\ (fk_2)} (r_2))$.

We note Pattern 3 as a structure: $P_3 = (R_3, r_3, R_2, r_2, R_1, r_1, IC_3)$, where IC_3= (pk_2, pk_3, fk_2) is a tuple with the integrity constraints relevant to P_3. Examples of P_3 binary relations include Siblings and Live-with (Person). Note that this pattern is an alternative to P_2 for N:M binary relations that are both symmetric and transitive. Moreover, it is worth noting here that Pattern 3 can also be used for a category-like relation (e.g. when R_3 is indexed by a category-name and has no other attributes). While it is uncommon to have a category-like relation without a category-id as its index in addition to a category-name attribute, I acknowledge that such relation when encountered will be identified wrongly as pattern 3 (i.e. a false-positive). Appendix B contains a sample of a valid Pattern 3 relation.

For some of the patterns we introduced, identifying the schema structure is not sufficient for classifying the binary relation associated with the pattern as candidate symmetric and/or candidate transitive. Specifically, P_1 and P_2 binary relations require further data analysis to determine whether they are candidate symmetric and/or candidate transitive. For P_3 however, we rely on the schema structure described in various RDB design sources from the literature when identifying such relation as candidate symmetric and transitive; data in P_3 does not have an identifiable signature that can confirm whether the relation is candidate symmetric/transitive.

In the rest of this chapter, we focus on the data analysis methods used with P_1 and P_2 binary relations. Prior to introducing the methods, we formally establish the connection between Symmetric and Transitive binary relations in the real-world with what we term as Symmetric Encoding and Transitive Encoding. These definitions are followed by lemmas describing properties of RDB binary relations related to P_1 and P_2.

Definition 4.7 (Symmetric Encoding): Given a real world (original) *1:1* or *N:M* symmetric binary relation r_{rw}. We define the *Symmetric Encoding* of r_{rw} as its implementation in RDB using a binary relation r_{rdb} that is minimal w.r.t. symmetry and conforms to pattern P_1 or P_2.

A real world binary relation can have various minimal encodings w.r.t. symmetry. For example, given $r_{rw} = \{(x,y),(y,x)\}$, r_{rw} can be encoded in RDB as $r_{rdb1} = \{(x,y)\}$ or $r_{rdb2} = \{(y,x)\}$. We denote the set of all possible symmetric encodings of r_{rw} using pattern P_y (where $y \in \{1, 2\}$) as $Encoding_{Symm}(r_{rw}, P_y)$.

Definition 4.8 (Acyclic Binary Relation): Given a binary relation *r* on domain *D*, we say *r* is *acyclic* if for any n > 1, the following holds:

$$(\neg \exists\, x_1,x_2,...,x_n \in D)\ (\ ((x_j, x_{j+1}) \in r \text{ for all } j \in \{1, .., n-1\}) \wedge ((x_n, x_1) \in r)\ )$$

Definition 4.9 (Non-Trivial Transitive Binary Relation): A transitive binary relation *r* on domain *D* is *non-trivial* if the following holds:

$$(\exists\, x_1, x_2, x_3 \in D)\ ((x_1, x_2), (x_2, x_3), (x_1, x_3) \in r\ )$$

In definition 4.9, we use the term non-trivial transitive binary relations to exclude transitive binary relations that contain non-chained vertices/nodes. For instance, while a binary relation containing only two tuples (x_1, x_2) and (x_3, x_4) is considered transitive from a mathematical standpoint, such relation makes less sense in the real world.

Definition 4.10 (Transitive Encoding): Given a real world *acyclic non-trivial* transitive binary relation r_{rw}. We define the *Transitive Encoding* of r_{rw} as its implementation in RDB using a binary relation r_{rdb} that is minimal w.r.t. transitivity and conforms to pattern P_1 or P_2.

We denote the set of all possible transitive encodings of r_{rw} using pattern P_y (where $y \in \{1, 2\}$) as $Encoding_{Trans}\ (r_{rw}\ ,\ P_y)$.

Lemma 4.1 (Transitive Closure): Given $r_{rw} = (r_1)^{+}_{Trans}$

$$(\forall\ (x, y) \in r_{rw})\ ((\ \exists\ x_1 = x, x_2, \ldots, x_{n-1}, x_n = y)\ (\ n > 1 \wedge (x_1, x_2), \ldots, (x_{n-1}, x_n) \in r_1)\ ).$$

The following lemmas (4.2 to 4.11) relate symmetric and transitive encodings to patterns P_1 and P_2.

Lemma 4.2 (Pattern 1 & Symmetric Encoding - 1): Given an RDB binary relation r_1 that conforms to P_1 (i.e. r_1 in $P_1 = (R_2, r_2, R_1, r_1, IC_1)$ from definition 4.4), and given a real world 1:1 symmetric binary relation r_{rw} from definition 4.7. If $r_1 \in \text{Encoding}_{\text{Symm}}(r_{rw}, P_1)$, then:

i) r_1 is 1:1, and

ii) $(\forall x, y, z \in D_{pk2})\ (\ ((x , y) \in r_1) \Rightarrow\ ((y , x) \notin r_1)\ \wedge (y , z) \notin r_1)\ )$.

Note that property (ii) is necessary to ensure that the binary relation r_1 continues to be 1:1 when its symmetric closure is computed (i.e. r_{rw} is 1:1).

Proof:

To prove lemma 4.2, we have to establish that 'i' and 'ii' hold given $r_1 \in \text{Encoding}_{\text{Symm}}(r_{rw}, P_1)$.

a) To prove 'r_1 is 1:1' ('i' in lemma 4.2), the following must hold:

$$(\ (\forall x, y, z \in D_{pk2})\ (\ ((x, y) \in r_1) \wedge (y \neq z) \Rightarrow\ ((x , z) \notin r_1)\ ) \wedge \quad [1]$$

$$(\forall x, y, z \in D_{pk2})\ (\ ((x, y) \in r_1) \wedge (x \neq z) \Rightarrow\ ((z , y) \notin r_1)\ )\ ) \quad [2]$$

- The symmetric relations r_{rw} is 1:1 and r_1 conforms to P_1 are given in lemma 4.2,
- From definition 4.4 (P_1), $r_1 \subseteq D_{pk2} \times D_{pk2}$ where $D_{pk2} = \pi_{pk_2}(r_2)$,
- For any $(x, y) \in r_1 \Rightarrow (x, y) , (y, x) \in r_{rw,}$ [3]

  $\Rightarrow (x, \ldots, y) \in r_2$ with x being a value in pk_2 of r_2 for some tuple *t*. Now,

  - Assume there is $(x, z) \in r_1\ \wedge z \neq y \Rightarrow (x, \ldots, z) \in r_{2,}$

    $\Rightarrow pk_2$ column(s) in r_2 will contain duplicate values, (x, …, y), (x, …, z), which

is a contradiction to primary key definition,

$\Rightarrow (x, z) \in r_1$ is false, $\Rightarrow$ [1] holds

- assume $(z, y) \in r_1 \Rightarrow (z, y), (y, z) \in r_{rw}$

  $\Rightarrow (z, y), (y, z), (x, y), (y, x) \in r_{rw}$ (from [3] above),

  $\Rightarrow r_{rw}$ is N:M, which is false (from lemma 4.2, r_{rw} is 1:1),

  $\Rightarrow (z, y) \in r_1$ is false, $\Rightarrow$ [2] holds

$\therefore$ 'r_1 is 1:1' (i.e. 'i' in lemma 4.2) holds. [4]

b) To prove 'ii' in lemma 4.2, i.e.

$$(\forall x, y, z \in D_{pk2}) (((x, y) \in r_1) \Rightarrow ((y, x) \notin r_1) \wedge ((y, z) \notin r_1))$$

- For any $(x, y) \in r_1 \Rightarrow (x, y) , (y, x) \in r_{rw}$

  - assume there is $(y, x) \in r_1 \Rightarrow (x, y), (y, x) \in r_1$,

    $\Rightarrow r_1 \in \text{Encoding}_{\text{Symm}} (r_{rw} , P_1)$ is false (i.e. r_1 is not minimal) $\Rightarrow$ Contradiction.

  - assume there is $(y, z) \in r_1 \wedge (x \neq z) \Rightarrow (y, z) , (z, y) \in r_{rw} \wedge (x, y) , (y, x) \in r_{rw}$

    $\Rightarrow r_{rw}$ is not 1:1 $\Rightarrow$ Contradiction, because r_{rw} is given as 1:1.

  $\therefore$ 'ii' holds. [5]

$\therefore$ Lemma 4.2 hold. (from [4] and [5])

- End of Proof.

Lemma 4.3 (Pattern 1 & Symmetric Encoding - 2): Given an RDB binary relation r_1 that conforms to P_1 (i.e. r_1 in $P_1 = (R_2, r_2, R_1, r_1, IC_1)$ from definition 4.4) and satisfies:

i) r_1 is 1:1, and

ii) $(\forall x, y, z \in D_{pk2}) (((x , y) \in r_1) \Rightarrow ((y , x) \notin r_1) \wedge (y , z) \notin r_1))$.

For such r_1, we can construct a 1:1 symmetric relation r_{con} such that $r_1 \in \text{Encoding}_{\text{Symm}}(r_{con}, P_1)$.

Unlike a real world binary relation r_{rw}, a binary relation r_{con} is mathematically constructed and might not make sense in the real world.

Proof:

In this proof, we need to establish that:

a) r_{con} is symmetric,

b) $r_1 \in \text{Encoding}_{\text{Symm}}(r_{con}, P_1)$, and

c) r_{con} is 1:1.

- By definition, r^+ is the symmetric closure of binary relation r on S if 1) r^+ is symmetric, 2) $r \subseteq r^+$, and 3) r^+ is a subset of any other symmetric relation on S that includes r. For any r, we can construct r^+.
- We construct r_{con} as the symmetric closure of r_1 (i.e. $r_{con} = (r_1)^+_{Sym}$).
- Proof of "r_{con} is symmetric" is established by definition; i.e. the symmetric closure of any binary relation is always symmetric.

  $\therefore$ 'a' holds. [1]
- To prove $r_1 \in \text{Encoding}_{\text{Symm}}(r_{con}, P_1)$, we need to establish that r_1 conforms to P_1 and r_1 is minimal w.r.t. symmetry (per definition 4.7).
  - r_1 conforms to P_1 is given in Lemma 4.3,
  - Assume r_1 is not minimal w.r.t. symmetry.

$\Rightarrow \exists$ (y, x) $\in r_1 \mid$ (x, y) $\in r_1$. This is false; It contradicts with property 'ii' of r_1:

$(\forall x, y, z \in D_{pk2})$ (((x , y) $\in r_1$) $\Rightarrow$ ((y , x) $\notin r_1$) $\wedge$ (y , z) $\notin r_1$)).

$\Rightarrow r_1$ is minimal w.r.t. symmetry.

- With r_1 conforming to P_1, $r_{con} = (r_1)^+_{Sym}$, and r_1 is minimal w.r.t. symmetry,

  $\Rightarrow r_1 \in \text{Encoding}_{\text{Symm}}(r_{con} , P_1)$

$\therefore$ 'b' holds. [2]

- To prove "r_{con} is 1:1",
  - It is given that r_1 is 1:1, $(\forall x, y, z \in D_{pk2})$ (((x , y) $\in r_1$) $\Rightarrow$ ((y , x) $\notin r_1$) $\wedge$ (y , z) $\notin r_1$)) and r_{con} is the symmetric closure of r_1.
  - From proof in the previous step (i.e. proof of 'b'), r_1 is minimal w.r.t. symmetry.
  - $\forall$ (x, y) $\in r_1$, (x, y) , (y, x) $\in r_{con}$
  - Assume r_{con} is N:M,

    $\Rightarrow (\exists$ (x, y), (x, z) $\in r_{con} \wedge (y \neq z)) \wedge$ [3]

    $(\exists$ (a, c), (b, c)$\in r_{con} \wedge (a \neq b))$ [4]

    $\Rightarrow$ If [3] or [4] is false, r_{con} can not be N:M. [5]

    $\Rightarrow$ For [3], $\exists$ (x, y), (x, z) $\in r_{con} \wedge (y \neq z) \Rightarrow$ (y, x), (z, x) $\in r_{con}$,

    $\Rightarrow$ Since $r_{con} = (r_1)^+_{Sym}$ and r_1 is minimal, r_1 will contain one pair from each of the following sets: {(x, y), (y, x)} and {(x, z), (z, x)}

    $\Rightarrow r_1 \in$ { {(x, y), (x, z)}, {(x, y), (z, x)}, {(y, x), (x, z)}, {(y, x), (z, x)} }

    $\Rightarrow r_1 =$ {(x, y), (x, z)} is false. It contradicts property 'i' of r_1,

$\Rightarrow r_1 = \{(x, y), (z, x)\} = \{(z, x), (x, y)\}$ is false. It contradicts property 'ii' of r_1,

$\Rightarrow r_1 = \{(y, x), (x, z)\}$ is false. It contradicts property 'ii' of r_1,

$\Rightarrow r_1 = \{(y, x), (z, x)\}$ is false. It contradicts property 'i' of r_1,

$\Rightarrow$ [3] is false $\Rightarrow$ From [5], "r_{con} is N:M" is false. [6]

- Assume r_{con} is M:1,

$\Rightarrow \exists (x, y), (z, y) \in r_{con} \wedge (x \neq z)$

$\Rightarrow (y, x), (y, z) \in r_{con}$ (because r_{con} is symmetric) $\Rightarrow$ "r_{con} is N:M"

$\Rightarrow$ "r_{con} is M:1" is false. [7]

$\Rightarrow r_{con}$ is not N:M and r_{con} is not M:1 (from [6] and [7]) $\Rightarrow$ r_{con} is 1:1

$\therefore$ 'c' holds. [8]

$\therefore$ Lemma 4.3 hold. (from [1], [2] and [8])

- End of Proof.

Lemma 4.4 (Pattern 1 & Symmetric Encoding - 3): Given an RDB binary relation r_1 that is 1;1 and conforms to P_1 (i.e. r_1 in $P_1 = (R_2, r_2, R_1, r_1, IC_1)$ from definition 4.4). The following are equivalent:

i) $\pi_{A1a}(r_1) \cap \pi_{A1b}(r_1) = \varnothing$

ii) $(\forall x, y, z \in D_{pk2})\ (((x , y) \in r_1) \Rightarrow ((y , x) \notin r_1) \wedge (y , z) \notin r_1))$.

Proof:

To prove that 'i' and 'ii' in Lemma 4.4 are equivalent, we need to establish:

a. 'i' $\Rightarrow$ 'ii'

b. ‘ii’ $\Rightarrow$ ‘i’

- From definition 4.4, R_1={ $(A_{1a} : D_{pk2})$, $(A_{1b} : D_{pk2})$ }

- Proof for ‘i’ $\Rightarrow$ ‘ii’:

  - Given ‘i’, we assume ‘ii’ is false.

  $\Rightarrow \exists (x, y) \in r_1 \wedge \exists (y, x) \in r_1$, or [1]

  $\exists (x, y) \in r_1 \wedge \exists (y, z) \in r_1$ [2]

  - For [1], $\pi_{A_{1a}}(r_1) = \{x, y\}$ and $\pi_{A_{1b}}(r_1) = \{y, x\}$

  $\Rightarrow \{x, y\} \cap \{y, x\} = \{x, y\} \neq \varnothing$, which contradicts ‘i’

  $\Rightarrow$[1] is false. [3]

  - For [2], $\pi_{A_{1a}}(r_1) = \{x, y\}$ and $\pi_{A_{1b}}(r_1) = \{y, z\}$

  $\Rightarrow \{x, y\} \cap \{y, z\} = \{y\} \neq \varnothing$, which contradicts ‘i’

  $\Rightarrow$ [2] is false. [4]

  $\therefore$ ‘i’ $\Rightarrow$ ‘ii’ holds (from [3] and [4]) [5]

- Proof for ‘ii’ $\Rightarrow$ ‘i’:

  - It is given that r_1 is 1:1 (in lemma 4.4)

  - Given ‘ii’, we assume ‘i’ is false.

  $\Rightarrow \pi_{A_{1a}}(r_1) \cap \pi_{A_{1b}}(r_1) \neq \varnothing$ [6]

  $\Rightarrow (\exists\ y)\ (y \in \pi_{A_{1a}}(r_1)) \wedge (y \in \pi_{A_{1b}}(r_1))$

  $\Rightarrow$ *Either* $\exists (x, y), (y, x) \in r_1$, *or* [7]

  $\exists (x, y) , (y, z) \in r_1 \wedge (x \neq z)$ [8]

  $\Rightarrow (x, y), (y, x) \in r_1$ is false; it contradicts ‘ii’

$\Rightarrow$ [7] is false. [9]

$\Rightarrow (x, y) , (y, z) \in r_1$ is false; it contradicts 'ii'

$\Rightarrow$ [8] is false. [10]

$\therefore$ 'ii' $\Rightarrow$ 'i' holds (from [9] and [10]) [11]

$\therefore$ 'i' $\Leftrightarrow$ 'ii' (from [5] and [11])

- End of Proof.

Lemma 4.5 (Pattern 1 & Transitive Encoding - 1): Given an RDB binary relation r_1 that conforms to P_1 (i.e. r_1 in $P_1 = (R_2, r_2, R_1, r_1, IC_1)$ from definition 4.4) and a real world acyclic non-trivial transitive binary relation r_{rw} (definition 4.10). If $r_1 \in \text{Encoding}_{\text{Trans}}$ (r_{rw} , P_1), then:

i) r_1 is acyclic, and

ii) $(\exists x_1, x_2, x_3 \in D_{pk2}) ((x_1, x_2), (x_2, x_3) \in r_1 \wedge (x_1 \neq x_3))$

Proof:

- In this proof, we need to establish that (i) and (ii) are true (given the premise).
- By definition, r^+ is the transitive closure of binary relation r on S if 1) r^+ is transitive, 2) $r \subseteq r^+$, and 3) r^+ is a subset of any other transitive relation on S that includes r. [1]
- r_{rw} is the transitive closure of r_1 (i.e. $r_{rw} = (r_1)^+_{Trans}$). [2]
- To prove 'i',
  - It is given that r_{rw} is acyclic and $r_{rw} = (r_1)^+_{Trans}$.
  - Assume r_1 is not acyclic (i.e. r_1 has a cycle)

    $\Rightarrow \exists (x_1, x_2), \ldots, (x_{n-1}, x_n), (x_n, x_1) \in r_1 \wedge n > 1$

$\Rightarrow \exists (x_1, x_2), \ldots, (x_{n-1}, x_n), (x_n, x_1) \in r_{rw}$ (per [1] and [2], $r_1 \subseteq r_{rw}$)

$\Rightarrow r_{rw}$ has a cycle. Contradiction (r_{rw} is given as acyclic in lemma 4.5)

$\therefore$ 'i' holds [3]

- To prove 'ii',
  - It is given that r_{rw} is acyclic non-trivial transitive relation and $r_{rw} = (r_1)^{+}_{Trans}$ (in lemma 4.5) and r_1 is acyclic (from [3])
  - Assume 'ii' is false,

$\Rightarrow (\neg \exists x_1, x_2, x_3 \in D_{pk2}) ((x_1, x_2), (x_2, x_3) \in r_1 \wedge (x_1 \neq x_3))$ [4]

$\Rightarrow$ Either $(\exists x_1, x_2, x_3 \in D_{pk2}) ((x_1, x_2), (x_2, x_3) \in r_1 \wedge (x_1 = x_3))$, or [5]

$(\neg \exists x_1, x_2, x_3 \in D_{pk2}) ((x_1, x_2), (x_2, x_3) \in r_1)$ [6]

  - For [5], $\Rightarrow \exists (x_1, x_2), (x_2, x_1) \in r_1 \Rightarrow r_1$ is acyclic. Contradiction with 'i'.
  - For [6],
    - It is given that r_{rw} is non-trivial $\Rightarrow \exists (x_1, x_2), (x_2, x_3) \in r_{rw}$
    - From [6] and Lemma 4.1,
      - $(x_1, x_2) \in r_{rw} \Rightarrow (x_1, x_2) \in r_1$. [7a]
      - $(x_2, x_3) \in r_{rw} \Rightarrow (x_2, x_3) \in r_1$. [7b]
      - From [7a] and [7b], $(x_1, x_2), (x_2, x_3) \in r_1$. Contradiction with [6].

$\therefore$ 'ii' holds [8]

$\therefore$ Lemma 4.5 holds (From [3] and [8])

- End of Proof.

Lemma 4.6 (Pattern 1 & Transitive Encoding - 2): Given an RDB binary relation r_1 that conforms to P_1 (i.e. r_1 in $P_1 = (R_2, r_2, R_1, r_1, IC_1)$ from definition 4.4) and satisfies:

i) r_1 is acyclic and

ii) $(\exists\, x_1, x_2, x_3 \in D_{pk2})\ ((x_1, x_2), (x_2, x_3) \in r_1 \wedge (x_1 \neq x_3))$

For such r_1, we can construct an acyclic non-trivial transitive relation r_{con} such that $r_1 \in \text{Encoding}_{\text{Trans}}(r_{con}, P_1)$.

Unlike a real world binary relation r_{rw}, a binary relation r_{con} is mathematically constructed and might not make sense in the real world.

Proof:

- In this proof, we need to establish:

  a) r_{con} is transitive,

  b) r_{con} is non-trivial transitive,

  c) $r_1 \in \text{Encoding}_{\text{Trans}}(r_{con}, P_1)$, and

  d) r_{con} is acyclic.

- We construct r_{con} as the transitive closure of r_1 (i.e. $r_{con} = (r_1)^+_{Trans}$) [1]

- Proof of r_{con} is transitive is established by the definition of transitive closure. i.e., the transitive closure of a binary relation is always transitive.

  $\therefore$ 'a' holds [2]

- To prove r_{con} is non-trivial transitive,

  - It is given that r_1 contains (x_1, x_2), (x_2, x_3) tuples in lemma 4.6 and $r_{con} = (r_1)^+_{Trans}$.

    $\Rightarrow r_1 \subseteq r_{con}$ (definition of transitive closure)

$\Rightarrow (x_1, x_2), (x_2, x_3) \in r_{con} \Rightarrow r_{con}$ is non-trivial transitive.

$\therefore$ 'b' holds [3]

- To prove $r_1 \in \text{Encoding}_{\text{Trans}}(r_{con}, P_1)$, we need to establish that r_1 conforms to P_1 and r_1 is minimal w.r.t. transitivity (definition 4.10).
  - r_1 conforms to P_1 is given in lemma 4.6.
  - For r_1 is minimal w.r.t. transitivity,
    - Assume r_1 is *not* minimal w.r.t. transitivity.

      $\Rightarrow (\exists (x, y) \in r_1) ((r_1)^+_{Trans} = (r_1 - (x, y))^+_{Trans})$

      $\Rightarrow (x, y) \in r_1 \Rightarrow (x, y) \in (r_1)^+_{Trans} \Rightarrow (x, y) \in (r_1 - (x, y))^+_{Trans})$

      $\Rightarrow$ From Lemma 4.1,

      $(\exists x_1 = x, x_2, \ldots, x_n = y) (n > 2 \wedge (x_i, x_{i+1}) \in (r_1 - (x, y)) \wedge 1 \leq i < n)$

      $\Rightarrow \exists (x_1, x_2) , (x_1, x_n) \in r_1$

      $\Rightarrow \exists (x_1, \ldots, x_2) \in r_2$ with x_1 being a value in pk_2 of r_2 for some tuple t.

      $\Rightarrow \exists (x_1, \ldots, x_n) \in r_2$. Contradiction. Values in pk_2 column(s) in r_2 cannot be duplicate (i.e. $\neg \exists (x_1, \ldots, x_2), (x_1, \ldots, x_n) \in r_2$).

      $\Rightarrow r_1$ is minimal w.r.t. transitivity

    $\therefore$ 'c' holds [4]

- To prove r_{con} is acyclic,
  - It is given that r_1 is acyclic, $r_{con} = (r_1)^+_{Trans}$.
  - Assume r_{con} is *not* acyclic (i.e. r_{con} has a cycle)

$\Rightarrow (\exists\ x_1, x_2, \ldots, x_n)\ (\ (n > 1) \wedge ((x_1, x_2), \ldots, (x_{n-1}, x_n), (x_n, x_1) \in r_{con})\ )$

$\Rightarrow (x_i, x_{i+1}) \in r_{con}$

$\Rightarrow\ (\exists\ y_{i\_1} = x_i\ , \ldots, y_{i\_ni} = x_{i+1})\ (\ (n_i > 1) \wedge ((y_{i\_j}, y_{i\_j+1}) \in r_1)\ )$

$\Rightarrow (\ (y_{1\_1}, y_{1\_2}), \ldots, (y_{1\_n1}, y_{2\_1}), \ldots, (y_{n\_1,} y_{n\_2}) \ldots, (y_{n\_nn,} y_{1\_1}) \in r_1\ )$

$\Rightarrow r_1$ has a cycle. Contradiction.

$\therefore$ 'd' holds [5]

$\therefore$ From [2], [3], [4] and [5], lemma 4.6 holds

- End of Proof.

4.6.2. Methods to Identify Candidate Symmetric and Candidate Transitive Binary Relations

Several structural patterns exist for modeling symmetric and/or transitive binary relations. The use of one or another depends on the cardinality and the design choices made by the database designer. In DM2ONT, we identified three structural patterns that are commonly used to encode (implement) real-world symmetric/transitive binary relations in RDB. These patterns were termed Pattern 1, 2 and 3 (or P_1, P_2 and P_3 for short). Once DM2ONT detects these structural/schema patterns, it performs data analysis (if necessary) to classify the binary relations associated with these patterns as *candidate symmetric* and/or *candidate transitive*.

Since data in P_3 binary relations do not exhibit any special characteristics, the schemas associated with P_3 binary relations are declared as candidate symmetric solely based on the schema definition. For P_1 and P_2 however, we perform further analysis

before declaring a P_1/P_2 binary relation schema as candidate symmetric and/or candidate transitive. Figure 9 depicts the *overall* process for determining *candidate symmetry* and *candidate transitivity* for all three patterns.

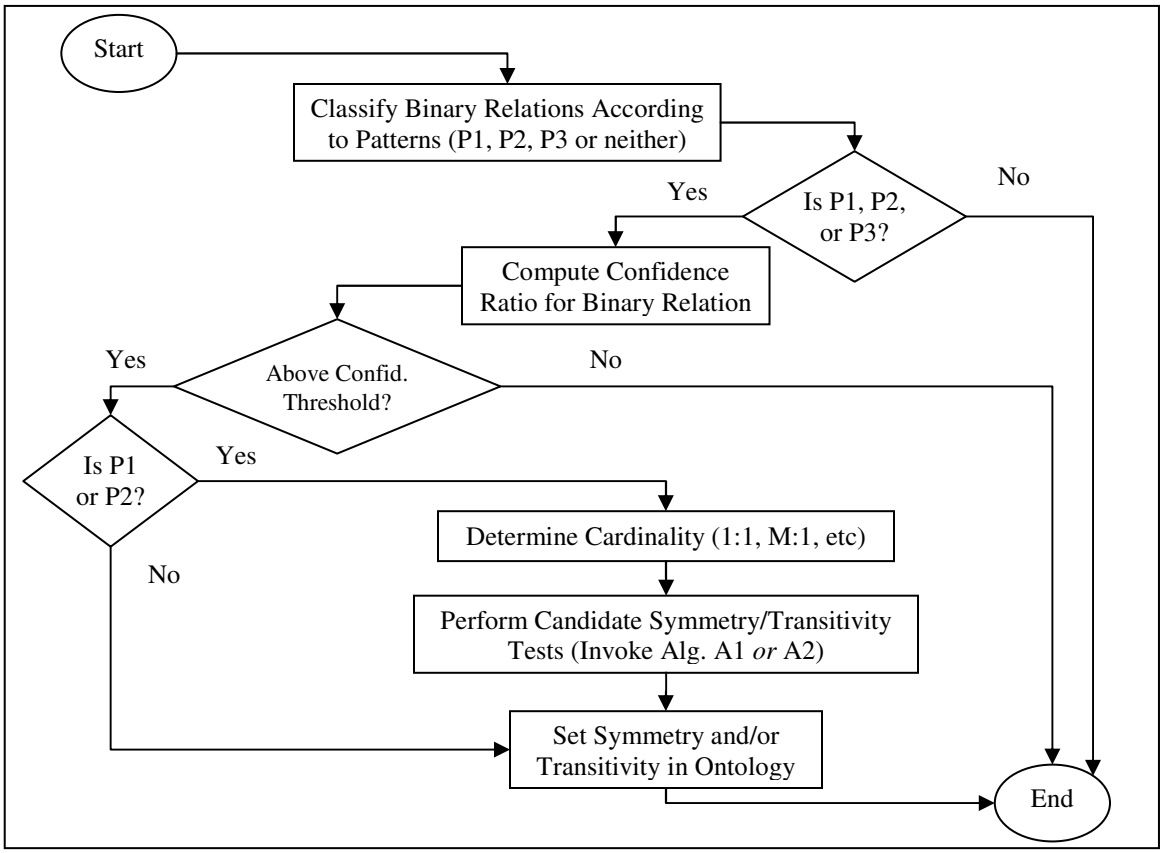


Figure 9: Identifying candidate symmetric and transitive binary relations - Overall process

The following two subsections present the algorithms used in DM2ONT for determining if a P_1 or P_2 binary relation is *candidate symmetric* or *candidate transitive*.

4.6.2.1. Identifying Candidate Symmetry and Candidate Transitivity for Pattern 1

This section presents Algorithm A1, which addresses candidate symmetry and candidate transitivity for binary relations that conform to Pattern 1 (definitions 4.4).

Definition 4.11 (Candidate Symmetric): We say a binary relation r_1 is *Candidate Symmetric* w.r.t. pattern P_y if Algorithm $A_y(P_y, card(r_1))$ returns isCandSymm=True, where $y \in \{1, 2\}$ and P_y is a structure conforming to Pattern y.

Definition 4.12 (Candidate Transitive): We say a binary relation r_1 is *Candidate Transitive* w.r.t. pattern P_y if Algorithm $A_y(P_y, card(r_1))$ returns isCandTrans=True, where $y \in \{1, 2\}$ and P_y is a structure conforming to Pattern y.

Algorithm A1 (Pattern 1 – Candidate Symmetric/Transitive):

```
01  Input: P1 (R2, r2, R1, r1, IC), card (r1)
02  Output: isCandSymm (Boolean), isCandTrans (Boolean)
03  Begin-Steps
04    //R1 (from structure P1) has two sets of attributes: A1a and A1b
05    Let isCandSymm = isCandTrans = false
06    If (card = = '1:1') Then
07       Let result_set1 = π A1a (r1) ∩ πA1b (r1)
08       If ( result_set1  = = ∅ ) Then
09          isCandSymm = true
10       End_If
11    End_If
12    If (isCandSymm = = false ) Then
13       Let isAcyclic = isTrivial = true
14       Let A1a_set = π A1a (r1)
15       Let ei = GetElement(1) (A1a_set)
16       While (ei ≠ null) and (isAcyclic)  Do
17          Let tc_set = π A1b (σA1a = ei (r1))
18          Let ej = GetElement(1) (tc_set)
19          While (ej ≠ null) and (isAcyclic)  Do
20             Let c = GetFirstElement (2) (π A1b (σA1a = ej (r1)))
21             If ( (c ≠ null ) and ((c ∈ tc_set) OR (c = = ei ) ) ) Then
22                isAcyclic = false
23             Else If (c ≠ null ) Then
24                tc_set = tc_set ∪ {c}
25                isTrivial = false
26             End_If_Else
27             ej = GetElement(1) (tc_set)
28          End_While (ej ≠ null…)
29          A1a_set = A1a_set – tc_set
30          ei = GetElement(1) (A1a_set)
31       End_While (ei ≠ null…)
32       If (isAcyclic) and (isTrivial = = false) Then
33          isCandTrans = true
34       End_If
35    End_If   // (isCandSymm = = false )
36  End-Steps
-------------------------
(1) GetElement (): A function that takes a set as an input and returns an element that has not been
    processed or null otherwise. It marks returned element as processed.
(2) GetFirstElement (): A function that returns the first element in the given set or null if the set is empty.
```

Theorem A1-T1: Algorithm A1 with input $P_1(R_2, r_2, R_1, r_1, IC_1)$ and card(r_1) = "1:1" terminates and its output satisfies the following properties:

i) $(\exists\ r_{rw})\ ((r_{rw}$ is 1:1) $\wedge$ (r_{rw} is symmetric) $\wedge$ ($r_1 \in \text{Encoding}_{\text{Symm}}\ (r_{rw}\ , P_1))\ \Rightarrow$

r_1 is Candidate Symmetric, and

ii) r_1 is Candidate Symmetric $\Rightarrow$

$(\exists\ r_{con})\ ((r_{con}$ is 1:1) $\wedge$ (r_{con} is symmetric) $\wedge$ ($r_1 \in \text{Encoding}_{\text{Symm}}(r_{con}\ , P_1))$.

Note that although r_{con} may not make sense in real world, r_{con} can be mathematically constructed.

Proof A1-T1:

Termination Analysis: Algorithm A1 is composed of two sections: lines 5-11 (first section) and lines 12-35 (second section). When invoked, one or both sections will be executed. Thus, asserting that A1 terminates requires a proof that each section terminates.

The first section is executed when r_1 is 1:1. It terminates because it is a sequence of statement without any loop, and the query operators used in line 7 are part of the DBMS with well defined behavior. Therefore, section one will terminate for all inputs.

The second section is executed only if isCandSymm flag is not set to true by the first section. The second section terminates only if the outer loop (lines 16-31) and inner loop (lines 19-28) terminate.

The outer loop is a While loop that terminates *either* when "ei = null" *or* "isAcyclic = False". The former condition in the outer loop does not involve any risk because ei obtains values from the finite set A1a_set before the loop (line 15) and within

the loop (line 30) using the GetElement() function, and GetElement() returns only the A1a_set values that have not been processed in previous iterations; moreover, A1a_set does not expand inside the loop (i.e. no elements are added to A1a_set inside the loop).

The inner loop terminates either when "ej = null" or "isAcycle = False". It terminates as well because ej values are drawn from a finite set (tc_set) using the GetElement() function in line 18 (before the loop) and line 27 (inside the loop). The tc_set is populated before and within the inner loop with values from A1b columns in r_1, which is finite. Therefore, algorithm A1 terminates for any given r_1.

Correctness Analysis: To prove the correctness of algorithm A1, we need to prove that 'i' and 'ii' in theorem A1-T1 hold.

1) In algorithm A1, isCandSymm is set to True *only* in line 9:

   "isCandSymm = True" $\Leftrightarrow$ If-Cond in line 6 is True *and* If-Cond in line 8 is True

   $\Leftrightarrow$ "r_1 is 1:1" and "$\pi_{A_{1a}}(r_1) \cap \pi_{A_{1b}}(r_1) = \varnothing$"

2) From Definition 4.11, "r_1 is Candidate Symmetric" is equivalent to algorithm A1 returning isCandSymm=True.

3) From (1) and (2) above, we obtain

   "r_1 is Candidate Symmetric" $\Leftrightarrow$ "r_1 is 1:1" and "$\pi_{A_{1a}}(r_1) \cap \pi_{A_{1b}}(r_1) = \varnothing$"

4) From Lemma 4.4, we learned:

$(\pi_{A_{1a}}(r_1) \cap \pi_{A_{1b}}(r_1) = \varnothing) \Leftrightarrow (\forall\ x,y,z \in D_{pk2})\ (((x,y) \in r_1) \Rightarrow ((y,x) \notin r_1) \wedge ((y,z) \notin r_1))$.

To prove 'i':

5) From Lemma 4.2, we learned:

$(\exists\ r_{rw})\ ((r_{rw}$ is 1:1) $\wedge$ (r_{rw} is symmetric) $\wedge$ ($r_1 \in \text{Encoding}_{\text{Symm}}\ (r_{rw}\ , P_1))\ \Rightarrow$

$((r_1$ is 1:1) $\wedge\ ((\forall\ x,y,z \in D_{pk2})\ (\ ((x,y) \in r_1) \Rightarrow ((y,x) \notin r_1) \wedge ((y,z) \notin r_1))$

6) From propositions (4) & (5) above, we obtain:

$(\exists\ r_{rw})\ ((r_{rw}$ is 1:1) $\wedge$ (r_{rw} is symmetric) $\wedge$ ($r_1 \in \text{Encoding}_{\text{Symm}}\ (r_{rw}\ , P_1))\ \Rightarrow$

$((r_1$ is 1:1) $\wedge\ (\pi_{A_{1a}}(r_1) \cap \pi_{A_{1b}}(r_1) = \varnothing))$

7) From (3) and (6):

$(\exists\ r_{rw})\ ((r_{rw}$ is 1:1) $\wedge$ (r_{rw} is symmetric) $\wedge$ ($r_1 \in \text{Encoding}_{\text{Symm}}\ (r_{rw}\ , P_1))\ \Rightarrow$

"r_1 is Candidate Symmetric"

8) $\therefore$ 'i' holds

<u>To prove 'ii':</u>

9) From Lemma 4.3, we learned:

$(r_1$ is 1:1) $\wedge\ (\forall\ x,y,z \in D_{pk2})\ ((x,y) \in r_1 \Rightarrow ((y,x) \notin r_1 \wedge (y,z) \notin r_1))\ \Rightarrow$

$(\exists\ r_{con})\ ((r_{con}$ is 1:1) $\wedge$ (r_{con} is symmetric) $\wedge\ (\ r_1 \in \text{Encoding}_{\text{Symm}}\ (r_{con}\ , P_1))$

10) From propositions (4) & (9) above, we obtain:

$(\ (r_1$ is 1:1) $\wedge\ (\pi_{A_{1a}}(r_1) \cap \pi_{A_{1b}}(r_1) = \varnothing)\ ) \Rightarrow$

$(\exists\ r_{con})\ ((r_{con}$ is 1:1) $\wedge$ (r_{con} is symmetric) $\wedge\ (\ r_1 \in \text{Encoding}_{\text{Symm}}\ (r_{con}\ , P_1))$

11) From (3) and (10), we obtain:

r_1 is Candidate Symmetric $\Rightarrow$

$(\exists\ r_{con})\ ((r_{con}$ is 1:1) $\wedge$ (r_{con} is symmetric) $\wedge$ ($r_1 \in \text{Encoding}_{\text{Symm}}\ (r_{con}\ , P_1))$

12) $\therefore$ 'ii' holds

13) $\therefore$ Algorithm A1 is correct w.r.t. to 'i' and 'ii' (per 8 and 12)

- End of Proof.

Lemma 4.7a (Pattern 1, Algorithm A1 isAcyclic - 1): Given an RDB binary relation r_1 that conforms to P_1 (i.e. r_1 in $P_1 = (R_2, r_2, R_1, r_1, IC_1)$ from definition 4.4) and $(\exists (x_1, x_2), (x_2, x_3)\ldots, (x_{n-1}, x_n), (x_n, x_1) \in r_1 \wedge (n > 2))$ with $x_1, x_2, \ldots, x_n$ as distinct elements, and given algorithm A1 with isCandSymm = false in Line 12 and ei = x_1 in line 15. We make the following assertion in the inner While-Loop header (line 19) for iteration step $k \in \{1, \ldots, n - 2\}$ and $n > 2$:

ej = $x_{k+1} \wedge$ tc_set = $\{x_2, \ldots, x_{k+1}\}$ with all elements marked as processed $\wedge$ isAcyclic= true

Proof:

- From definition 4.4, r_1 is 1:1 or M:1

  $\Rightarrow (\forall x_1, x_2, x_3 \in D_{pk2})\ (\ ((x_1, x_2) \in r_1) \wedge (x_2 \neq x_3)) \Rightarrow ((x_1, x_3) \notin r_1)\ )$ [1]

- It is given $\exists (x_1, x_2), (x_2, x_3)\ldots, (x_{n-1}, x_n) \in r_1 \wedge (n > 2)$ [2]

- Basis Step (k=1):

  - isAcyclic = true (line 13)

  - tc_set = $\pi_{A1b}(\sigma_{A1a = ei}(r_1))$ (line 17)

  $\Rightarrow$ (From [1], $(x_1, x_2) \in r_1$ in [2], and given ei = x_1), $\sigma_{A1a = x1}(r_1) = \{(x_1, x_2)\}$

  $\Rightarrow \pi_{A1b}\{(x_1, x_2)\} = \{x_2\} \Rightarrow$ tc_set = $\{x_2\} = \{x_{k+1}\}$ (line 17)

  $\Rightarrow$ ej = GetElement(tc_set) = x_2 and mark it as processed (line 18)

  $\Rightarrow$ In line 19, ej = $x_2 = x_{k+1}$, tc_set = $\{x_2\} = \{x_{k+1}\}$ with x_2 marked as processed,

isAcyclic = true

$\therefore$ Lemma 4.7a holds for step k=1 [3]

- Induction Step: We assume for step k ($1 \leq k < n-2$), the following holds:

ej = $x_{k+1} \wedge$ tc_set =$\{x_2, .., x_{k+1}\}$ with all elem. marked as processed $\wedge$ isAcyclic = true [4]

- We prove for step k+1:

- tc_set = $\{x_2, \ldots, x_{k+1}\}$

$\Rightarrow$ (From [1], $(x_{k+1}, x_{k+2}) \in r_1$ in [2], and ej= x_{k+1}), $\sigma_{A_{1a} = xk+1}(r_1)) = \{(x_{k+1}, x_{k+2})\}$

$\Rightarrow \pi_{A_{1b}}\{(x_{k+1}, x_{k+2})\} = \{x_{k+2}\}$

$\Rightarrow$ c = GetFirstElement($\{x_{k+2}\}$) $\Rightarrow$ c = x_{k+2} (line 20)

$\Rightarrow$ c = $x_{k+2} \neq$ null

$\Rightarrow$ c = $x_{k+2} \notin$ tc_set (tc_set = $\{x_2, \ldots, x_{k+1}\}$ and $x_2, \ldots, x_{k+1}, x_{k+2}$ are distinct)

$\Rightarrow$ c = $x_{k+2} \neq$ ei (ei = x_1 and $k \geq 1$ are given, and x_1 and x_{k+2} are distinct)

$\Rightarrow$ "If $((c \neq \text{null}) \wedge ((c \in \text{tc\_set}) \vee (c == \text{ei})))$" Condition is false. (line 21)

$\Rightarrow$ "(c $\neq$ null)" is true in IF-Cond. (line 23)

$\Rightarrow$ tc_set = $\{x_2, \ldots, x_{k+1}\} \cup \{c\}$ (line24) $\Rightarrow$ tc_set = $\{x_2, \ldots, x_{k+1}, x_{k+2}\}$

- (From [4], $x_2, .., x_{k+1}$ in tc_set are marked as processed but x_{k+2} is not),

$\Rightarrow$ ej = GetElement (tc_set) = x_{k+2} and markt it as processed (line27)

$\Rightarrow$ End of inner While-Loop iteration (line 28) with

ej =$x_{k+2} \wedge$ tc_set =$\{x_2, \ldots, x_{k+2}\}$ with all elem. marked as processed $\wedge$ isAcyclic remained true.

$\Rightarrow$ In the next iteration (line 19):

ej $= x_{(k+1)+1}$ $\wedge$ tc_set $= \{x_2, \ldots, x_{(k+1)+1}\}$ with all elem. marked as processed $\wedge$ isAcyclic = true. [5]

$\therefore$ Lemma 4.7a holds (from [3] and [5])

- End of Proof.

Lemma 4.7b (Pattern 1, Algorithm A1, isAcyclic – 2): Given an RDB binary relation r_1 that conforms to P_1 (i.e. r_1 in $P_1 = (R_2, r_2, R_1, r_1, IC_1)$ from definition 4.4) and given algorithm A1 with isCandSymm = false in Line 12:

r_1 has trivial cycle $(\exists (x_1, x_2), (x_2, x_1) \in r_1) \Rightarrow$ "isAcyclic = false" in line 32 of A1

Proof:

- From definition 4.4, r_1 is 1:1 or M:1

$\Rightarrow (\forall x_1, x_2, x_3 \in D_{pk2})\ (\ ((x_1, x_2) \in r_1) \wedge (x_2 \neq x_3)) \Rightarrow ((x_1, x_3) \notin r_1)\ )$ [1]

- It is given that $\exists (x_1, x_2), (x_2, x_1) \in r_1$ and isCandSymm = false in Line 12 of A1.

$\Rightarrow$ the IF-Cond block (lines 12-35) will be executed

$\Rightarrow$ isAcyclic = true (Line 13)

$\Rightarrow$ A1a_set $= \pi_{A1a}(r_1)$ (Line 14)

$\Rightarrow \{x_1, x_2\} \subseteq$ A1a_set (from $(x_1, x_2), (x_2, x_1) \in r_1$) [2]

- Outer While-Loop (line 16-31) will end either when:
  - "isAcyclic = false", or [3]
  - ei = null [4]
- For [3], $\Rightarrow$ isAcyclic = false in line 32.

$\therefore$ Lemma 4.7b holds for [3] [5]

- For [4],

⇒ All A1a_set elements were processed one by one (including x_1, x_2 from [2]):

- For $x_1 \in$ A1a_set, either

- $ei = x_1$ in outer loop (line 15 or 30), or [6]

- $x_1 \in$ tc_set from a previous iteration (line 29) [7]

⇒ For [6], $\exists$ $ei = x_1$ in one of the iterations in outer While-Loop block (16-31)

⇒ $tc\_set = \pi_{A1b}(\sigma_{A1a = x1}(r_1))$ (line 17)

⇒ (From [1] and $(x_1, x_2) \in r_1$ in [2]), $\sigma_{A1a = x1}(r_1) = \{(x_1, x_2)\}$

⇒ $\pi_{A1b}\{(x_1, x_2)\} = \{x_2\} \Rightarrow tc\_set = \{x_2\}$

⇒ $ej = GetElement(tc\_set) = x_2$ (line 18 and in inner Loop block 19-28)

⇒ $c = GetFirstElement(\pi_{A1b}(\sigma_{A1a = x2}(r_1))$ (line 20)

⇒ (From [1] and $(x_2, x_1) \in r_1$ in [2]), $\sigma_{A1a = x2}(r_1) = \{(x_2, x_1)\}$

⇒ $\pi_{A1b}\{(x_2, x_1)\} = \{x_1\} \Rightarrow c = x_1$

⇒ "$(c \neq null) \wedge (c == ei)$" is true in IF-Cond. (line 21)

⇒ isAcyclic = false (line 22).

⇒ Algorithm A1 exits inner While-Loop (Lines 19-28)

⇒ Algorithm A1 exits outer While-Loop (Lines 16-31)

⇒ isAcyclic = false in Line 32.

$\therefore$ Lemma 4.7b holds for [6] [8]

⇒ For [7], $x_1 \in$ tc_set from a previous iteration

⇒ $\exists (y_1, y_2), \ldots, (y_{n-1}, y_n), (y_n, x_1), (x_1, x_2), (x_2, x_1) \in r_1 \wedge n > 1$ [9]

$\Rightarrow$ $\exists$ ei = y_1 in an iterations of the outer While-Loop block (16-31)

$\Rightarrow$ From [9] and Lemma 4.7a, after n+1 iterations in the inner loop (k= n+1), ej = x2, tc_set = $\{y_2, \ldots, y_n, x_1, x_2\}$ with all elements marked as processed $\wedge$ isAcyclic= true, (line 19)

$\Rightarrow$ c = GetFirstElement ($\pi_{A_{1b}}$ ($\sigma_{A_{1a} = x2}$ (r_1)) (line 20)

$\Rightarrow$ (From [1] and (x_2, x_1) $\in$ r_1 in [9]), c = x_1

$\Rightarrow$ "(c $\neq$ null) $\wedge$ (c $\in$ tc_set)" is true in IF-Cond. (line 21)

$\Rightarrow$ isAcyclic = false (line 22).

$\Rightarrow$ Algorithm A1 exits inner While-Loop (Lines 19-28)

$\Rightarrow$ Algorithm A1 exits outer While-Loop (Lines 16-31)

$\Rightarrow$ isAcyclic = false in Line 32.

$\therefore$ Lemma 4.7b holds for [7] [10]

$\therefore$ Lemma 4.7b holds (from [5], [8] and [10])

- End of Proof.

Lemma 4.7c (Pattern 1, Algorithm A1 isAcyclic - 3): Given an RDB binary relation r_1 that conforms to P_1 (i.e. r_1 in P_1 = (R_2, r_2, R_1, r_1, IC_1) from definition 4.4) and given algorithm A1 with isCandSymm = false in Line 12. The following are equivalent:

i) isAcyclic is true in line 32 of A1 given r_1, and

ii) r_1 is acyclic.

Proof:

To establish that 'i' and 'ii' in Lemma 4.7c are equivalent, we need to prove:

a. ‘i’ $\Rightarrow$ ‘ii’

b. ‘ii’ $\Rightarrow$ ‘i’

- From definition 4.4, R_1={ $(A_{1a} : D_{pk2})$, $(A_{1b} : D_{pk2})$ }

- From definition 4.4, r_1 is 1:1 or M:1

$\Rightarrow (\forall\ x_1, x_2, x_3 \in D_{pk2})\ (\ ((x_1, x_2) \in r_1) \wedge (x_2 \neq x_3)) \Rightarrow ((x_1, x_3) \notin r_1)\ )$ [1]

- Proof for ‘i’ $\Rightarrow$ ‘ii’:

  - Given ‘i’ (i.e. “isAcyclic = True” in line 32 of A1),

    - Assume ‘ii’ is false (i.e. r_1 has cycle(s))

    $\Rightarrow \exists\ (x_1, x_2), \ldots, (x_{n-1}, x_n), (x_n, x_1)\ \in r_1 \wedge (n > 1)$ [2]

    $\Rightarrow$ Either: $\exists\ (x_1, x_2), (x_2, x_1) \in r_1$, or [3]

    $\exists\ (x_1, x_2), (x_2, x_3)\ldots, (x_{n-1}, x_n), (x_n, x_1) \in r_1 \wedge (n > 2) \wedge x_1, x_2, \ldots, x_n$ are distinct [4]

    - Initially, isAcyclic = True (line 13 in A1)

    - For [3],

      - From Lemma 4.7b:

      r_1 has trivial cycle ($\exists\ (x_1, x_2), (x_2, x_1) \in r_1$) $\Rightarrow$ “isAcyclic = false” in line 32 of A1.

      $\Rightarrow$ “isAcyclic = false”. Contradiction. [5]

    - For [4],

      - It is given in [4] that $(x_1,x_2),\ldots, (x_n, x_1) \in r_1 \wedge (n > 2) \wedge x_1, x_2, \ldots, x_n$ are distinct

      $\Rightarrow$ In Algorithm A1, cycle (x_n, x_1) will be detected either when

        - ei = x_1, or [6]

        - $x_1 \in$ tc_set (from a previous iteration) [7]

- For [6], let ei = x_1 in line 15

  - From Lemma 4.7a:

$\exists (x_1, x_2), (x_2, x_3)\ldots, (x_{n-1}, x_n), (x_n, x_1) \in r_1 \wedge (n > 2) \wedge x_1, x_2, \ldots, x_n$ are distinct $\Rightarrow$ ej = $x_{k+1} \wedge$ tc_set $=\{x_2, \ldots, x_{k+1}\}$ with all elements marked as processed $\wedge$ isAcyclic= true in the inner While-Loop header (line 19-28) for iteration step $k \in \{1, \ldots, n - 2\}$ for $n > 2$. [8]

  - For the last iteration in the inner While-Loop, iteration "n – 1", let k = n – 1 :

  $\Rightarrow$ From [8], ej = $x_{k+1} = x_{(n-1)+1} = x_n$ (line 19)

  $\Rightarrow$ From [8], tc_set = $\{x_2, \ldots, x_{k+1}\} \Rightarrow$ tc_set = $\{x_2, \ldots, x_n\}$ (line 19)

  $\Rightarrow$ c = GetFirstElement ($\pi_{A1b} (\sigma_{A1a = xn} (r_1))$ (line 20)

    $\Rightarrow$ From [1] and [4], $(x_n, x_1) \in r_1 \Rightarrow \pi_{A1b} (\sigma_{A1a = xn} (r_1)) = x_1$

  $\Rightarrow$ c = x_1 (line 20) $\Rightarrow c \neq$ null $\wedge$ c = e_i

  $\Rightarrow$ "((c $\neq$ null) $\wedge$ ((c $\in$ tc_set) $\vee$ (c = = ei)))" is true in IF-Cond. (line 21)

  $\Rightarrow$ isAcyclic = false (line 22).

  $\Rightarrow$ Algorithm A1 exits inner While-Loop (Lines 19-28)

  $\Rightarrow$ Algorithm A1 exits outer While-Loop (Lines 16-31)

  $\Rightarrow$ isAcyclic = false in Line 32. Contradiction. [9]

- For [7], $x_1 \in$ tc_set from a previous iteration

  $\Rightarrow \exists (y_1, y_2), \ldots, (y_{my-1}, y_{my}), (y_{my}, x_1), (x_1, x_2), \ldots, (x_{nx-1}, x_{nx}), (x_{nx}, x_1) \in r_1 \wedge$ $(m_y > 1) \wedge (n_x > 2) \wedge y_j$ and x_j are distinct [10]

  $\Rightarrow \exists (a_{y1}, a_{y2}), \ldots, (a_{ymy}, a_{x1}), (a_{x1}, a_{x2}), \ldots, (a_{xnx-1}, a_{xnx}), (a_{xnx}, a_{x1}) \in r_1$

$\Rightarrow \exists (a_1, a_2), (a_2, a_3), \ldots, (a_{n-1}, a_n), (a_n, a_i) \in r_1 \wedge (n > 2) \wedge (1 < i < n)$ [11]

- Assume $\exists$ ei = a_1 in an iterations of the outer While-Loop block (16-31)

$\Rightarrow$ From [8] (Lemma 4.7a) and [11], and ei = a_1 :

(ej = $a_{k+1} \wedge$ tc_set =$\{a_2, \ldots, a_{k+1}\}$ with all elements marked as processed $\wedge$ isAcyclic= true) in the inner Loop header (line 19) for iteration step $k \in \{1, \ldots, n - 2\}$ and $\wedge$ n > 2. [12]

$\Rightarrow$ For the last iteration in the inner loop, iteration "n – 1", let k = n - 1:

$\Rightarrow$ From [12], ej = $a_{k+1} = a_{(n-1)+1} = a_n$ (line 19)

$\Rightarrow$ From [12], tc_set = $\{a_2, ..., a_{k+1}\} = \{a_2, ..., a_{(n-1)+1} = a_n\}$ (line 19)

$\Rightarrow$ c = GetFirstElement ($\pi_{A1b} (\sigma_{A1a = an} (r_1))$ (line 20)

$\Rightarrow$ From [1] and [11], $(a_n, a_i) \in r_1 \Rightarrow \pi_{A1b} (\sigma_{A1a = an} (r_1)) = a_i \ (1 < i < n)$

$\Rightarrow$ c = a_i (line 20) $\Rightarrow c \neq$ null $\wedge c \in$ tc_set (for $a_1 < a_i < a_n$)

$\Rightarrow$ "((c $\neq$ null) $\wedge$ ((c $\in$ tc_set) $\vee$ (c = = ei)))" is true in IF-Cond. (line 21)

$\Rightarrow$ isAcyclic = false (line 22).

$\Rightarrow$ Algorithm A1 exits inner While-Loop (Lines 19-28)

$\Rightarrow$ Algorithm A1 exits outer While-Loop (Lines 16-31)

$\Rightarrow$ isAcyclic = false in Line 32. Contradiction. [13]

$\therefore$ 'i' $\Rightarrow$ 'ii' holds (from [5], [9] and [13]) [14]

- Proof for 'ii' $\Rightarrow$ 'i':

- Given 'ii' (i.e. r_1 is acyclic),

$\Rightarrow \forall$ n > 1,

$(\neg \exists\ x_1,x_2,..., x_n \in D_{pk2})\ (\ (\ (x_j, x_{j+1}) \in r_1$ for all $j \in \{1, ..,n-1\}\ ) \wedge ((x_n,x_1) \in r_1)\ )$

$\Rightarrow (\forall\ n > 1)\ (\neg \exists\ (x_1, x_2), \ldots, (x_{n-1}, x_n), (x_n, x_1)\ \in r_1\ )\ \wedge$ [15]

$(\forall\ n > 2)\ (\neg \exists\ (x_1, x_2), (x_2, x_3), \ldots, (x_{n-1}, x_n), (x_n, x_i) \in r_1\ \wedge (\ 1 < i < n)$ [16]

- Initially, isAcyclic = True (line 13 in A1)

- Assume 'i' is false (i.e. "isAcyclic = False" in line 32 in A1)

$\Rightarrow$ Algorithm A1 executed the "IF (isCandSymm = = false) Then" block (lines 12-35) and A1 reached the "isAcyclic = False" statement (line 22).

- It is given in Lemma 4.7c that "isCandSymm = false" in line 12

- "isAcyclic = False" is reached when: [17]

  - The outer While-Loop (lines 16-31) is executed at least once, [17a]

  - The inner While-Loop (lines 19-28) is executed at least once, and [17b]

  - "If ((c ≠ null) and ((c ∈ tc_set) OR (c = = ei))) Then" is True (line 21) [17c]

- For [17a], A1 iterates over the outer While-Loop at least once if $r_1 \neq \varnothing$.

- For [17b], A1 iterates over the inner While-Loop at least once if $r_1 \neq \varnothing$.

- For [17c], the IF-Cond (line 21) is reached in one or more iterations in inner loop

  - Let us consider the step in which we reach the statement in [17]

  - $\exists$ A1a_set = $\pi_{A_{1a}}(r_1)$ (line 14)

  - $\exists$ ei ∈ A1a_set, $\exists$ tc_set such that A1 (line 15-31)

    1) initially sets tc_set to "$\pi_{A_{1b}}(\sigma_{A_{1a} = ei}(r_1))$", and (line 17)

    2) iteratively adds x_j to tc_set for every x_i in tc_set, (x_i, x_j) in r_1, $x_j \notin$ tc_set and $x_j \neq$ ei. (line 19-28)

$\Rightarrow$ tc_set will contain all vertices/nodes directly or indirectly connected to ei until we reach the last vertex or detect a cycle (e.g. if $r_1 = \{(x_1, x_2), (x_2, x_3), (x_3, x_4), (x_5, x_6)\}$, tc_set for ei=$x_1$ will be $\{x_2, x_3, x_4\}$)

- $(\exists\ ej,\ c)\ ((ej \in tc\_set) \wedge (c = GetFirstElement\ (\pi_{A1b}(\sigma_{A1a = ej}(r_1)))))$ (line 20)
- From [17c], "isAcyclic = False" when the following IF-Condition is true:

  "((c ≠ null) and ((c ∈ tc_set) OR (c = = ei)))" (line 21)

  $\Rightarrow$ either $(ei = x_1 \wedge ej = x_n \wedge c = x_i \wedge (x_1 \in A1a\_set) \wedge (x_n, x_i \in tc\_set))$, or [18]

  $(ei = x_1 \wedge ej = x_n \wedge c = x_1 \wedge (x_1 \in A1a\_set) \wedge (x_n \in tc\_set))$ [19]

  - For [18],

    $\Rightarrow (\exists\ n, i)\ ((n > 2) \wedge (1 < i < n) \wedge (\exists\ (x_1, x_2), (x_2, x_3), \ldots, (x_{n-1}, x_n), (x_n, x_i) \in r_1))$

    $\Rightarrow$ Contradiction with [16]. [20]

  - For [19],

    $\Rightarrow (\exists\ n > 1)\ (\exists\ (x_1, x_2), \ldots, (x_{n-1}, x_n), (x_n, x_1) \in r_1)$

    $\Rightarrow$ Contradiction with [15]. [21]

$\therefore$ 'ii' $\Rightarrow$ 'i' holds (from [20] and [21]) [22]

$\therefore$ 'i' $\Leftrightarrow$ 'ii' (from [14] and [22])

- End of Proof.

Lemma 4.8 (Pattern 1, Algorithm A1 isTrivial): Given an RDB binary relation r_1 that conforms to P_1 (i.e. r_1 in $P_1 = (R_2, r_2, R_1, r_1, IC_1)$ from definition 4.4) and is acyclic, and given algorithm A1 with isCandSymm = false in Line 12. The following are equivalent:

i) "isTrivial = false" in line 32 of A1 given r_1, and

ii) $(\exists\, x_1,x_2, x_3 \in D_{pk2})\ (\ (x_1, x_2), (x_2, x_3) \in r_1\ \wedge (x_1 \neq x_3)\ )$

Proof:

To establish that 'i' and 'ii' in Lemma 4.8 are equivalent, we need to prove:

a. 'i' $\Rightarrow$ 'ii'

b. 'ii' $\Rightarrow$ 'i'

- From definition 4.4, R_1={ $(A_{1a} : D_{pk2})$, $(A_{1b} : D_{pk2})$ }

- From definition 4.4, r_1 is 1:1 or M:1 (i.e. $(x_1, x_2) \in r_1 \wedge (x_2 \neq x_3) \Rightarrow (x_1, x_3) \notin r_1$) [1]

- Proof for 'i' $\Rightarrow$ 'ii':

  - Given 'i' (i.e. "isTrivial = False" in line 32 of A1),

    - isTrivial is set to False only in line 25 in A1.

    - Initially, isAcyclic = true and isTrivial = true (line 13 in A1)

    - Assume 'ii' is false, $\Rightarrow (\neg\, \exists\, x_1,x_2, x_3 \in D_{pk2})\ ((x_1,x_2), (x_2,x_3) \in r_1\ \wedge (x_1 \neq x_3)))$

      $\Rightarrow$ Either $(\forall\, x_1,x_2, x_3 \in D_{pk2})\ ((x_1, x_2), (x_2, x_3) \in r_1 \Rightarrow (x_1 = x_3))$), or [2]

      $(\neg\, \exists\, (x_1, x_2), (x_2, x_3) \in r_1)$ [3]

    - For [2],

      $\Rightarrow (\forall\, x_1,x_2, x_3 \in D_{pk2})\ ((x_1, x_2), (x_2, x_3) \in r_1 \Rightarrow (x_1 = x_3))$

      $\Rightarrow \exists\, (x_1, x_2), (x_2, x_1) \in r_1 \Rightarrow$ r_1 is acyclic. Contradiction. [4]

    - For [3],

      $\Rightarrow\ (\forall\, x, y , z \in D_{pk2})\ (\ (x, y) \in r_1 \Rightarrow (y, x) \notin r_1 \wedge (y, z) \notin r_1)$

      $\Rightarrow\ (x_1, y_1), \ldots, (x_n, y_n) \in r_1 \Rightarrow (y_1, a_1), \ldots, (y_n, a_n) \notin r_1 \wedge a_1, \ldots, a_n \in D_{pk2}$

      $\Rightarrow\ \{x_1, x_2, \ldots, x_n\} \subseteq$ A1a_set (line 14)

      $\Rightarrow$ Given r_1 is acyclic, outer loop iterates over all elem. in A1a_set, including x_1

$\Rightarrow$ ei = x_1 in an iteration in outer While-Loop (line 16-31) [5]

$\Rightarrow$ tc_set = $\{y_1\}$ (line 17)

$\Rightarrow$ ej = y_1 in iteration 1 in inner While-Loop (line 19-28)

$\Rightarrow$ c = null (line 20)

$\Rightarrow$ "$((c \neq null) \wedge (c \in tc\_set \vee c == ei)$" is false in IF-Cond. (line 21-23)

$\Rightarrow$ "$(c \neq null)$" is false in IF-Cond. (line 23-26)

$\Rightarrow$ Exit inner While-Loop (line 19-28) [6]

$\Rightarrow$ Another iteration of outer loop is repeated (steps [5] to [6]) for every x_i in A1a_Set ($1 < i \leq n$) with ei = x_i, tc_set = $\{y_i\}$, $e_j = y_i$. In every iteration, c = null.

$\Rightarrow$ "isTrivial = false" is never reached (line 25)

$\Rightarrow$ Exit outer While-Loop (line 16-31)

$\Rightarrow$ isTrivial = true (line 32). Contradiction. [7]

$\therefore$ 'i' $\Rightarrow$ 'ii' holds (from [4] and [7]) [8]

- Proof for 'ii' $\Rightarrow$ 'i':
  - Given 'ii' (i.e. $(\exists x_1, x_2, x_3 \in D_{pk2})\ ((x_1, x_2), (x_2, x_3) \in r_1 \wedge (x_1 \neq x_3))$)
    - Initially, isAcyclic =True and isTrivial = True outside both loops (line 13 in A1)
    - Assume 'i' is false (i.e. "isTrivial = True" in line 32 in A1)
      - Given r_1 is acyclic, outer loop will iterate over all elem. in A1a_set, including x_1
      - A1a_set $\supseteq \{x_1, x_2, \ldots\}$ (line 14)

      $\Rightarrow$ ei = x_1 in an iteration in outer While-Loop (lines 16-31)

      $\Rightarrow$ tc_set = $\{x_2\}$ (line 16)

$\Rightarrow$ ej = x_2 in iteration 1 in inner While-Loop (lines 19-28)

$\Rightarrow$ c = x_3 (line 20)

$\Rightarrow$ "(c $\neq$ null) $\wedge$ (c $\in$ tc_set $\vee$ c == ei)" is false in IF-Cond. (lines 21-23)

$\Rightarrow$ (c $\neq$ null) is true in Else_IF-Cond (line23-26)

$\Rightarrow$ tc_set = {x_2, x_3} (line 24),

$\Rightarrow$ "isTrivial = false". (line 25)

$\Rightarrow$ Repeat inner while-Loop for every ej in tc_set. (lines 19-28)

$\Rightarrow$ Repeat outer while-Loop for every ei in A1a_set. (lines 16-31)

$\Rightarrow$ We exit outer while-Loop with isTrivial = false. Contradiction

$\therefore$ 'ii' $\Rightarrow$ 'i' holds [10]

$\therefore$ 'i' $\Leftrightarrow$ 'ii' (from [9] and [10])

- End of Proof.

Theorem A1-T2: Algorithm A1 with input $P_1(R_2, r_2, R_1, r_1, IC_1)$ and card(r_1) = "1:1" or "M:1" terminates and its output satisfies the following properties:

i) $(\exists\ r_{rw})$ ((r_{rw} is acyclic) $\wedge$ (r_{rw} is non-trivial) $\wedge$ (r_{rw} is transitive) $\wedge$

$(r_1 \in \text{Encoding}_{\text{Trans}}(r_{rw}, P_1)) \Rightarrow r_1$ is Candidate Transitive, and

ii) r_1 is Candidate Transitive $\Rightarrow$

$(\exists\ r_{con})$ ((r_{con} is acyclic) $\wedge$ (r_{con} is non-trivial) $\wedge$ (r_{con} is transitive) $\wedge$

$(r_1 \in \text{Encoding}_{\text{Trans}}(r_{con}, P_1))$.

Note that although r_{con} may not make sense in real world, r_{con} can be mathematically constructed.

Proof A1-T2:

Termination Analysis: See Termination Analysis in Proof A1-T1.

Correctness Analysis: To prove the correctness of algorithm A1 w.r.t. theorem A1-T2, we need to prove "i" and "ii" in Theorem A1-T2.

1) From Definition 4.12, "r_1 is Candidate Transitive" is equivalent to algorithm A1 returning isCandTrans = True.

2) From algorithm A1, *isCandTrans* is set to True *only* in line 33. Therefore,

   "isCandTrans = True" $\Leftrightarrow$ "isCandSymm=False" (line 12) and

   "isAcyclic = True" (line 32) and "isTrivial = False" (line 32)

3) From (1) and (2) above, we obtain

   "r_1 is Candidate Transitive" $\Leftrightarrow$ "isAcyclic=True" $\wedge$ "isTrivial=false" $\wedge$

   "isCandSymm=False"

4) From Lemma 4.7c, we learned:

   "isAcyclic = True" in A1 Line 32 given r_1 $\Leftrightarrow$ r_1 is acyclic.

5) From Lemma 4.8, we learned:

   "isTrivial = false" in A1 Line 32 given r_1 is Acyclic $\Leftrightarrow$

   $(\exists\, x_1, x_2, x_3 \in D_{pk2})\ (\ (x_1, x_2), (x_2, x_3) \in r_1 \wedge (x_1 \neq x_3)\ )$.

6) From [3], [4] and [5], we obtain:

"r_1 is Candidate Transitive" $\Leftrightarrow$ "r_1 is acyclic" $\wedge$

$((\exists\ x_1,x_2, x_3 \in D_{pk2})\ (\ (x_1, x_2), (x_2, x_3) \in r_1\ \wedge (x_1 \neq x_3)\ )\ )$ $\wedge$ "isCandSymm=False".

7) From A1, "isCandSym = False" only if

$$(r_1 \text{ is } \textit{not } 1{:}1) \vee (\ (r_1 \text{ is } 1{:}1) \wedge ((\pi_{A_{1a}}(r_1) \cap \pi_{A_{1b}}(r_1) \neq \varnothing)$$

8) From Lemma 4.4, we learned:

$(\pi_{A_{1a}}(r_1) \cap \pi_{A_{1b}}(r_1) = \varnothing) \Leftrightarrow (\forall\ x,y,z \in D_{pk2})\ (((x,y) \in r_1) \Rightarrow ((y,x) \notin r_1) \wedge ((y,z) \notin r_1))$

$\Rightarrow ((\pi_{A_{1a}}(r_1) \cap \pi_{A_{1b}}(r_1) \neq \varnothing) \Leftrightarrow (\ (\exists\ x,y,z \in D_{pk2})\ (\ (x,y), (y,x) \in r_1 \vee (x,y), (y,z) \in r_1)\ )$

9) From [7] and [8], we obtain "isCandSym = False" if:

$(r_1 \text{ is } \textit{not } 1{:}1) \vee (\ (r_1 \text{ is } 1{:}1) \wedge (\ (\exists\ x,y,z \in D_{pk2})\ (\ (x,y), (y,x) \in r_1 \vee (x,y), (y,z) \in r_1\ )\ )$

10) From [6] and [9], we obtain

"r_1 is Candidate Transitive" $\Leftrightarrow$

$(r_1$ is acyclic$) \wedge (\exists\ x_1,x_2, x_3 \in D_{pk2})\ (\ (x_1, x_2), (x_2, x_3) \in r_1\ \wedge (x_1 \neq x_3)\ )\ \wedge$

$((r_1 \text{ is } \textit{not } 1{:}1) \vee (\ (r_1 \text{ is } 1{:}1) \wedge (\ (\exists\ x,y,z \in D_{pk2})\ (\ (x,y), (y,x) \in r_1 \vee (x,y), (y,z) \in r_1))\ )\ )$

$\Leftrightarrow$

$((r_1$ is acyclic$) \wedge (\exists\ x_1,x_2,x_3 \in D_{pk2})\ (\ (x_1,x_2), (x_2,x_3) \in r_1 \wedge (x_1 \neq x_3)\ )\ \wedge (r_1 \text{ is } \textit{not } 1{:}1)\ ) \vee$

$((r_1$ is acyclic$) \wedge (\exists\ x_1,x_2,x_3 \in D_{pk2})\ (\ (x_1,x_2), (x_2,x_3) \in r_1\ \wedge (x_1 \neq x_3)\ )\ \wedge (r_1 \text{ is } 1{:}1) \wedge$

$(\ (\exists\ x,y,z \in D_{pk2})\ (\ (x,y), (y,x) \in r_1 \vee (x,y), (y,z) \in r_1\ )\ )$

11) We assert that $(r_1$ is acyclic$) \Rightarrow (\ (\neg \exists\ x, y \in D_{pk2})\ (\ (x, y), (y, x) \in r_1\ )\ )$

12) We assert that:

$(\exists\ x_1,x_2,x_3 \in D_{pk2})\ ((x_1,x_2), (x_2,x_3) \in r_1 \wedge (x_1 \neq x_3)) \Leftrightarrow (\exists\ x,y,z \in D_{pk2})\ ((x,y), (y,z) \in r_1)$

13) From [10], [11] and [12], we obtain:

"r_1 is Candidate Transitive" $\Leftrightarrow$

$((r_1 \text{ is acyclic}) \wedge (\exists\, x_1,x_2,x_3 \in D_{pk2})\ (\ (x_1,x_2), (x_2,x_3) \in r_1 \wedge (x_1 \neq x_3)\ ) \wedge (r_1 \text{ is } \textit{not}\ 1{:}1)\ ) \vee$

$((r_1 \text{ is acyclic}) \wedge (\exists\, x_1,x_2,x_3 \in D_{pk2})\ (\ (x_1,x_2), (x_2,x_3) \in r_1\ \wedge (x_1 \neq x_3)\ ) \wedge (r_1 \text{ is } 1{:}1)\ )$

$\Leftrightarrow$

$(r_1 \text{ is acyclic}) \wedge (\exists\, x_1,x_2,x_3 \in D_{pk2})\ (\ (x_1,x_2), (x_2,x_3) \in r_1\ \wedge (x_1 \neq x_3)\ ) \wedge$

$((r_1 \text{ is } \textit{not}\ 1{:}1) \vee (r_1 \text{ is } 1{:}1)\ )$

$\Leftrightarrow\ (r_1 \text{ is acyclic}) \wedge (\exists\, x_1,x_2,x_3 \in D_{pk2})\ (\ (x_1,x_2), (x_2,x_3) \in r_1\ \wedge (x_1 \neq x_3)\ )$

<u>To prove 'i':</u>

14) From Lemma 4.5, we learned:

$(\exists\, r_{rw})\ ((r_{rw} \text{ is acyc.}) \wedge (r_{rw} \text{ is non-trivial}) \wedge (r_{rw} \text{ is trans.}) \wedge (r_1 \in \text{Encoding}_{\text{Trans}}\ (r_{rw}\ , P_1))$

$\Rightarrow\ (r_1 \text{ is acyclic}) \wedge (\exists\, x_1,x_2,x_3 \in D_{pk2})\ (\ (x_1,x_2), (x_2,x_3) \in r_1\ \wedge (x_1 \neq x_3)\ )$

15) From [13] and [14], we obtain:

$(\exists\, r_{rw})\ ((r_{rw} \text{ is acyc.}) \wedge (r_{rw} \text{ is non-triv.}) \wedge (r_{rw} \text{ is trans.}) \wedge (r_1 \in \text{Encoding}_{\text{Trans}}\ (r_{rw}\ , P_1))$

$\Rightarrow\ r_1$ is Candidate Transitive

$\therefore$ "i" holds

<u>To prove 'ii':</u>

16) From Lemma 4.6, we learned:

$(r_1 \text{ is acyclic}) \wedge (\exists\, x_1,x_2,\ x_3 \in D_{pk2})\ ((x_1,\ x_2), (x_2,\ x_3) \in r_1 \wedge (x_1 \neq x_3)) \Rightarrow$

$(\exists\, r_{con})\ ((r_{con} \text{ is acyc.}) \wedge (r_{con} \text{ is non-triv.}) \wedge (r_{con} \text{ is trans.}) \wedge (r_1 \in \text{Encoding}_{\text{Trans}}(r_{con}\ , P_1))$

17) From [13] and [16], we obtain:

"r_1 is Candidate Transitive" $\Rightarrow$

($\exists\ r_{con}$) ((r_{con} is acyc.) $\wedge$(r_{con} is non-triv.) $\wedge$(r_{con} is trans.) $\wedge$($r_1 \in \text{Encoding}_{\text{Trans}}(r_{con}, P_1)$)

$\therefore$ "ii" holds

18) $\therefore$ Algorithm A1 is correct w.r.t. to 'i' and 'ii' (per [15] and [17])

- End of Proof.

4.6.2.2. Identifying Candidate Symmetry and Candidate Transitivity for Pattern 2

This section presents Algorithm A2, which addresses candidate symmetry and candidate transitivity for binary relations that conform to Pattern 2 (definitions 4.5).

Algorithm A2 (Pattern 2 – Candidate Symmetric & Transitive):

01 **Input:** P_2 (R_2, r_2, R_1, r_1, IC_2), card (r_1)
02 **Output:** isCandSymm (Boolean), isCandTrans (Boolean)
03 **Begin-Steps**
04 //R_1 (from Pattern 2) has two sets of attributes: A_{1a} and A_{1b}
05 Let isCandSymm = isCandTrans = false
06 Let result_set = $r_1 \bowtie ((r_1.A_{1a} = tx.A_{1b})$ and $(r_1.A_{1b} = tx.A_{1a}))\ \rho\ tx\ (r_1)$
07 If (result_set = = ∅) Then
08 isCandSymm = true
09 Let isTransMin = isAcyclic = isTrivial = true
10 isAcyclic = CheckAcyclic(1) (r_1)
11 If (isAcyclic) Then
12 Let A1a_set = $\pi_{A_{1a}}$ (r_1)
13 Let ei = GetElement(2) (A1a_set)
14 While (ei ≠ null) and (isTransMin) Do
15 Let processed_tuples = $\sigma_{A_{1a} = ei}$ (r_1)
16 Let ei_direct = tc_set = $\pi_{A_{1b}}$ (processed_tuples)
17 While ((isTransMin) and ((∃ (b,c) ∈ r_1) (b ∈ tc_set ∩ A1a_set) and ((b, c) ∉ processed_tuples)) Do
18 processed_tuples = processed_tuples ∪ {(b, c)}
19 If (c ∈ ei_direct) Then
20 isTransMin = false
21 Else
22 tc_set = tc_set ∪ {c}
23 isTrivial = false
24 End_If_Else
25 End_While //((isTransMin) and …)
26 ei = GetElement(2) (A1a_set)
27 End_While (ei ≠ null and …)
28 If (isTransMin) and (isTrivial = = false) Then
29 isCandTrans = true
30 End_If
31 End_If //(isAcyclic)
32 End_If //(result_set = = ∅)
33 **End-Steps**

(1) CheckAcyclic (): A function that returns true if the given binary relation is acyclic and false otherwise.
(2) GetElement (): A function that takes a set as an input and returns an element that has not been processed or null otherwise. It marks returned element as processed.

Lemma 4.9 (Pattern 2 & Transitive Encoding - 1): Given an RDB binary relation r_1 that conforms to P_2 (i.e. r_1 in $P_2 = (R_2, r_2, R_1, r_1, IC_2)$ from definition 4.5) and a real world acyclic non-trivial transitive binary relation r_{rw} (definition 4.10). If $r_1 \in \text{Encoding}_{\text{Trans}}(r_{rw}, P_2)$, then:

i) r_1 is acyclic,

ii) $(\exists\, x_1, x_2, x_3 \in D_{pk2})\ ((x_1, x_2), (x_2, x_3) \in r_1 \wedge (x_1 \neq x_3))$, and

iii) r_1 is minimal w.r.t. transitivity.

Proof:

- The proof for (i) and (ii) is similar to the proof presented for Lemma 4.5.
- For (iii), r_1 is minimal w.r.t. transitivity by the definition of Transitive Encoding (definition 4.10).

Lemma 4.10 (Pattern 2 & Transitive Encoding - 2): Given an RDB binary relation r_1 that conforms to P_2 (i.e. r_1 in $P_2 = (R_2, r_2, R_1, r_1, IC_2)$) and satisfies:

i) r_1 is acyclic,

ii) $(\exists\, x_1, x_2, x_3 \in D_{pk2})\ ((x_1, x_2), (x_2, x_3) \in r_1 \wedge (x_1 \neq x_3))$, and

iii) r_1 is minimal w.r.t. transitivity.

For such r_1, we can construct an acyclic non-trivial transitive relation r_{con} such that $r_1 \in \text{Encoding}_{\text{Trans}}(r_{con}, P_2)$.

Unlike a real world binary relation r_{rw}, a binary relation r_{con} is mathematically constructed and might not make sense in the real world.

Proof:

- We construct r_{con} as the transitive closure of r_1 (i.e. $r_{con} = (r_1)^{+}_{Trans}$)
- The proof for (i) and (ii) is similar to the proof presented for Lemma 4.6.

- For (iii) and by the definition of Transitive Encoding, r_1 is minimal w.r.t. transitivity.

Lemma 4.11a (Pattern 2, Algorithm A2, isTransMin - 1): Given an RDB binary relation r_1 that conforms to P_2 (i.e. r_1 in $P_2 = (R_2, r_2, R_1, r_1, IC_2)$ from definition 4.5) and $(\exists\ x_1, \ldots, x_n \in D_{pk2} \wedge n > 2)\ ((x_1, x_2), \ldots, (x_{n-1}, x_n) \in r_1 \wedge\ (x_1, x_n) \in r_1)$ with $x_1, \ldots, x_n$ as distinct elements, and given algorithm A2 with "result_set = ∅" in Line 7, "isAcyclic = True" in Line 11 and ei = x_1 in Line 13. We make the following assertions in the inner While-Loop header (line 17):

i. isTransMin $\Leftrightarrow (\neg\ \exists\ (x_1 = ei, x_2), \ldots, (x_{n-1}, x_n), (x_1, x_n) \in$ processed_tuples $\wedge\ n > 2)$

ii. $(\forall\ b \in$ tc_set$)\ (\exists\ x_1 = ei, x_2, \ldots, x_n = b \in D_{pk2} \wedge n \geq 2)$

$((x_1, x_2), \ldots, (x_{n-1}, x_n) \in$ processed_tuples $\wedge\ x_2, \ldots, x_n \in$ tc_set$)$

Proof:

- It is given $(\exists\ x_1, \ldots, x_n \in D_{pk2} \wedge n > 2)\ ((x_1, x_2), \ldots, (x_{n-1}, x_n), (x_1, x_n) \in r_1) \wedge ei = x_1$ [1]
- IsTransMin = True (line 9 in A2)
- From [1], ei = x_1 (line 13)
- From isTransMin = True and ei ≠ null, A2 enters outer While-Loop (lines 14-27)

$\Rightarrow$ From processed_tuples = $\sigma_{A1a = x1}\ (r_1)$ (line 15)

$\Rightarrow$ From $\exists\ (x_1, x_2), (x_1, x_n) \in r_1$ in [1], processed_tuples ≠ null.

- Let us assume processed_tuples = $\{(ei, y_k)\}$, where $(ei, y_k) \in r_1 \wedge 1 \leq k \leq m$

$\Rightarrow$ ei_dir = tc_set = $\{y_1, \ldots, y_m\}$ (line 16)

- Initialization Step:
  - isTransMin = True (line 9)

- processed_tuples = $\{(ei, y_k)\}$, k = 1, …, m (line 15)

- From isAscyclic=true (given) and processed_tuples = $\{(ei, y_k)\}$, ei $\notin \{y_k\}$

$\Rightarrow \neg \exists (x_1,x_2),\ldots,(x_{n-1},x_n), (x_1,x_n) \in$ processed_tuples $\wedge$ n > 2

$\Rightarrow$ 'i' holds

- $\forall$ b $\in$ tc_set $\Rightarrow$ b $\in \{y_k\} \Rightarrow \exists (x_1$=ei, x_2= $y_k) \in$ processed_tuples

$\Rightarrow$ 'ii' holds

$\therefore$ Lemma 4.11a holds for initialization step [2]

- Induction Step:

  - Assume at step k, 'i' and 'ii' hold at line 17.

  - If isTransMin = false or ($\neg \exists$ (b, c) $\in r_1$) (b $\in$ (tc_set $\cap$ A1a_set) $\wedge$

    (b, c) $\notin$ processed_tuple), inner while loop ends

  - If isTransMin = True and ($\exists$ (b, c) $\in r_1$) (b $\in$ (tc_set $\cap$ A1a_set) $\wedge$

    (b, c) $\notin$ processed_tuple), A2 executes inner while-loop (lines 17-25) [3]

  - $\text{processed\_tuples}_{k+1} = \text{processed\_tuples}_k \cup \{(b, c)\}$ (line 18)

  - if (c $\in$ ei_direct) (lines 19-21) $\Rightarrow$ $(x_1, c) \in \text{processed\_tuples}_{k+1}$

    $\Rightarrow$ isTransMin = False (line 20)

    $\Rightarrow$ End of inner while-loop (line 25)

    $\Rightarrow \text{tc\_set}_{k+1} = \text{tc\_set}_k$

    $\Rightarrow$ From [3] and $\text{tc\_set}_{k+1} = \text{tc\_set}_k$,

    $\Rightarrow b \in \text{tc\_set}_{k+1} \wedge (\exists x_1 = ei, x_2, \ldots, x_n = b \in D_{pk2} \wedge n \geq 2)$ ($(x_2, \ldots, x_n \in \text{tc\_set}_{k+1}) \wedge$

    $(x_1$=ei, $x_2),\ldots,(x_{n-1},x_n$=b), (b, c) $\in \text{processed\_tuples}_{k+1} \wedge$

$(x_1, c) \in processed\_tuples_{k+1}) \wedge isTransMin = False$

$\Rightarrow$ 'i' is true [4]

- From $tc\_set_{k+1} = tc\_set_k$,

$\Rightarrow \forall b \in tc\_set_{k+1}, b \in tc\_set_k$

$\Rightarrow (\exists x_1=ei, x_2, \ldots, x_n = b \in D_{pk2} \wedge n \geq 2) ((x_1, x_2),\ldots,(x_{n-1}, x_n) \in processed\_tuples_{k+1} \wedge$

$x_2, \ldots, x_n \in tc\_set_{k+1})$

$\Rightarrow$ 'ii' is true [5]

- if ($c \notin ei\_direct$) (lines 21-24)

$\Rightarrow (x_1, c) \notin processed\_tuples_{k+1}$ $\Rightarrow$ From Initialization step, $(x_1, c) \notin r_1$

$\Rightarrow$ From loop header, $(b,c) \notin processed\_tuple_k \Rightarrow (b, c) \neq (x_1, c)$

- $tc\_set_{k+1} = tc\_set_k \cup \{(c)\}$ (line 22)

$\Rightarrow$ End of inner while-loop (line 25)

- From [3],

isTransMin=True and ($\exists (b,c) \in r_1$) ($b \in (tc\_set \cap A1a\_set) \wedge (b,c) \notin processed\_tuple$)

$\Rightarrow$ From step k, $(b \in tc\_set_k) \wedge (\forall b \in tc\_set_k) (\exists x_1=ei, x_2, \ldots, x_n = b \in D_{pk2} \wedge n \geq 2)$

$(\exists (x_1, x_2),\ldots, (x_{n-1}, x_n) \in processed\_tuples_k \wedge x_2, \ldots, x_n= b \in tc\_set_k)$

$\Rightarrow$ For next iteration (line 17)

- $isTransMin = True \wedge c \in tc\_set_{k+1} \wedge (b,c) \in processed\_tuples_{k+1} \wedge$

$(x_1,c) \notin processed\_tuples_{k+1}$ [6]

$\Rightarrow (\neg \exists (x_1=ei, x_2), \ldots, (x_{n-1}, x_n=b), (x_n, c), (x_1, c) \in processed\_tuples_{k+1} \wedge n > 2)$

$\Rightarrow$ 'i' is true [7]

$\Rightarrow \forall\, b \in tc\_set_{k+1,}\ (\exists\, x_1{=}ei, x_2,\ldots, x_n{=}b \in D_{pk2} \wedge n \geq 2)$

$((x_1, x_2),\ldots,(x_{n-1}, x_n{=}b) \in processed\_tuples_{k+1} \wedge\ x_2, \ldots, x_n \in tc\_set_{k+1})$

$\Rightarrow$ 'ii' is true [8]

$\therefore$ Lemma 4.11a holds (from [4], [5], [7] and [8])

- End of Proof.

Lemma 4.11b (Pattern 2, Algorithm A2, isTransMin - 2): Given an RDB binary relation r_1 that conforms to P_2 (i.e. r_1 in $P_2 = (R_2, r_2, R_1, r_1, IC_2)$ from definition 4.5) and given algorithm A2 with "result_set = ∅" in line 7 and "isAcyclic = True" in Line 11. The following are equivalent:

i) isTransMin is True in line 28 of A2 given r_1, and

ii) r_1 is minimum w.r.t. transitivity.

Proof:

To establish that 'i' and 'ii' in Lemma 4.11b are equivalent, we need to prove:

a. 'i' $\Rightarrow$ 'ii'

b. 'ii' $\Rightarrow$ 'i'

- From definition 4.5, R_1={ $(A_{1a} : D_{pk2})$, $(A_{1b} : D_{pk2})$ }
- Proof for 'i' $\Rightarrow$ 'ii':
  - Given 'i' (i.e. "isTransMin = True" in line 28 of A2),
    - Assume 'ii' is false (i.e. r_1 is not minimum w.r.t. transitivity)

$\Rightarrow (\exists\, (x_i, x_j) \in r_1)\ (\ (r_1 - \{(x_i, x_j)\})^{+}_{Trans} = (r_1)^{+}_{Trans}\ )$

$\Rightarrow (\exists\ x_1,\ldots,x_n \in D_{pk2} \wedge n > 2)\ ((x_1,x_2),\ldots,(x_{n-1},x_n),(x_1,x_n) \in r_1) \wedge x_1,\ldots, x_n$ are distinct [1]

- It is given that "result_set = ∅" in line 7

$\Rightarrow$ isTransMin = True (line 9 in A2)

- It is given that "isAcyclic = True" in line 11

- From [1], $\{ x_1,\ldots,x_{n-1}\} \subseteq \pi_{A_{1a}} (r_1)$

$\Rightarrow\ \{x_1,\ldots,x_{n-1}\} \subseteq$ A1a_set (line 12)

$\Rightarrow$ ei = getElement (A1a_set) $\Rightarrow$ ei $\in \{x_1,\ldots,x_{n-1}\} \Rightarrow$ ei $\neq$ null (line 13)

- From isTransMin = True and ei $\neq$ null,

$\Rightarrow$ A2 outer while-loop will execute at least once (lines 14-27)

$\Rightarrow$ The outer while-loop will exit when either:

- isTransMin = False $\Rightarrow$ Contradiction [2]

- we iterate over all ei $\in$ A1a_set, including ei = x_1 [3]

- For [3] and ei = x_1,

- From Lemma 4.11a, we proved that the following assertions in line 17 hold:

- isTransMin $\Leftrightarrow (\neg\exists\ (x_1$=ei,$x_2),\ldots,(x_{n-1},x_n), (x_1,x_n) \in$ processed_tuples $\wedge\ n > 2)$ [4]

- $(\forall\ b \in$ tc_set) $(\exists\ x_1$=ei, $x_2, \ldots, x_n = b \in D_{pk2} \wedge n \geq 2)$

$((x_1, x_2),\ldots,(x_{n-1}, x_n) \in$ processed_tuples $\wedge\ x_2, \ldots, x_n \in$ tc_set)) [5]

- Given [1], [4] and ei = x_1, isTransMin = False

$\Rightarrow$ Algorithm A2 exits inner While-Loop with isTransMin = False (Lines 17-25)

$\Rightarrow$ Algorithm A2 exits outer While-Loop with isTransMin = False (Lines 14-27)

$\Rightarrow$ isTransMin = False in Line 28. Contradiction. [6]

$\therefore$ 'i' $\Rightarrow$ 'ii' holds (from [2] and [6]) [7]

- Proof for 'ii' $\Rightarrow$ 'i':

  - Given 'ii' (i.e. r_1 is minimum w.r.t. transitivity),

    $\Rightarrow$ ($\neg \exists$ $(x_i, x_j) \in r_1$) ($(r_1 - \{(x_i, x_j)\})^+_{Trans} = (r_1)^+_{Trans}$)

    $\Rightarrow$ ($\forall$ n > 2) ($\neg \exists$ $(x_1, x_2), \ldots, (x_{n-1}, x_n), (x_1, x_n) \in r_1$ [8]

    - Assume 'i' is false (i.e. "isTransMin = False" in line 28 in A2)

    - In A2, isTransMin is set to False in line 20

    $\Rightarrow$ Execution reached the statement "isTransMin = False" in line (20).

    $\Rightarrow$ From [4] and given [8], isTransMin = True. Contradiction.

    $\therefore$ 'ii' $\Rightarrow$ 'i' holds [9]

$\therefore$ 'i' $\Leftrightarrow$ 'ii' (from [7] and [9])

- End of Proof.

Theorem A2-T3: Algorithm A2 with input $P_2(R_2, r_2, R_1, r_1, IC_2)$ and card(r_1) = "N:M" terminates and its output satisfies the following properties:

i) ($\exists$ r_{rw}) ((r_{rw} is N:M) $\wedge$ (r_{rw} is symmetric) $\wedge$ ($r_1 \in$ $\text{Encoding}_{\text{Symm}}$ (r_{rw} , P_2))) $\Rightarrow$

r_1 is Candidate Symmetric, and

ii) r_1 is Candidate Symmetric $\Rightarrow$

($\exists$ r_{con}) ((r_{con} is N:M) $\wedge$ (r_{con} is symmetric) $\wedge$ ($r_1 \in$ $\text{Encoding}_{\text{Symm}}$($r_{con}$, P_2))).

Note that although r_{con} may not make sense in real world, r_{con} can be mathematically constructed.

Proof A2-T3:

Termination Analysis: Algorithm A2 is composed of two sections: lines 5-6 (first section) and lines 7-32 (second section). When A2 is invoked, it executes the first section and optionally executes the second section depending on the outcome from the first. Thus, asserting that A2 terminates requires a proof that each section terminates.

The first section terminates because it is a sequence of statement without any loop, and the query operators used in line 6 are part of the DBMS with well defined and tested behavior. Therefore, section one will terminate for any given input.

The second section is processed only if the result_set computed in the first section (line 6) is empty. The second section terminates only if the external function isAcyclic() (line 10), outer loop (lines 14-27) and inner loop (lines 17-25) terminate. We assert that the external isAsyclic() function is implemented to terminate for any given input.

The outer loop is a While loop that terminates *either* when "ei = null" *or* "isTransMin = False". The former condition in the outer loop does not involve any risk because ei obtains values from a finite set (A1a_set) before the loop (line 13) and within the loop (line 26) using the GetElement() function; moreover, GetElement() returns only the A1a_set values that have not been processed. A1a_set is populated with values from column A1a in r_1 before the loop and does not expand within the loop.

The inner loop terminates either when "isTransMin = False", *or* when A2 iterates over all tuples (b, c) in r_1 such that b is a value in both tc_set and A1a sets *and* (b, c) has not been processed in the inner loop for the given ei value. The latter condition deals with 3 sets: A1a, tc_set, and processed_tuples, all of which are finite; A1a is a set of values

from A1a column in r_1, tc_set contains elements (from A1b in r_1) that are directly or indirectly connected to the given ei value (from the outer loop), and processed_tuples contains tuples (b, c) that have been processed for the given ei value. Therefore, algorithm A2 terminates regardless of the given input.

Correctness Analysis: To prove the correctness of algorithm A2 w.r.t. theorem A2-T3, we need to prove "i" and "ii" in Theorem A2-T3:

1) In algorithm A2, isCandSymm is set to True *only* in line 8:

$$\text{"isCandSymm = True"} \Leftrightarrow \text{If-Cond in line 7 is True}$$

$$\Leftrightarrow \text{"}r_1 \bowtie_{((r_1.A_{1a} = tx.A_{1b}) \text{ and } (r_1.A_{1b} = tx.A_{1a}))} \rho\ tx\ (r_1) = \varnothing\text{"}$$

2) From Definition 4.11, "r_1 is Candidate Symmetric" is equivalent to algorithm A2 returning isCandSymm=True.

3) From (1) and (2) above, we obtain

'r_1 is Candidate Symmetric' $\Leftrightarrow$ '$r_1 \bowtie_{((r_1.A_{1a}=tx.A_{1b}) \text{ and } (r_1.A_{1b}=tx.A_{1a}))} \rho\ tx\ (r_1) = \varnothing$'

4) The query '$r_1 \bowtie_{((r_1.A_{1a}=tx.A_{1b}) \text{ and } (r_1.A_{1b}=tx.A_{1a}))} \rho\ tx\ (r_1)$' in line 6 uses a theta join to find tuples that violate the symmetric minimality principle (i.e. it finds if $\exists$ (x,y), (y, x) $\in$ r_1). An empty result set returned by this query indicates that there are no such tuples (i.e. $\neg \exists$ (x,y), (y, x) $\in$ r_1). The existence of at least one tuple in the result set is deemed sufficient to declare r_1 as not minimal w.r.t. symmetry.

5) From (4), we assert:

'$r_1 \bowtie_{((r_1.A_{1a}=tx.A_{1b}) \text{ and } (r_1.A_{1b}=tx.A_{1a}))} \rho\ tx(r_1) = \varnothing$' $\Leftrightarrow$ r_1 is minimal w.r.t. symmetry

- To prove 'i',
  - A.1) Given $(\exists\ r_{rw})\ ((r_{rw} \text{ is N:M}) \wedge (r_{rw} \text{ is symmetric}) \wedge (r_1 \in \text{Encoding}_{\text{Symm}}(r_{rw}, P_2)))$,
  - A.2) Assume 'r_1 is Candidate Symmetric' is false (i.e. algorithm A2 did not return isCandSym = True per definition 4.11),
  - A.3) $\Rightarrow$ '$r_1 \bowtie_{((r_1.A_{1a}=tx.A_{1b}) \text{ and } (r_1.A_{1b}=tx.A_{1a}))} \rho\ tx(r_1) = \varnothing$' is false (line 7),
  - A.4) $\Rightarrow$ From (5), r_1 is not minimal w.r.t. symmetry.
  - A.5) $\Rightarrow$ Per definition 4.7, $r_1 \notin \text{Encoding}_{\text{Symm}}(r_{rw}, P_2)$. Contradiction.
  - A.6) $\therefore$ 'i' holds

- To prove 'ii',
  - B.1) Given "r_1 is Candidate Symmetric", assume "$(\exists\ r_{con})\ ((r_{con} \text{ is N:M}) \wedge (r_{con} \text{ is symm.}) \wedge (r_1 \in \text{Encoding}_{\text{Symm}}(r_{con}, P_2)))$" is false.
  - B.2) We construct r_{con} as the symmetric closure of r_1 (i.e. $r_{con} = (r_1)^{+}_{Symm}$)
  - B.3) $\Rightarrow$ "r_{con} is N:M" is true (from r_1 is N:M and $r_1 \subseteq r_{con}$).
  - B.4) $\Rightarrow$ "r_{con} is symmetric" is true (from $r_{con} = (r_1)^{+}_{Symm}$).
  - B.5) $\Rightarrow$ For "$r_1 \in \text{Encoding}_{\text{Symm}}(r_{con}, P_2)$" is false,
    - B.6) $\Rightarrow$ r_1 is not minimal w.r.t. symmetry and P_2,
    - B.7) $\Rightarrow$ '$r_1 \bowtie_{((r_1.A_{1a}=tx.A_{1b}) \text{ and } (r_1.A_{1b}=tx.A_{1a}))} \rho\ tx(r_1) = \varnothing$' is false.
    - B.8) $\Rightarrow$ IF-Condition block (lines 7-32) is not executed
    - B.9) $\Rightarrow$ The statement "isCandSymm = True" (line 8) is not reached
    - B.10) $\Rightarrow$ isCandSymm remains False.
    - B.11) $\Rightarrow$ "r_1 is Candidate Symmetric" is false (from (2) and B.10). Contradiction.
  - B.12) $\therefore$ 'ii' holds
- $\therefore$ Algorithm A2 is correct w.r.t. to 'i' and 'ii' (per A.6 and B.12)
- End of Proof.

Theorem A2-T4: Algorithm A2 with input $P_2(R_2, r_2, R_1, r_1, IC_2)$ and card(r_1) = "N:M" terminates and its output satisfies the following properties:

i) $(\exists\ r_{rw})\ ((r_{rw}$ is acyclic$) \wedge (r_{rw}$ is non-trivial$) \wedge (r_{rw}$ is transitive$) \wedge$
$(r_1 \in \text{Encoding}_{\text{Trans}}(r_{rw}, P_2)) \Rightarrow r_1$ is Candidate Transitive, and

ii) r_1 is Candidate Transitive $\Rightarrow$
$(\exists\ r_{con})\ ((r_{con}$ is acyclic$) \wedge (r_{con}$ is non-trivial$) \wedge (r_{con}$ is transitive$) \wedge$
$(r_1 \in \text{Encoding}_{\text{Trans}}(r_{con}, P_2))$.

Although r_{con} may not make sense in real world, r_{con} can be mathematically constructed.

Proof A2-T4:

Termination Analysis: See Termination Analysis in Proof A2-T3.

Correctness Analysis: To prove the correctness of algorithm A2 w.r.t. theorem A2-T4, we need to prove "i" and "ii" in Theorem A2-T4:

1) From Definition 4.12, "r_1 is Candidate Transitive" is equivalent to algorithm A2 returning isCandTrans = True.
2) From algorithm A2, *isCandTrans* is set to True (line 29) only when
'$r_1 \bowtie_{((r_1.A_{1a}=tx.A_{1b}) \text{ and } (r_1.A_{1b}=tx.A_{1a}))} \rho\ tx(r_1) = \varnothing$' is true (lines 06-07), isAcyclic is true (line 11), isTransMin is true (line 28) and isTrivial is False (line 28). Otherwise, *isCandTrans* is False (line 05).
3) From (1) and (2) above, we obtain
"r_1 is Candidate Transitive" $\Leftrightarrow$ "isAcyclic=True" $\wedge$ "isTransMin=True" $\wedge$ "isTrivial=False" $\wedge$ " '$r_1 \bowtie_{((r_1.A_{1a}=tx.A_{1b}) \text{ and } (r_1.A_{1b}=tx.A_{1a}))} \rho\ tx(r_1) = \varnothing$' is true"

4) We assert that the external checkAcyclic() function returns true if r_1 is acyclic (line 10). The flag/variable isAcyclic holds the output from checkAcyclic(). Therefore,

"isAcyclic = True" in A2 Line 10 given $r_1 \Leftrightarrow$ r_1 is acyclic.

5) We assert that

r_1 is Acyclic $\Rightarrow$ '$r_1 \bowtie_{((r_1.A_{1a}=tx.A_{1b}) \text{ and } (r_1.A_{1b}=tx.A_{1a}))} \rho\ tx(r_1) = \varnothing$' is true"

6) From [3], [4] and [5], we obtain:

"r_1 is Candidate Transitive" $\Leftrightarrow$ "r_1 is acyclic" $\wedge$ "isTransMin=True" $\wedge$ "isTrivial=False"

7) From Lemma 4.11b, we learned:

"isTransMin = True" in A2 Line 28 given $r_1 \Leftrightarrow$ r_1 is minimal w.r.t. transitivity.

8) Similar to Lemma 4.8, we assert

"isTrivial = False" in A2 Line 28 given $r_1 \Leftrightarrow$

$$(\exists\ x_1,x_2, x_3 \in D_{pk2})\ (\ (x_1, x_2), (x_2, x_3) \in r_1 \wedge (x_1 \neq x_3)\ ).$$

9) From [6], [7] and [8], we obtain:

"r_1 is Candidate Transitive" $\Leftrightarrow$ "r_1 is acyclic" $\wedge$ "r_1 is minimal w.r.t. transitivity" $\wedge$

$((\exists\ x_1,x_2, x_3 \in D_{pk2})\ (\ (x_1, x_2), (x_2, x_3) \in r_1\ \wedge (x_1 \neq x_3)\ )\ )$

10) From Lemma 4.9, we learned:

$r_1 \in \text{Encoding}_{\text{Trans}}\ (r_{rw}\ , P_2) \Rightarrow (r_1$ is acyclic) $\wedge$ (r_1 is minimal w.r.t. transitivity) $\wedge$

$$(\exists\ x_1,x_2,x_3 \in D_{pk2})\ (\ (x_1,x_2), (x_2,x_3) \in r_1 \wedge (x_1 \neq x_3)\ )$$

11) From [9] and [10], we obtain:

$r_1 \in \text{Encoding}_{\text{Trans}}\ (r_{rw}\ , P_2) \Rightarrow$ "r_1 is Candidate Transitive"

$\therefore$ "i" holds

12) From Lemma 4.10, we learned:

(r_1 is acyclic) $\wedge$ ($\exists$ x_1,x_2, $x_3 \in D_{pk2}$) ((x_1, x_2), (x_2, x_3) $\in r_1 \wedge (x_1 \neq x_3)$) $\wedge$

(r_1 is minimal w.r.t. transitivity) $\Rightarrow r_1 \in \text{Encoding}_{\text{Trans}}(r_{con}, P_2)$.

13) From [9] and [12], we obtain:

"r_1 is Candidate Transitive" $\Rightarrow r_1 \in \text{Encoding}_{\text{Trans}}(r_{con}, P_2)$

$\therefore$ "ii" holds

14) $\therefore$ Algorithm A2 is correct w.r.t. to 'i' and 'ii' (per [11] and [13])

- End of Proof.

4.7. Summary

Binary relations exist in various real-life scenarios. Some of these relations exhibit characteristics such as symmetry and/or transitivity. With DBMS(s) lacking the explicit support for symmetric and transitive binary relations, database designers typically rely on data modeling patterns to capture them. Using these patterns, such databases can avoid common modeling pitfalls associated with data inconsistency and storage overage.

Since ontology languages (e.g. OWL) provide the grammar to annotate binary relations as symmetric and/or transitive, and given the business value for semi-automating the generation of explicit ontology models, I investigated in this research methods to identify binary relations in relational databases that are *likely* to be symmetric or transitive. Identifying such relations required detecting certain structural patterns, and in some cases analyzing data instances. Similar to other data analysis methods in DM2ONT, the identification methods here take into account the number of data instances supporting the finding and process only those that pass a confidence threshold.

CHAPTER 5: Heuristic Methodology to Measure the Relative Explicitness of Ontology Models

5.1. Introduction

Ontology evaluation addresses the problem of assessing ontology models from the standpoint of a particular criterion, typically to determine which model better suits a specific purpose [15]. Given the rapid progress in ontologies development and their use in information systems in recent years, several approaches have been proposed to evaluate them. Generally speaking, these approaches evaluate ontologies by assessing the formal properties of the knowledge representation language, reviewing the ontology against application use cases or domain requirements, examining the ontology using design guidelines, or aligning the ontology with other models (e.g. a reference ontology model) [15][66]. In order to gain a better sense of the ontology's content, and depending on the context and availability of resources, one or more evaluation criteria can be used. Although some of the existing methods produce formal measurements, apparently none has measured the explicitness of one ontology model in comparison to another. In this context, an ontology model is considered relatively more explicit about the domain if it *contains more relevant axioms than the other model*.

When evaluating similar ontology models -- whether two ontology models produced by different agents to describe the same subject-area or different versions of the same ontology model -- there is often a need to measure the *explicitness* of one model compared to another. This measure can be used in conjunction with other evaluation methods to assess the ontology models, say to select which one is more suited for the domain. For the explicitness evaluation to be meaningful, it needs to take into account only the axioms that are relevant; i.e. those that are both valid and valuable for the domain. The validity of axioms can be established by comparing them against a reference ontology, domain requirements, or ontology design guidelines. On the other hand, the value added by an axiom can be determined by assigning weights to the different types of axioms depending on their significance in the domain.

Unlike other ontology evaluation approaches, this methodology expects the ontology entities that are being compared to represent the same real-world entity but differ only in their characteristics. Therefore, our aim is to provide a formal measure that could assist in determining the degree by which these *similar* entities differ or match one another. In our methodology, we therefore establish correspondence between entities before measuring the explicitness.

Here, we present a heuristic methodology that measures the relative explicitness of an ontology model in comparison to another. The methodology in its general form allows comparing the explicitness of two domain ontology models, or it can be specialized to measure the explicitness of a domain ontology model against a reference ontology. Although there is not formal and widely accepted definition for what

constitutes *reference ontology*, here we adapt Burgun's definition in [19] and view reference ontology as one that i) represents knowledge about a particular domain, ii) is independent from specific purpose, and iii) describes the domain in comprehensive and adequate manner.

In the next section, we will discuss related work. Next, we will formally define ontology models for our methodology, and follow it by describing the main phases involved in the methodology. Subsequently, we present a case study using a fragment of simple ontology models generated by two tools that translate Relational Database (RDB) models into ontology models [3][65].

5.2. Related Work

Over the years, various authors have investigated ontology evaluation in an attempt to provide a means to assess the quality of ontologies. Obrst et al [66] and Brank et al [15] provided surveys on existing evaluation methods. Both authors advocated advancing the field in a sound and systematic manner in order to transform ontology engineering into a scientific discipline. Obrst et al discussed five different criteria for evaluating ontologies: 1) assessing the expressivity and other formal properties of the ontology representation language, 2) evaluating the ontology against use cases and domain requirements, 3) measuring semantic agreement among domain experts, 4) comparing ontologies using semantic similarity and distance methods, and 5) performing alignment with other ontologies. Similarly, Brank et al classified existing methods into one of four categories: 1) methods that compare the ontology against "gold-standard", 2) methods that use the

ontology in an application context and evaluate the result, 3) methods that evaluate the ontology using domain data (e.g. corpus), and 4) methods that rely on human assessing the ontology using predefined criteria, requirements, etc.

Measuring similarity among ontologies is often considered part of evaluating ontologies. In Maedche and Staab [59], the authors focused on evaluating the vocabulary in the new ontology with that in a reference model, where the emphasis is on the lexical similarities between the terms found in these models. However, Maedche and Staab's method neither takes into account certain axioms (e.g. restrictions) nor measures the difference in explicitness between ontologies. Euzenat and Valtchev [35], on the other hand, measured similarity between two OWL-Lite ontologies using ontology alignment techniques (e.g. finding equivalence/subsumption relationships between models). Although they considered most of the features in OWL-Lite when conducting the comparison, their method is better suited for comparing ontologies when the match between constructs is unknown. Unlike their method, our methodology establishes correspondence between a pair of ontology entities in the matching step before measuring the difference in explicitness between the pair of matched entities.

A method which evaluates one or more ontology models by comparing them using domain data was proposed by Brewster et al [16]. In their method, they proposed a three-step process to address the evaluation: 1) identify keywords/terms in the ontologies and a corpus using clustering techniques, 2) perform term expansions using a lexical database (e.g. WordNet), 3) conduct mapping between terms. Unlike our method however, they focused only on classes and relationships names.

More comprehensive methods were proposed by Burton-Jones et al [20] and most recently by Park et al [67]. In Burton-Jones et al, the authors proposed a suite of metrics for evaluating the overall quality of ontologies based on semiotic theory. The suite consists of several criteria: Syntactic, Semantic, Pragmatic, and Social. In their method, they measured the overall quality by summing the weighted scale of each metric. Park et al [67] proposed a method based on Burton-Jones framework. Park et al, however, disregarded the Social aspect; moreover, they validated their method using four different tools, and concluded that these tools lacked the ability to automate the knowledge extraction process. Although these methods assessed the overall quality of one or more ontologies, neither measured the explicitness of one ontology entity over another.

5.3. Definition of an Ontology Model

The methodology to measure explicitness adheres to the ontology model definition presented in this section. This definition provides a layer of abstraction over ontology representation languages (e.g. OWL, RDFS, etc.), which will allow the methodology to be applied to ontology models even when they use different representation languages.

Here, an ontology model is defined as a set of ontology classes. Formally, let m be an ontology model: $m = \{c_1, c_2, \ldots, c_n\}$, where c_i $(1 \leq i \leq n)$ is an ontology class.

An ontology class is defined as a structure that has a name/label, a set of data-type properties, a set of object properties, and a set of super-class names (to identify the super-classes to which this class is a specialization of). Formally, let c_i be an ontology class in

the ontology model *m*: $c_i = (nc_i , DP(c_i), OP(c_i), SC(c_i))$, where nc_i is a name that uniquely identifies class c_i (e.g. URI) within ontology model *m*.

A set of data-type properties is defined formally as follows: Let $DP(c_i)$ be the set of data-type properties attached to class c_i: $DP(c_i) = \{dp_{i-1}, dp_{i-2}, \ldots , dp_{i-n_i} \}$, where dp_{i-j} $(1 \leq j \leq n_i)$ is a data-type property in the class c_i. Similarly, a set of object properties is defined as follows: Let $OP(c_i)$ be the set of object properties that are attached to class c_i: $OP(c_i) = \{op_{i-1}, op_{i-2}, \ldots , op_{i-m_i} \}$, where op_{i-j} $(1 \leq j \leq m_i)$ is an object property in the class c_i. Also, a set of super-classes is defined as a collection of names for the classes identified as super-classes of a particular class. Formally, let $SC(c_i)$ be the set of super-class names for class c_i: $SC(c_i) = \{nc_1,nc_2,\ldots,nc_{q_i} \}$, where nc_j $(1 \leq j \leq q_i)$ is a name of another ontology class (say c_j) that serves as a super-class for class c_i -- whose name is nc_i -- and $nc_j \neq nc_i$ (to prevent direct circular references).

A data-type property is defined as a structure that has a name/label, a data-type, and a set of data-type property restrictions. Formally, let dp_{i-j} be a data-type property in the set $DP(c_i)$: $dp_{i-j} = (ndp_{i-j} , dt_{i-j} , DPR(dp_{i-j}))$, where ndp_{i-j} is a unique name within the ontology model *m*, dt_{i-j} is the data type for dp_{i-j} (drawn from a finite set of data types; e.g. XML Schema simple types), and $DPR(dp_{i-j})$ is a set of data-type property restrictions as defined later in this section. In a similar manner, an object property is defined as a structure that has a name, a target ontology class, and a set of object property restrictions. Formally, let op_{i-j} be an object property in the set $OP(c_i)$: $op_{i-j} = (nop_{i-j} , t\text{-}nc_{i-j} , OPR(op_{i-j}))$, where nop_{i-j} is a unique name in the ontology model *m*, $t\text{-}nc_{i-j}$ is the name of the target

class where $t\text{-}nc_{i\text{-}j} = nc_k$ (nc_k is the name of class c_k) or $t\text{-}nc_{i\text{-}j} = nc_i$ (i.e. recursive relation), and $OPR\ (op_{i\text{-}j})$ is a set of object property restrictions as defined later in this section.

A data-type property restriction set is defined as a collection of restrictions that applies to a specific data-type property in a class. Formally, let $DPR(dp_{i\text{-}j})$ be the set of data-type property restrictions for data-type property $dp_{i\text{-}j}$: $DPR(dp_{i\text{-}j}) = \{dpr_{i\text{-}j\text{-}1}, dpr_{i\text{-}j\text{-}2}, \ldots, dpr_{i\text{-}j\text{-}n_{i\text{-}j}}\}$, where $dpr_{i\text{-}j\text{-}k}$ ($1 \leq k \leq n_{i\text{-}j}$) is a data-type property restriction in data-type property $dp_{i\text{-}j}$. Similarly, an object property restriction set is defined as follows: Let $OPR(op_{i\text{-}j})$ be the set of object property restrictions for object property $op_{i\text{-}j}$: $OPR(op_{i\text{-}j}) = \{opr_{i\text{-}j\text{-}1}, opr_{i\text{-}j\text{-}2}, \ldots, opr_{i\text{-}j\text{-}m_{i\text{-}j}}\}$, where $opr_{i\text{-}j\text{-}k}$ ($1 \leq k \leq m_{i\text{-}j}$) is an object property restriction in object property $op_{i\text{-}j}$.

A data-type property restriction is defined as a structure that has a restriction type and possibly restriction value(s). Formally, let $dpr_{i\text{-}j\text{-}k}$ be a data-type property restriction in the data-type property restriction set $DPR(dp_{i\text{-}j})$: $dpr_{i\text{-}j\text{-}k} = (dprt_{i\text{-}j\text{-}k}, dprv_{i\text{-}j\text{-}k})$, where $dprt_{i\text{-}j\text{-}k} \in \{$*restricted-values, lower-bound-cardinality, upper-bound-cardinality, functional, inverse-functional*$\}$, and $dprv_{i\text{-}j\text{-}k}$ is either null or value(s) suitable for the associated restriction type.

Finally, an object property restriction is defined as a structure that has an object property restriction type and possibly restriction value(s). Formally, let $opr_{i\text{-}j\text{-}k}$ be an object property restriction in the set of object property restrictions $OPR(op_{i\text{-}j})$: $opr_{i\text{-}j\text{-}k} = (oprt_{i\text{-}j\text{-}k}, oprv_{i\text{-}j\text{-}k})$, where $oprt_{i\text{-}j\text{-}k} \in \{$*transitive, symmetric, lower-bound-cardinality, upper-bound-cardinality, functional*$\}$, and $oprv_{i\text{-}j\text{-}k}$ is either null or a value suitable for the associated restriction type.

5.4. Methodology to Measure Explicitness

In this heuristic methodology, we measure the relative explicitness of two ontology models that describe the same subject-area by 1) eliminating erroneous and extraneous axioms from both models, 2) matching axioms across models, and 3) computing the difference. Figure 10 depicts the phases and artifacts in the methodology.

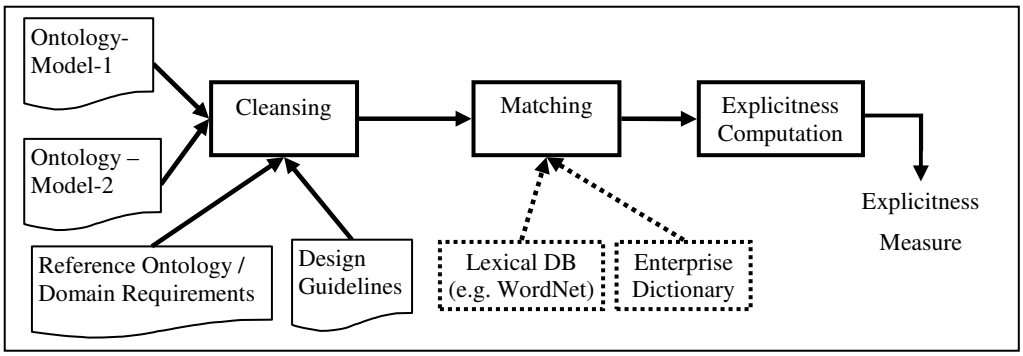


Figure 10: Methodology to measure explicitness between two Ontology models

In this methodology, we compute the difference in explicitness between two ontology entities by designating one of them as an anchor and computing a real-valued measure ranging between positive/negative one inclusive [-1 … 1]. An ontology entity can be an ontology model, class, or property. The anchor entity is considered relatively more explicit than the non-anchor entity when the computed measure is positive, equally explicit when the measure is zero, and less explicit when negative. The higher the value, the more explicit the anchor entity is. In the general form of the methodology, the two ontology entities can be from two domain ontology models generated by different agents. In the specialized form, the explicitness of an ontology entity from a domain ontology

model can be compared against an entity from a reference ontology by designating the latter as the anchor.

5.4.1. Cleansing Phase

In this phase, the input ontology models for which explicitness is to be measured will be compared against a reference ontology, domain requirements, or ontology design guidelines. The aim here is to eliminate all invalid or redundant axioms (i.e. false-positives) from the input models. Intuitively, the reference ontology and domain requirements are considered the gold-standard containing all the true-positive axioms. Axioms in the input models are discarded if they do not have a corresponding axiom in the reference ontology or domain requirements. In the absence of a reference ontology or domain requirements, design guidelines can be used to purge any axioms that are deemed invalid or redundant. Design guidelines may include eliminating data-type properties if the same facts are conveyed by object properties, or purging axioms if they are invalid (e.g. upper-bound-cardinality < 1, lower-bound-cardinality < 0) or redundant (e.g. removing upper-bound-cardinality=1 when a property is also set as functional). The result of this phase will be two semantically valid ontology models.

5.4.2. Matching Phase

Ontology matching, which is also known as ontology mapping, is part of an active research area called *Ontology Integration* [51][92]. In ontology matching, the aim is to relate ontology entities that have the same or closest intended meaning. Over the years,

researchers have proposed various approaches to (semi-) automate the process by matching entities using shared ontology (e.g. reference ontology), heuristic techniques (e.g. lexical and feature –based), or machine learning [64].

In the matching phase of this methodology, correspondence between entities is established either manually or using an existing (external) approach. The input to this phase is two semantically valid ontology models (the result of the previous phase). On the other hand, the output is sets of matched class pairs, data-type property pairs, and object property pairs. In cases where an ontology entity does not have a match, the pair will contain a null value to denote the lack of correspondence.

In this research, we adapted a lexical and feature based technique [35] to match entities according to the type, membership, and name of the entity using a top-down approach. Moreover, the matching generated is one-to-one at most, with an entity on one side having either a one or zero match to an entity on the other side. The matching process occurs as follows:

1. Entities of type class in the anchor model are matched with classes in the other model using the class name, which may be matched through an edit-distance technique [53]. If the result of this step for a particular class falls under a certain threshold, a lexical database (e.g. WordNet) and/or an enterprise dictionary (corporate/business dictionary) can be used to enhance the matching. The output from this step will be a set of matched class-pairs of the form:

   $MC\ (m_1\ ,\ m_2) = \{(c_{1\text{-}1}\ ,\ c_{2\text{-}1}\ ),(null\ ,\ c_{2\text{-}2}\ ),(c_{1\text{-}2}\ ,\ null\ ),\ \ldots\ \}$,

where m_1 and m_2 are ontology models with m_1 as an anchor, and $c_{1\text{-}i}$ and $c_{2\text{-}j}$ are classes in models m_1 and m_2 respectively. In the matched class-pair set *MC,* an element in a pair may be null if a class from one model does not have a corresponding class in the other model.

2. Next, for every pair of classes in the matched class-pair set *MC*, elements of the data-type and object property sets within one class in the pair are matched with elements of the same type in the other class. These elements are matched using the element name and their characteristics. For instance, data-type property elements are matched using their names and data-type while object property elements are matched using their names and the name of the target class. Similar to the class matching step, if the result of the name matching for any element falls under a certain threshold, a lexical database and/or enterprise dictionary can be explored. The result of this step are sets of matched data-type property pairs and object property pairs:

Matched Data-Property $MDP(c_{1\text{-}i}, c_{2\text{-}j}) = \{(dp_{1\text{-}i\text{-}1}, dp_{2\text{-}j\text{-}1}), (null, dp_{2\text{-}j\text{-}2}), (dp_{1\text{-}i\text{-}2}, null), \ldots\}$,

Matched Object-Property $MOP(c_{1\text{-}i}, c_{2\text{-}j}) = \{(op_{1\text{-}i\text{-}1}, op_{2\text{-}j\text{-}1}), (null, op_{2\text{-}j\text{-}2}), (op_{1\text{-}i\text{-}2}, null), \ldots\}$,

where $c_{1\text{-}i}$ and ,$c_{2\text{-}j}$ are pair of matched classes and $c_{1\text{-}i}$ is an anchor class.

5.4.3. Explicitness Computation Phase

Using the sets of matched pairs of classes, and data-type and object properties, the difference in explicitness between two entities is computed using the explicitness measurement method as defined in this section. In this method, two similar ontology entities are compared in order to measure which ontology entity is more explicit than the

other. This method is flexible enough to measure the explicitness at different levels of granularity by comparing two models, two classes, or two properties, and taking into account only the constructs that make up the entities being measured. Furthermore, to measure the explicitness, the method allows assigning different weight to the different types of ontology constructs.

Definition 5.1 (Ontology Explicitness Measure): Given an entity of category *E* and two ontology entities e_i and e_j, explicitness of e_i (anchor entity) with respect to e_j is computed using the following real-valued family of functions:

$$explicitness_{entity\text{-}cat}(e_i, e_j): E^2 \rightarrow [-1 \ldots 1],$$

where $entity\text{-}cat \in \{M, C, SCS, DPS, OPS, DP, OP, DT, DPRS\}$, $e_i \in \{m_i, c_i, dp_i, op_i\}$, $e_j \in \{m_j, c_j, dp_j, op_j\}$, and e_i and e_j are of the same entity category (i.e. both are models, classes, etc.).

The $explicitness_{entity\text{-}cat}$ functions are characterized as being both reflexive and symmetric:

For all ontology entities e_i and e_j:

i) $\text{explicitness}_{entity\text{-}cat}(e_i, e_i) = 0$ (reflexivity)

ii) $|\text{explicitness}_{entity\text{-}cat}(e_i, e_j)| = |\text{explicitness}_{entity\text{-}cat}(e_j, e_i)|$ (symmetry on absolute values)

5.4.3.1. Explicitness for Ontology Models: To determine the explicitness of one ontology model over the other, we use the $explicitness_M$ function as defined below:

Definition 5.2 (Explicitness Measure for Ontology Models): Given two ontology models m_1 and m_2 and the matched classes set *MC*, explicitness of m_1 (the anchor model) with respect to m_2 is computed as:

$$explicitness_M(m_1, m_2) = \frac{\sum_{k=1}^{|MC|} explicitness_C((c_{1-i}, c_{2-j})_k)}{|MC|},$$

where $(c_{1\text{-}i}, c_{2\text{-}j})_k$ is the *k-th* matched class-pair in *MC* for models m_1 and m_2, and $|MC|$ is the number of class-pairs in MC (from the matching phase).

5.4.3.2. Explicitness for Ontology Classes: The explicitness measure for a pair of matched classes is computed using the $explicitness_C$ function:

Definition 5.3 (Explicitness Measure for Ontology Classes): Given a pair of ontology classes c_1 and c_2, explicitness of c_1 (the anchor class) with respect to c_2 is computed as:

$$explicitness_C(c_1, c_2) = \omega_{SCS} \times explicitness_{SCS}(c_1, c_2) + \omega_{DPS} \times explicitness_{DPS}(c_1, c_2) + \omega_{OPS} \times explicitness_{OPS}(c_1, c_2)$$

where ω_x is the weight for ontology category *x*, the weights ω_x are normalized (i.e. $\omega_{DPS} + \omega_{OPS} + \omega_{SCS} = 1$), and $explicitness_{SCS}$, $explicitness_{DPS}$ and $explicitness_{OPS}$ are as defined in the next sections.

5.4.3.3. Explicitness for Super-Class Set: Computing the explicitness of the super-class set in classes c_1 and c_2 is achieved using the $explicitness_{SCS}(c_1, c_2)$ function:

Definition 5.4 (Explicitness Measure for Super-Class Set): Given the super-class sets $SC(c_1)$ in class c_1 and $SC(c_2)$ in class c_2, the $explicitness_{SCS}$ function is computed as:

$$explicitness_{SCS}(c_1, c_2) = \frac{|SC(c_1)| - |SC(c_2)|}{Max(Max(|SC(c_1)|, |SC(c_2)|), 1)},$$

where $|SC(c_x)|$ is the number of elements in the $SC(c_x)$ set, and Max(m , n) is a function that returns the greater of the two values: *m* and *n*.

5.4.3.4. Explicitness for Data-Type Properties: Explicitness of all data-type properties in classes c_i and c_j is measured using the $explicitness_{DPS}(c_1, c_2)$ function, which requires summation of the explicitness for each data-type property pair in the matched data-type property set *MDP*.

Definition 5.5 (Explicitness Measure for Data-Type Property Sets): Given the two matched ontology classes c_1 and c_2 and their matched data-type property set *MDP*, the $explicitness_{DPS}$ function is computed as:

$$explicitness_{DPS}(c_1, c_2) = \frac{\sum_{k=1}^{|MDP|} \text{explicitness}_{DP}((dp_{1-x}, dp_{2-y})_k)}{|MDP|}$$

where $(dp_{1\text{-}x}, dp_{2\text{-}y})_k$ is the k-th pair of matched data-type properties in *MDP* for classes c_1 and c_2, and |MDP| is the number of pairs in MDP.

Definition 5.6 (Explicitness Measure for Data-Type Properties): Given a pair of matched data-type properties dp_1 and dp_2, explicitness of dp_1 (the anchor property) with respect to dp_2 is computed as:

$$explicitness_{DP}(dp_1, dp_2) = \omega_{DT} \times \text{explicitness}_{\text{DT}}(\text{dp}_1, \text{dp}_2) + \omega_{DPRS} \times \text{explicitness}_{\text{DPRS}}(\text{dp}_1, \text{dp}_2)$$

where ω_{DT} is the weight assigned to data-type form of axioms, ω_{DPRS} is the weight assigned to data-type property restriction sets, and the weights ω_x are normalized (i.e. $\omega_{DT} + \omega_{DPRS} = 1$).

Definition 5.7 (Explicitness Measure for Data-Type Axioms): Given the two data-types dt_1 and dt_2 in data-type properties dp_1 and dp_2 respectively, explicitness of dt_1 (the anchor data-type) with respect to dt_2 is computed as:

$$explicitness_{DT}(dp_1, dp_2) = \begin{cases} 1 ; (dt_1 \neq null) \wedge (dt_2 = null) \\ 0 ; ((dt_1 \neq null) \wedge (dt_2 \neq null)) \vee ((dt_1 = null) \wedge (dt_2 = null)) \\ -1 ; (dt_1 = null) \wedge (dt_2 \neq null) \end{cases}$$

Definition 5.8 (Explicitness Measure for Data-Type Property Restriction Sets): Given two data-type property restriction sets $DPR(dp_1)$ and $DPR(dp_2)$ in data-type properties dp_1 and dp_2 respectively, explicitness of $DPR(dp_1)$ (the anchor) with respect to $DPR(dp_2)$ is computed as:

$$explicitness_{DPRS}(dp_1, dp_2) = \sum_{k \in DPRT} \omega_k \times Find(\mathrm{DPR}(\mathrm{dp}_1), k) \; - \sum_{k \in DPRT} \omega_k \times Find(\mathrm{DPR}(\mathrm{dp}_2), k),$$

where $DPRT$ is the set of data-type property restriction types, ω_k is the weight assigned to data-type property restriction of type k, and $Find(DPR(dp_x), k)$ is a binary-valued function that returns 1 if restriction k exists in $DPR(dp_x)$ and 0 otherwise. Note that the weights ω_k are normalized (i.e. $\sum_{k \in DPRT} \omega_k = 1$).

5.4.3.5. Explicitness for Object-Type Properties: The explicitness for the object properties in the pair of matched classes c_1 and c_2 is measured using the

explicitness$_{OPS}$(c_1,c_2) function. This is computed by summing the explicitness for each object property pair in the matched object property set *MOP*.

Definition 5.9 (Explicitness Measure for Object Property Sets): Given the two matched ontology classes c_1 and c_2 with their matched object property set *MOP*, the $explicitness_{OPS}$ function is computed as:

$$explicitness_{OPS}(c_1, c_2) = \frac{\sum_{k=1}^{|MOP|} \text{explicitness}_{OP}((op_{1-x}, op_{2-y})_k)}{|\text{MOP}|}$$

where $(op_{1\text{-}x}, op_{2\text{-}y})_k$ is the *k-th* pair of matched object properties in *MOP* for classes c_1 and c_2, and |MOP| is the cardinality of the *MOP* set.

Definition 5.10 (Explicitness Measure for Object Properties): Given a pair of matched object properties op_1 and op_2 from the matched object property set *MOP*, and the two object property restriction sets $OPR(op_1)$ and $OPR(op_2)$ in op_1 and op_2 respectively, explicitness of op_1 (the anchor) with respect to op_2 is computed as:

$$explicitness_{OP}(op_1, op_2) = \sum_{k \in OPRT} \omega_k \times Find(\text{OPR}(\text{op}_1), k) \; - \sum_{k \in OPRT} \omega_k \times Find(OPR(op_2), k)$$

where *OPRT* is the set of object property restriction types, ω_k is the weight assigned to object property restriction of type *k*, and $Find(OPR(op_x), k)$ is a binary-valued function returning 1 if the restriction of type *k* exists in $OPR(op_x)$ and 0 otherwise. Note that the weights ω_k are normalized as well.

5.5. Case Studies

Using this methodology, we conducted two case studies to measure the relative explicitness of ontology models. In these, we used tools that automatically translate relational databases into ontology models [3][65]. In the first case study, each tool generated a domain ontology model from a sample database instance provided by IBM DB2 [48]; one which models employee project assignment. In the second case study, we used the same tools to generate domain ontology models from a sample database instance provided by Microsoft SQL Server [61], one which models an e-commerce business. In each case study, we used domain requirements and ontology design guidelines to cleanse the generated domain ontology models. Next, we manually matched entities across the domain models. Last, we performed the explicitness computation using the result from the cleansing and matching phases.

In the first case study, each domain ontology model had approximately eight classes and sixty properties (both data-type and object). With the domain ontology model generated by [3] set as an anchor, we computed the explicitness at the model level. The result from the $explicitness_M$ function was 0.24, which indicates that the anchor model is more explicit about the domain (i.e. contains more relevant axioms) than the model generated by [65]. The second case study dealt with approximately fourteen classes and ninety properties. Similar to the first case-study, we set the model generated by [3] as an anchor, and computed the explicitness for the model. The $explicitness_M$ result was 0.26. Due to space limitations here, we will present in this chapter a fragment of the first case study; complete results for both case studies are included in Appendix F.

Table 9 and Table 10 show, in abstract form, a part of the domain ontology models generated by these tools. A value of "Y" under a restriction-type (column) indicates that the ontology model has a restriction set for a particular property (row). Moreover, a struck-through value (e.g. ~~XYZ~~) indicates that the value is determined to be invalid or redundant based on the cleansing phase, and as such, the value will not be considered in the computation phase. To conserve space, we combined the data-type and object properties restriction types since some are common in both types of properties (e.g. lower-card, upper-card, functional).

Table 9: Abstract representation of domain ontology generated by DM2ONT (om_1) [3].

	Property			Restrictions						
Class	Type	Name	Range	Restricted-Values	Lower Card	Upper Card	functional	Inverse-Functional	Transitive	Symmetric
Employee (Emp)	Data	Emp-ID	String		Y		Y	Y		
	Data	Emp-Name	String		Y		Y			
	Data	Gender	String	Y	Y		Y			
	Object	WorksIn	Dept		Y		Y			
Department (Dept)	Data	Dept-ID	String		Y		Y	Y		
	Data	Dept-Name	String		Y		Y	Y		
	Object	Admin	Dept		Y		Y		Y	
	Object	Admin-Inv	Dept		Y					
	Object	WorksIn-Inv	Emp		Y					

Table 10: Abstract representation of domain ontology from DataMaster (om_2) [65].

	Property			Restrictions						
Class	Type	Name	Range	Restricted-Values	Lower Card	Upper Card	functional	Inverse-Functional	Transitive	Symmetric
Employee (Emp-2)	Data	Emp-ID-2	String				Y			
	Data	Emp-Name-2	String				Y			
	Data	Gender-2	String				Y			
	~~Data~~	~~WorksIn-D-2~~	~~String~~				~~Y~~			
	Object	WorksIn-O-2	Dept-2				Y			
Department (Dept-2)	Data	Dept-ID-2	String				Y			
	Data	Dept-Name-2	String				Y			
	~~Data~~	~~Admin-D-2~~	~~String~~				~~Y~~			
	Object	Admin-O-2	Dept-2				Y			

In the matching phase, we set the ontology model in Table 9 as an anchor before performing the matching. The result of this phase was the following sets of matched class-pairs (*MC*), data-type property pairs (*MDP*) and object property pairs (*MOP*):

- *MC(om_1,om_2) ={(Emp, Emp-2),(Dept, Dept-2)},*
- *MDP(Emp,Emp-2) ={(Emp-ID, Emp-ID-2),(Emp-Name, Emp-Name-2),(Gender, Gender-2)},*
- *MOP(Emp, Emp-2) ={(WorksIn, WorksIn-O-2},*
- *MDP(Dept, Dept-2) ={(Dept-ID, Dept-ID-2), (Dept-Name, Dept-Name-2)},*
- *MOP(Dept, Dept-2) ={ (Admin, Admin-O-2), (Admin-Inv, null), (WorksIn-Inv, null)}.*

In the explicitness computation phase, we used the weights shown in Table 11 to calculate the explicitness of model om_1 (shown in Table 9) in relation to om_2 (Table 10).

Table 11: Weights assigned to the different types of ontology construct.

	Class-Level-Weights			Data-Type-Property-Level-Weights							Object-Property-Level-Weights				
Var	ω_{DPS}	ω_{OPS}	ω_{SCS}	ω_{DT}	ω_{DPRS}	ω_{rstVal}	ω_{lowe}	ω_{upper}	ω_{func}	ω_{invF}	ω_{tran}	ω_{sym}	ω_{lowe}	ω_{uppe}	ω_{func}
Value	0.45	0.45	0.1	0.05	0.95	0.25	0.25	0.0	0.2	0.3	0.3	0.25	0.25	0.0	0.2

Using the formula presented in section 5.4.3 and the weights in Table 11, the overall explicitness of om_1 compared to om_2 was 0.35, which means that om_1 is more explicit than om_2. This result is inline with our expectation and explanation of what constitute explicitness at the model level since om_1 (shown in Table 9) clearly contains *more* relevant axioms than om_2 (shown in Table 10). Furthermore, we swapped the entities in the explicitness functions to have those from om_2 as an anchor, and re-computed the explicitness functions. The result of the $explicitness_M$ with om_2 as an anchor was -0.35 (negative), which confirms the symmetry property stated in section 5.4.3.

Table 12 summarizes the result from each formula for the employee/employee-2 classes, and the overall explicitness for the two models.

Table 12: Summary of explicitness calculations for ontology fragment in case-study 1

Functions	Expression	Result
$explicitness_{DPRS}$ (Emp-ID,Emp-ID-2)	0.75 – 0.2	0.55
$explicitness_{DP}$ (Emp-ID,Emp-ID-2)	(0.05 * 0) + (0.95 * 0.55)	0.523
$explicitness_{DPRS}$ (Emp-Name,Emp-Name-2)	0.45 – 0.2	0.25
$explicitness_{DP}$ (Emp-Name,Emp-Name-2)	(0.05 * 0) + (0.95 * 0.25)	0.238
$explicitness_{DPRS}$ (Gender,Gender-2)	0.7 – 0.2	0.5
$explicitness_{DP}$ (Gender,Gender-2)	(0.05 * 1) + (0.95 * 0.5)	0.475
$explicitness_{OP}$ (WorksIn,WorksIn-O-2)	0.45 – 0.2	0.25
$explicitness_{DPS}$(Emp, Emp-2)	(0.5225 + 0.2375 + 0.475) / 3	0.412
$explicitness_{OPS}$(Emp, Emp-2)	(0.25) / 1	0.25
$explicitness_{SCS}$(Emp, Emp-2)	(0 – 0) / 1	0
$explicitness_{C}$ (Emp, Emp-2)	(0.1 * 0) + (0.45 * 0.4117) + (0.45 * 0.25)	0.298
$explicitness_{C}$ (Dept, Dept-2)	(0.1 * 0) + (0.45 * 0.5225) + (0.45 * 0.35)	0.393
$explicitness_{M}$ (om_1, om_2)	(0.32025 + 0.41513) / 2	0.345

5.6. Summary

Given the increased interest in the use of ontologies, many researches have investigated means to assess the content of ontologies. Although there are several alternatives available, to the best of our knowledge, none of them attempts to measure how explicit one ontology entity is compared to another. We believe that explicitness should be *one of* the criteria on which ontology can be judged.

Here, we presented a heuristic methodology to measure the relative explicitness between two *similar* ontology entities (i.e. describing the same real-world entity). Using this methodology, a formal measure was computed to identify which entity was more/less explicit about the domain. This methodology takes into account only axioms that are relevant to the domain and by weighing them according to their type and significance in the domain. Two case studies were conducted using ontologies generated by tools that translate RDB into ontology models.

CHAPTER 6: Validation and Results of Testing

6.1. Introduction

Validation of every aspect of the DM2ONT framework requires resources that are beyond the scope of my thesis. For instance, validating the ability of DM2ONT to perform all the RDB/ORDB translation rules outlined in Chapter 3 requires access to large production RDB/ORDB database instances that contain many tables, all types of constructs and relationships supported by DM2ONT, and thousands of rows. Moreover, validating the correctness of the translation requires the involvement of several ontology modelers or access to reference ontology models, tools to translate between ontology and RDB models, and a tool to compare ontology models to the reference ontology. To bring the scope to a realistic size, I focused on validating several aspects of the framework.

6.2. Background

In order to compare the validity and explicitness of an ontology model generated by DM2ONT with that of similar tools, I explored setting up an experiment where the ontology models generated by DM2ONT and similar tools are compared with the reference ontology model used to generate the source database. Formally, let *rom* be a reference ontology model, *owl2rdb* be an existing tool that translates ontology models

into RDB models, *rdb* be a relational database generated from *rom* using the *owl2rdb* tool, om_1 be an ontology model generated from *rdb* using the DM2ONT framework, om_i (where $i \in \{2, 3\}$) be ontology models generated from *rdb* using $rdb2owl_i$ (where $i \in \{2, 3\}$) methods, and *oc* be an ontology comparison tool. The experiment was then to use *oc* to compare both om_1 and om_i to *rom* and prove that om_1 is more explicit than om_i. Figure11 depicts the evaluation method originally envisioned. By conducting this type of experiment, we can be assured that DM2ONT managed to a) retrieve the correct ontology axioms, and b) retrieve more axioms that are correct than similar tools.

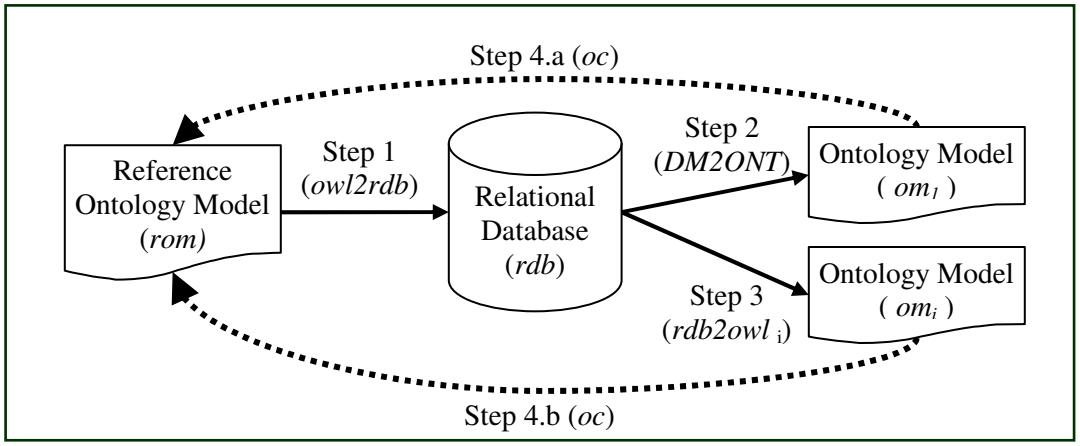


Figure 11: Earlier validation experiment.

Upon careful review of the relevant methods and tools available in the literature, I encountered the following (sorted by steps shown in Figure 11):

– **Translating Reference Ontology into Relational Database (Step 1):** There are several methods to translate from an ontology model to an RDB model [80,83,85]. Among these methods, only one performs the translation from an OWL-based

ontology model to an RDB model [87]. Unfortunately, this method has not been implemented. Because conducting manual translation using the methods' description in the literature is likely to be considered subjective (e.g. understanding of readers, details withheld by authors), this path has not been pursued.

- **Relational database to OWL-based Ontology models (Step 3):** Although the literature review (Chapter 2) found eighteen different methods for translating RDB models into ontology models, only ten were found to have been implemented. Among these ten methods, only six generate OWL-based ontology models that describe the subject-area [24,28,46,47,54,65], and only two of these six methods are publicly available [24,65]. In order to obtain access to the implementation of more methods, I made multiple attempts to contact the authors of the remaining four OWL-based methods, both directly and through Prof. Sibley's personal contacts in the regions where the authors are located. Unfortunately, I was unable to obtain access to the implementation of any of the four OWL-based methods. For the same subjectivity reason, performing the translation manually was not considered.
- **OWL-Based Ontology Comparison Tools (Step 4):** Due to the challenges encountered in Step 1 (i.e. the lack of an automated OWL-based method for translating Ontology to RDB), this path was not explored.

6.3. Implementation of the DM2ONT Prototype

In this research, I implemented a software prototype for the DM2ONT framework. The implementation of DM2ONT is composed of several components (as outlined in Chapter

3). Moreover, the framework was implemented with extensibility, interoperability, and maintainability in mind.

To provide extensibility, DM2ONT uses an abstract ontology representation internally; i.e. intermediate object representation. Using such representation allows the translation from different source models and to different target ontology representations; this is illustrated in Figure 12. This made it possible to isolate the source and target representations in their own components. Extending the prototype to support translation from a new type of source data model, such as XML, would require developing only one source component. This new source component would have to obtain information about the source model and populate the intermediate object representation. To support different ontology representations, such as RDFS, a target component would be needed to translate from the intermediate object representation to the desired target representation.

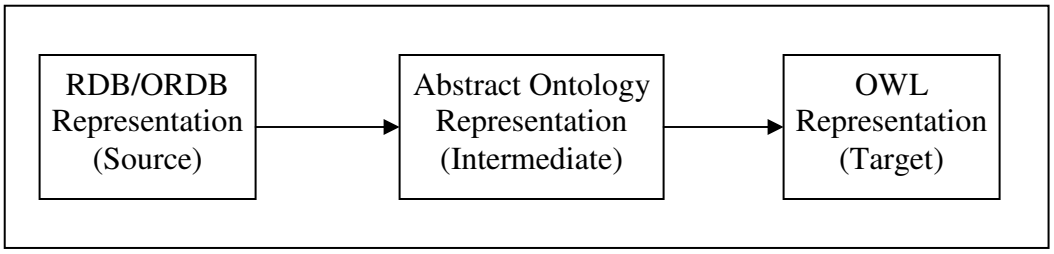


Figure 12: DM2ONT high-level architecture (extensibility view).

For portability and interoperability purposes, the prototype was implemented using Java and JDBC. Portability was achieved by using Java, which eliminates the need to rebuild (i.e. re-compile and re-link) the code when running the prototype in different platforms. To allow interoperating with different types of DBMS(s), DM2ONT used the

JDBC interface instead of the native API clients provided by DBMS vendors. While native API clients typically provide better performance and support for features, the testing performed with the JDBC interface yielded an acceptable level of performance, and provided coverage for all the RDB/ORDB features of interest in DM2ONT.

In order to provide maintainability, the prototype included comments explaining the rationale behind certain implementation decisions in the code. Furthermore, to allow changing certain parameters without changing the code, DM2ONT used a property file to control certain aspects of framework including connectivity to the source DBMS, type of analysis to perform, threshold values, and name and location of the target ontology file.

Lastly, there are various software metrics in use nowadays to measure the size and complexity of a software development effort, such as the Source Lines Of Code (SLOC), and number of classes, etc. In this respect, DM2ONT prototype had approximately 6000 SLOC and 25 Java classes.

6.4. Validation of DM2ONT using Various Databases Instances

In addition to the complex RDB and ORDB instances that I developed and used for unit and functional testing, and to avoid validating DM2ONT using biased database instances that I personally develop, I used two different public-domain RDB instances for a larger validation effort. The choice of using RDB-based instances – rather than both RDB and ORDB -- in the validation phase was made to allow comparison between DM2ONT and other tools, which are capable of translating only RDB-based instances. Furthermore, in

order to conduct the validation in a realistic manner, I needed to use substantial samples of RDB instances, both in terms of schema definition and data volume.

After reviewing popular database text books [30][72], I found that they contain only “toy examples”, which are typically composed of only two to three tables and relationships with relatively few rows per table. Next, I reviewed the sample database instances that are packaged with major commercial DBMS(s) such as IBM DB2, Microsoft SQL Server, and Oracle Database. The samples included by these vendors contained RDB instances with an acceptable level of sophistication. Table 13 shows various characteristics of these sample databases.

Table 13 - Characteristics of the sample RDB instances included with major DBMS(s)

Metrics	IBM DB2	Oracle Database	Microsoft SQL Server
Number of Tables	8	7	14
Number of Relationships	12	8	15
Number of Columns	48	35	72
Primary & Foreign Keys	Yes	Yes	Yes
Unique Constraint	No	No	No
Not Null Constraint	Yes	Yes	Yes
n:m Relationships	No	No	Yes
ISA (Type 1 & Type 2)	No	No	No
Transitive Relations	Yes	Yes	Yes
Symmetric Relations	No	No	No
Enumrated Values	Yes	Yes	Yes

First, I downloaded and configured the sample RDB database provided by IBM DB2 [48], and executed both DM2ONT and one of publicly-available methods that I

have been able to obtain -- DataMaster [65] -- against this sample database. The result of executing DM2ONT and DataMaster were ontology models. These models were validated both syntactically and semantically (see sections 6.5 and 6.7).

Second, I reviewed the sample RDB database packaged with Oracle and found that it was structurally similar to the IBM DB2 sample database as shown in Table 13. In fact, the Oracle sample database appeared to be less sophisticated than the IBM DB2 sample database and therefore, I did not explore the Oracle sample database further.

Third, I obtained and reviewed one of the sample RDB databases that was packaged with MS SQL Server [61]. As shown in Table 13, the MS SQL Server sample database was more sophisticated than the IBM DB2 sample database. Accordingly, I downloaded the MS SQL Server script for creating the sample database and setup the database instance in an IBM DB2 environment; the choice of using only the IBM DB2 environment was to avoid having to obtain, install and configure MS SQL Server software. With the database instance created and configured, I executed both DM2ONT and DataMaster against this database. The ontology models generated by these two methods were also validated syntactically and semantically.

Lastly, the two sample database instances obtained from IBM DB2 and MS SQL Server constituted the base for the two case studies used in the validation I conducted. Specifically, the first case-study used the sample RDB instance from IBM DB2, while the second case-study used that from MS SQL Server. Appendix C contains the scripts used to create the two database instances.

6.5. Syntactic Examination of the Generated Ontology Model

To validate the syntax of the OWL ontology models generated by DM2ONT, I used two publicly-available syntactic validation tools [90][96]; these tools are also known as ontology validators. Using these tools, I confirmed that the two ontology models produced by DM2ONT in both case studies were valid from a syntactic standpoint.

6.6. Functional Verification of DM2ONT

DM2ONT translates RDB and ORDB models into OWL ontologies based on translation methods/rules discussed in Chapter 3. These rules stated the type of RDB/ORDB constructs and patterns for which DM2ONT looks, the type of analysis and inference DM2ONT performs, and the type of ontology constructs that DM2ONT generates. Using this information, and the generated OWL ontology models, I manually examined and traced back all of the generated OWL constructs to the source RDB/ORDB instances, and thus, functionally verified that DM2ONT generated models according to the rules presented in Chapter 3.

6.7. Semantic Validation using Domain Requirements and Comparison with an Existing Approach

In order to validate the ontology models generated by DM2ONT from a semantic standpoint, and to verify the effectiveness and efficiency of DM2ONT in comparison to similar tools, I conducted an experiment that validated the models generated by DM2ONT and DataMaster [65] against domain requirements. The aim of this experiment

was to validate whether DM2ONT generated ontology models that a) satisfied most of the domain requirements (e.g. high recall), b) contained minimum invalid axioms (e.g. high precision), and c) were better than models generated by similar tools in terms of recall and precision.

In this experiment, I used the recall and precision measurement from the *Information Retrieval* (IR) field [12]. These two measurements have been widely used as means to measure the effectiveness and efficiency of IR methods. By using domain requirements, I was able to measure both the comprehensiveness and accuracy (i.e. recall and precision respectively) of the domain ontology models generated by DM2ONT and DataMaster. Intuitively, the set of domain requirements for each case study was assumed to contain all the axioms that were *relevant* to the domain. These axioms included both the True-Positive (TP) axioms that a domain ontology model contains and False-Negative (FN) axioms that a domain ontology neglects. Using the domain requirements, axioms in domain ontology models that were generated by DM2ONT and DataMaster were classified as either True-Positive (TP) or False-Positive (FP). Using this classification, the recall and precision were calculated using the formulas:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad (1)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad (2)$$

where TP represents the total number of valid axioms found in a domain ontology model, FN is the total number of valid axioms expected, but not found, in the domain

ontology model, and FP is the total number of invalid axioms found in the domain ontology model.

Figure 13 illustrates the experiment and steps involved. This experiment was performed on the two case-studies: first case-study with the IBM DB2 sample RDB instance as a source and the second case-study with the MS SQL Server sample RDB instance as a source.

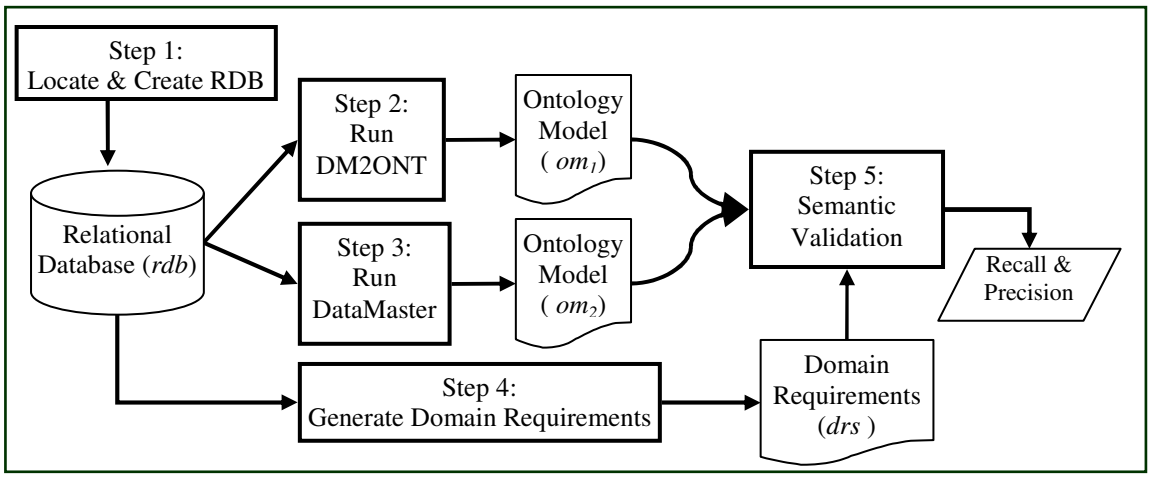


Figure 13: Semantic validation experiment.

- **Step 1: Locate and create RDB:** As discussed in section 6.4, I used the sample RDB instances that were packaged with IBM DB2 and MS SQL Server. Appendix C contains the scripts used to create these RDB instances in an IBM DB2 environment.
- **Step 2: Run DM2ONT:** In this step, I executed DM2ONT against the relational database *rdb* and as a result, DM2ONT generated the ontology model om_1. Appendix D shows the DM2ONT property file used to controls the behavior of DM2ONT.

- **Step 3: Run DataMaster:** Similar to Step 2, I executed DataMaster against the relational database *rdb*, which generated the ontology model om_2. Appendix D includes a screen-shot showing the settings used when running DataMaster.
- **Step 4: Generate Domain Requirements:** In this step, I reviewed the relational database instance *rdb* and manually extracted the domain requirements addressed by this database. Granted this step might be considered subjective, I believe the error rate in retrieving these requirements is minimal given the size of the databases used in this experiment. In the first case-study with the IBM DB2 sample RDB instance as the source, approximately 180 domain requirement statements were retrieved. On the other hand, the second case-study with the MS SQL Server sample RDB instance as the source yielded approximately 280 domain requirement statements. Appendix E includes the set of domain requirements for each case study.
- **Step 5: Semantic Validation:** In this step, I compared the domain ontology models generated by DM2ONT and DataMaster with the domain requirements of the databases that were translated by these tools. The aim in this comparison was to classify the ontology axioms found in the domain ontology models into one of two types: axioms that have corresponding domain requirements (i.e. True-Positive (TP)) and axioms that do not (i.e. False-Positive (FP)). Using this classification, I was able to compute the recall and precision – as defined in formulas 1 and 2 -- for the ontology models generated by DM2ONT and DataMaster. Table 14 provides a summary of the result from this step for both case studies. Appendix E shows all of the ontology axioms along with their classification.

The results shown in Table 14 suggest that the ontology models generated by DM2ONT were superior compared to that of DataMaster. Furthermore, the recall and precision for the ontology models generated by DM2ONT were significantly higher than that of the ontology models generated by DataMaster. The result of this experiment also demonstrated not only the ability of DM2ONT to retrieve almost all of the correct axioms, but also its tendency to return fewer incorrect axioms.

Table 14 – Various metrics for the semantic validation experiment.

	Case Study 1			Case Study 2		
	Domain Req.	DM2ONT	DataMaster	Domain Reqs.	DM2ONT	DataMaster
Total number of valid axioms (TP)	180	173	97	277	263	151
Total number of missed axioms (FN)	n/a	7	83	n/a	14	126
Total number of invalid axioms (FP)	n/a	2	32	n/a	1	37
Recall	n/a	0.961	0.539	n/a	0.949	0.545
Precision	n/a	0.989	0.752	n/a	0.996	0.803

6.8. Explicitness Measurement against Existing Approaches

To determine the relative explicitness of the ontology models generated by DM2ONT in comparison to that of DataMaster, I used the explicitness measurement methodology that I developed in this research (Chapter 5). These ontology models are considered similar because they were created from the same source database instance, and are likely to

contain similar classes and properties. Figure 14 depicts this experiment and shows artifacts that were generated, and the tools that were used.

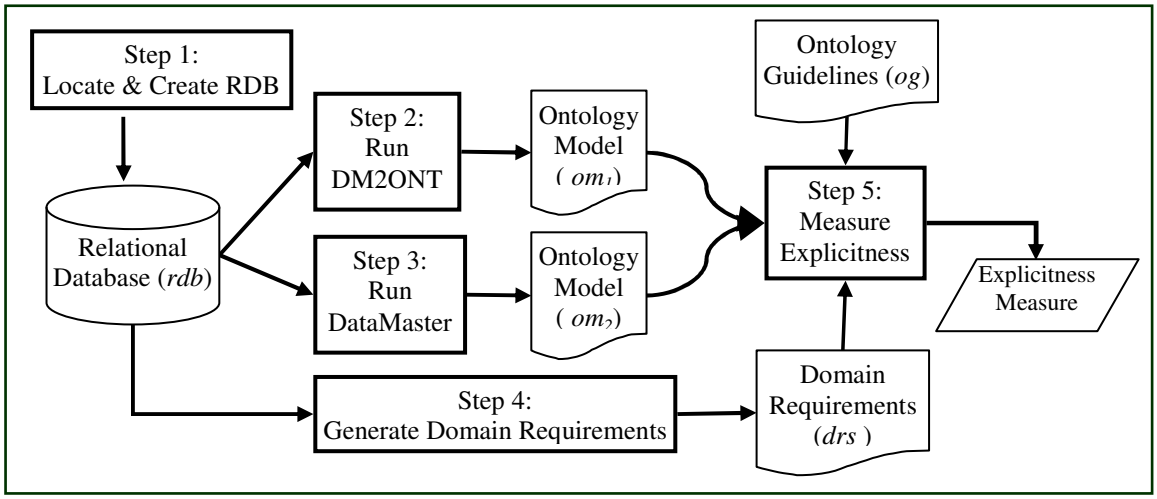


Figure 14: Measuring Explicitness between ontology models

Similar to the Semantic Validation experiment presented in Section 6.7, the explicitness measurement experiment was performed once for each of the two case studies (i.e. Case-Study 1 with IBM DB2 sample RDB instance as a source and Case-Study 2 with MS SQL Server sample RDB instance as a source). Moreover, this experiment could reuse most of the artifacts generated by the semantic validation experiment since the latter has already been conducted. Specifically, when the semantic validation experiment presented in Section 6.7 has already been conducted, only Step 5 in the explicitness measurement experiment needs to be performed. Therefore, in this dissertation, I used the ontology models and domain requirements generated in the

previous experiment and advanced directly to step-5 in the explicitness measurement experiment. Below is a description of the steps involved in this experiment:

- **Step 1 to Step 4:** See section 6.7.
- **Step 5: Explicitness Measurement Methodology:** The final step in this experiment was performed using the methodology presented in Chapter 5. In each case-study, the inputs to this methodology were a pair of ontology models generated by DM2ONT and DataMaster and the domain requirements served by the source database instance. The output was a real-value in the range of [-1 … 1], indicating which model is relatively more explicit about the domain. As discussed in Chapter 5, the methodology is composed of three phases:

  Step 5.1 - Cleansing Phase: Using the domain requirements, the two ontology models generated by DM2ONT and DataMaster were cleared of all invalid axioms (i.e. False-Positives). The results of this phase are domain ontology models that contain valid axioms only (i.e. True-Positive axioms).

  Step 5.2 - Matching Phase: Using the results from the step 5.1, I established correspondence between entities (i.e. classes, data-type and object properties) of the same type across the two cleansed ontology models. In this experiment, the cleansed ontology model from DM2ONT was designated as an anchor. Appendix F contains the sets of matched class-pairs, data-type property pairs, and object-pairs for each case-study.

  Step 5.3 - Explicitness Computation Phase: Using an MS Excel spreadsheet specifically designed to compute the explicitness (see Chapter 5), I populated it

with data from the previous two phases. The result was an explicitness measure indicating whether the anchor entity was more/less explicit than the other.

With the ontology model generated by DM2ONT in Case-Study 1 designated as the anchor model, the computed explicitness measure was 0.24. This indicated that DM2ONT generated a model that was more explicit than that generated by DataMaster. In Case-Study 2, and with the ontology model generated by DM2ONT also designated as the anchor model, the explicitness measure was 0.26. This result also denoted that DM2ONT generated a model that was more explicit than that generated by DataMaster. Table 15 and Table 16 illustrate the explicit measurement in case studies 1 and 2 respectively, at the model and class levels. In these tables, ontology entity e_1 (the anchor) corresponds to an entity generated by DM2ONT and e_2 corresponding to an entity generated by DataMaster.

Table 15 - Explicitness measurement in case-study 1 (IBM DB2 sample RDB)

$explicitness_{entity\text{-}cat}(e_1, e_2)$	**Explicitness Measure**
$explicitness_M(DM2ONT\text{-}om_1, DataMaster\text{-}om_2)$	0.24
$explicitness_C(activity_1, activity_2)$	0.35
$explicitness_C(department_1, department_2)$	0.30
$explicitness_C(employee_1, employee_2)$	0.21
$explicitness_C(empprojact_1, empprojact_2)$	0.27
$explicitness_C(emp\_photo_1, emp\_photo_2)$	0.17

explicitness$_{\text{entity-cat}}$ (e_1 , e_2)	**Explicitness Measure**
$explicitness_C\ (emp\_resume_1\ ,\ emp\_resume_2\ )$	0.17
$explicitness_C\ (project_1\ ,\ project_2\ )$	0.30
$explicitness_C\ (project\_activity_1\ ,\ project\_activity_2)$	0.15

Table 16 - Explicitness measurement in case-study 2 (MS SQL Server Sample RDB)

explicitness$_{\text{entity-cat}}$ (e_1 , e_2)	**Explicitness Measure**
$explicitness_M\ (DM2ONT\text{-}om_1\ ,\ DataMaster\text{-}om_2\ )$	0.26
$explicitness_C\ (Customers_1\ ,\ Customers_2\ )$	0.22
$explicitness_C\ (Customer\_Payment\_Methods_1\ ,\ Customer\_Payment\_Methods_2\ )$	0.23
$explicitness_C\ (Invoices_1\ ,Invoices_2\ )$	0.26
$explicitness_C\ (Orders_1\ ,Orders_2\ )$	0.26
$explicitness_C\ (Order\_Items_1\ ,Order\_Items_2\ )$	0.25
$explicitness_C\ (Payments_1\ ,Payments_2\ )$	0.26
$explicitness_C\ (Products_1\ ,\ Products_2\ )$	0.24
$explicitness_C\ (Ref\_Invoice\_Status\_Codes_1\ ,\ Ref\_Invoice\_Status\_Codes_2\ )$	0.35
$explicitness_C(Ref\_Order\_Item\_Status\_Codes_1,Ref\_Order\_\_Item\_Status\_Codes_2)$	0.23
$explicitness_C\ (Ref\_Order\_Status\_Codes_1\ ,\ Ref\_Order\_Status\_Codes_2\ )$	0.23
$explicitness_C\ (Ref\_Payment\_Methods_1\ ,\ Ref\_Payment\_Methods_2\ )$	0.23
$explicitness_C\ (Ref\_Product\_Types_1\ ,\ Ref\_Product\_Types_2\ )$	0.39
$explicitness_C\ (Shipments_1\ ,\ Shipments_2\ )$	0.26

The explicitness measurements shown in Table 15 and Table 16 for the classes and the overall models generated by DM2ONT demonstrate the ability of DM2ONT to generate classes and models that are more explicit about the domain than those generated by a similar tool. In cases where the explicitness measurements are low (e.g. less than 0.20), it can be noted that these were tables with less than 15 rows. Modifying the source database by adding more rows can significantly increase the explicitness measurement for classes, and thus increase the overall explicitness for the model.

6.9. Summary

The main objective of this dissertation was to develop an extensible framework for translating different types of data models into ontology models that are explicit about the domain they describe. In order to validate this framework, and based on the availability of resources, I employed different types of techniques to validate different aspects of the framework and its output.

First, DM2ONT was validated by implementing it as prototype using Java and JDBC. The prototype covered all the methods discussed in Chapter 3. Moreover, the implementation of DM2ONT provided features such as extensibility, interoperability and maintainability.

Second, the ontology models generated by DM2ONT were validated syntactically using tools that were publicly available. Using these validation tools, I confirmed that DM2ONT generated ontologies that conformed to the OWL specifications.

Third, I used various database models to validate the framework from a functional standpoint. To validate all the translation rules in the framework, I manually created various RDB and ORDB database instances – both schema and data – that covered all of the RDB/ORDB constructs and patterns handled by the framework. The framework was manually validated to confirm that all ontology constructs were generated according to the translation rules discussed in Chapter 3.

Fourth, using two sample RDB instances that were packaged with commercial DBMS(s) and sets of domain requirements covered by these RDB instances, I computed the recall and precision on the ontology models generated by DM2ONT and another similar tool. The results of the recall and precision from this experiment demonstrated the comprehensiveness and accuracy of the ontology models generated DM2ONT. Such results should raise the confidence of ontology modelers in the ontologies generated by DM2ONT. Although the recall and precision for the ontologies generated by DM2ONT were significantly high, I still maintain that these should be reviewed manually by the modelers and validated by domain experts.

Fifth, using the explicitness measurement methodology developed in this dissertation, I conducted an experiment to compute the explicitness of the ontology models generated by DM2ONT against that of a similar tool (i.e. DataMaster). The result from this experiment showed that DM2ONT was able to generate ontology models that are more explicit about the domain than those generated by DataMaster.

Lastly, it is worth noting that the results from the semantic validation and explicitness measurement experiments were highly dependent on the data models used as

a source. For instance, the source data models were constrained to have only RDB constructs and to limit the changes to the RDB instance used. The use of ORDB constructs will tip the scale to the side of DM2ONT because none of the tools similar to DM2ONT support ORDB constructs. Moreover, adjusting the sample RDB instances to include IS-A and symmetric binary relations, more transitive binary relations, or simply adding more rows to small tables will drastically give an advantage to DM2ONT. In these two experiments, I limited the changes to the sample RDB instances to avoid using biased data sets.

CHAPTER 7: Conclusion and Future Research

This chapter provides a summary of the research I conducted in this dissertation and proposes directions for future research.

7.1. Conclusion

In this dissertation, I developed an extensible framework for generating ontologies from various types of data models and concurrently devised a heuristic methodology for measuring the relative explicitness (i.e., relative amount of terminological content) of one ontology model in comparison to another. Both the framework and the methodology developed are practical as they facilitate the development and evaluation of ontologies, which are essential components in any semantic-based solution.

In order to ascertain the viability and effectiveness of the DM2ONT framework, I implemented it as a prototype and conducted various experiments. The prototype I implemented took into account characteristics such as extensibility, interoperability and maintainability (see Section 6.3). The experiments I conducted demonstrated not only the ability of DM2ONT to generate ontologies that are syntactically and semantically correct, but also its ability to generate ontologies that are more explicit than those generated by any similar tools that have been published. To assess the ontologies generated by DM2ONT, I conducted two experiments. Each experiment was executed using two

different case-studies, which were based on public-domain data. The first experiment used domain requirements and computed the *recall* and *precision* from the Information Retrieval field. The second experiment employed the methodology I devised for measuring the explicitness of ontologies. The results of these experiments demonstrated the superiority of DM2ONT in comparison to another method (which was one of the only two ontology translation approach that I was able to access from prior researchers).

Lastly, this research resulted in four peer-reviewed papers that my dissertation director and I co-authored and published in various scholarly avenues (see Appendix A). Conducting this research and contributing to the literature required knowledge in several disciplines including databases, artificial intelligence, and software engineering. Moreover, this research addressed several gaps in the literature related to building ontologies from data models and evaluating similar ontologies.

7.2. Future Research

The aim in my research was to facilitate the development and evaluation of ontologies. Advancing this objective requires further research in two primary areas: the DM2ONT framework and the explicitness methodology.

The DM2ONT framework was designed with extensibility in mind: thus it would be relatively simple to extend it to allow the generation of ontologies from other prevalent data models such as XML. With many organizations adopting XML as a means for exchanging information within and across organizational boundaries, these organizations can leverage their existing XML data and schema when developing

ontologies for new fields. For the RDB/ORDB models, data mining techniques can be explored in an attempt to allow the inference of more IS-A relationships by adopting clustering techniques. The overall framework could also be extended to utilize a lexical database (such as WordNet), which would allow refining the classes and properties names in the generated ontologies. Furthermore, the framework could be enhanced to maintain a library of ontology components that would be used to guide the construction of new ontologies. Such an enhancement would allow the incorporation of existing components into the process of building new ontologies and thus benefit from past experience when building similar or new ontologies.

In the explicitness measurement methodology, the matching and explicitness computation phases could also be extended further. Currently, the matching phase performs a top-down one-to-one matching between ontology entities. An enhancement to the matching phase could incorporate one-to-many and/or bottom-up matching between entities. This would allow greater flexibility when measuring the explicitness between entities that differ in structure (e.g. data property for person's full-name matched with multiple data properties such as first-name, middle-initials, and last-name). Additionally, the computations phase could be semi-automated by developing an XSLT style-sheet that could transform the ontologies to be evaluated into the abstract/formal model used by the methodology, and in a format that can be ingested easily by a spreadsheet processor such as Microsoft Excel.

Pursuing such directions would ultimately improve the ontology engineering and evaluation processes and thus, accelerate the deployment of semantic-based solutions.

APPENDIX A: Research Publications

This research has resulted in the following publications:

1. Albarrak, Khalid M. and Sibley, Edgar H. 2009. Translating Relational & Object-Relational Database Models into OWL Models. Proceedings of the IEEE International Conference on Information Reuse and Integration, IRI 2009, 10-12 August 2009, Las Vegas, Nevada, USA. pp. 336-341
2. Albarrak, Khalid M. and Sibley, Edgar H. 2010. An Extensible Framework for Generating Ontology Models from Data Models, International Transactions on System Science and Applications (ITSSA), Vol. 6, No. 2/3, August 2010,. pp. 97-112.
3. Albarrak, Khalid M. and Sibley, Edgar H. 2011. A survey of methods that transform data models into Ontology models. Proceedings of the IEEE International Conference on Information Reuse and Integration, IRI 2011, 3-5 August 2011, Las Vegas, Nevada, USA. pp. 58-65
4. Albarrak, Khalid M. and Sibley, Edgar H. 2012. Methodology to Measure Expressivity between Ontology Models. To appear in <TBD> 2012.

APPENDIX B: Symmetry/Transitivity - Sample Relations

This appendix contains sample relations for the three patterns discussed in Chapter 4.

B.1 Sample Relations

This section contains samples of various relations -- both relation schemas and relation instances -- for the different patterns that were introduced in Chapter 4.

B.1.1 Pattern 1:

This pattern is used with one-to-one binary relations that are Symmetric or Transitive, and with one-to-many binary relations that are Transitive:

One-to-one - Symmetric:

- Table Schema: Person = (id (PK), name, gender, spouse-id (FK ref Person (id)))
- Table Instance:

Id	*Name*	*gender*	*Spouse-id*
1	John	M	2
2	Jane	F	
3	Riyadh	M	
4	Faisal	M	
5	Tami	F	6
6	Tom	M	

One-to-one - Transitive:

- Table Schema: Next-in-queue = (id (PK), name, next-id (FK ref Next-in-queue (id)))
- Table Instance:

Id	*Name*	*next-id*
1	Item 1	2
2	Item 2	3
3	Item 3	

One-to-many - Transitive:

- Table Schema: Employee = (id (PK), name, mgr-id (FK ref Employee (id)))

- Table Instance:

Id	*Name*	*Mgr-id*
1	Edgar	
2	Khalid	1
3	Jane	1
4	Faisal K.	2
5	Riyadh K.	2

B.1.2 Pattern 2:

This pattern is mostly used with many-to-many relations that are Symmetric, Transitive, both Symmetric and Transitive, or neither:

Many-to-many - Non-Symmetric and Non-Transitive:

- Table Schema: Person = (id (PK), name, gender)

Knows = (id1 (FK ref Person (id)), id2 (FK ref Person (id)), PK(id1, id2))

- Table Instance:

Person

Id	*Name*
1	Person 1
2	Person 2
3	Person 3
4	Person 4

Knows

id1	*id2*
1	2
1	3
2	1
3	4
4	1

Many-to-many - Symmetric and Non-Transitive:

- Table Schema: Country = (id (PK), name)

Border-with = (id1 (FK ref Country (id)), id2 (FK ref Country (id)) , PK(id1, id2))

- Table Instance:

Country

Id	*Name*
1	Jordan
2	Saudi
3	Iraq
4	Kuwait

Border-with

id1	*id2*
1	2
1	3
2	3
3	4
4	2

Many-to-many Non-Symmetric and Transitive:

- Table Schema: Product = (id (PK), name)

Composed-of = (id1 (FK ref Product (id)), id2 (FK ref Product (id)), PK(id1, id2))

- Table Instance:

Product

Id	*Name*
1	Product 1
2	Product 2
A	Product 3
B	Product 4
C	Product 5
D	Product 6
E	Product 7

Composed-of

id1	*id2*
1	A
1	B
2	A
2	C
A	D
B	D
B	E

Many-to-many Symmetric and Transitive:

- Table Schema: Person = (id (PK), name)

Sibling = (id1 (FK ref Person (id)), id2 (FK ref Person (id)) , PK(id1, id2))

- Table Instance:

Person

Id	*Name*
1	Khalid
2	Sami
3	Faris
4	Riyadh
5	Faisal

Sibling

id1	*id2*
1	2
2	3
4	5

B.1.3 Pattern 3:

This pattern is used with many-to-many binary relations that are both Symmetric and Transitive. This pattern is considered an alternative to Pattern 2 for binary relations that are *both* Symmetric and Transitive.

Many-to-many Symmetric and Transitive:

- Table Schema: Person = (id (PK), name, sibling-set-id (FK ref Sibling-set (set-id)))

Sibling-set = (set-id (PK))

- Table Instance:

Person

Id	*Name*	sibling-set-id
1	Khalid	S1
2	Sami	S1
3	Faris	S1
4	Riyadh	S2
5	Faisal	S2

Sibling-set

Set-id
S1
S2

APPENDIX C: Sample database scripts used in validation

This appendix contains the sample SQL scripts used to setup the source RBD instances in Case-Study One and Case-Study Two. These scripts contain the Data Definition Language (DDL) and Data Manipulation Language (DML) statements needed to create the RDB schemas and populate them with data. Note that these scripts were slightly modified to correct some of the data and enable setting up the schemas in an IBM DB2 environment.

1. Case-Study One:

In this case-study, I used a source RDB instance that was packaged with IBM DB2. This sample RDB instance models employees' project assignment in an enterprise. Below are the DDL and DML scripts used:

1.1. DDL Script for IBM DB2 Sample RDB Instance:

```
-- Clean up
ALTER TABLE "HR"."DEPARTMENT" DROP CONSTRAINT "RDE";
ALTER TABLE "HR"."DEPARTMENT" DROP CONSTRAINT "ROD";
ALTER TABLE "HR"."EMPLOYEE" DROP CONSTRAINT "RED";
ALTER TABLE "HR"."EMPPROJACT" DROP CONSTRAINT
"EMPPROJACT_EMPLOYEE_FK1";
ALTER TABLE "HR"."EMPPROJACT" DROP CONSTRAINT "REPAPA";
ALTER TABLE "HR"."EMP_PHOTO" DROP CONSTRAINT "FK_EMP_PHOTO";
ALTER TABLE "HR"."EMP_RESUME" DROP CONSTRAINT
"FK_EMP_RESUME";
ALTER TABLE "HR"."PROJECT" DROP CONSTRAINT "FK_PROJECT_1";
ALTER TABLE "HR"."PROJECT" DROP CONSTRAINT "FK_PROJECT_2";
ALTER TABLE "HR"."PROJECT" DROP CONSTRAINT "RPP";
ALTER TABLE "HR"."PROJECT_ACTIVITY" DROP CONSTRAINT
"PROJACT_ACT_FK";
ALTER TABLE "HR"."PROJECT_ACTIVITY" DROP CONSTRAINT "RPAP";
```

```
ALTER TABLE "HR"."ACTIVITY" DROP CONSTRAINT
"ACTIVITY_ACTKWD_UN";
ALTER TABLE "HR"."ACTIVITY" DROP CONSTRAINT "PK_ACT";
ALTER TABLE "HR"."DEPARTMENT" DROP CONSTRAINT
"DEPARTMENT_DEPTNAME_UN";
ALTER TABLE "HR"."DEPARTMENT" DROP CONSTRAINT
"PK_DEPARTMENT";
ALTER TABLE "HR"."EMPLOYEE" DROP CONSTRAINT "NUMBER";
ALTER TABLE "HR"."EMPLOYEE" DROP CONSTRAINT "PK_EMPLOYEE";
ALTER TABLE "HR"."EMPPROJACT" DROP CONSTRAINT "EMPPROJACT_PK";
ALTER TABLE "HR"."EMP_PHOTO" DROP CONSTRAINT "PK_EMP_PHOTO";
ALTER TABLE "HR"."EMP_RESUME" DROP CONSTRAINT
"PK_EMP_RESUME";
ALTER TABLE "HR"."PROJECT" DROP CONSTRAINT "PK_PROJECT";
ALTER TABLE "HR"."PROJECT_ACTIVITY" DROP CONSTRAINT
"PK_PROJACT";
DROP INDEX "HR"."XDEPT2";
DROP INDEX "HR"."XDEPT3";
DROP INDEX "HR"."XEMP2";
DROP INDEX "HR"."XPROJ2";
DROP TABLE "HR"."ACTIVITY";
DROP TABLE "HR"."DEPARTMENT";
DROP TABLE "HR"."EMPLOYEE";
DROP TABLE "HR"."EMPPROJACT";
DROP TABLE "HR"."EMP_PHOTO";
DROP TABLE "HR"."EMP_RESUME";
DROP TABLE "HR"."PROJECT";
DROP TABLE "HR"."PROJECT_ACTIVITY";
DROP SCHEMA "HR" RESTRICT;
-- ----------------------
-- CREATE SCHEMA & TABLES
CREATE SCHEMA "HR";
CREATE TABLE "HR"."ACTIVITY" (
                "ACTNO" SMALLINT NOT NULL,
                "ACTKWD" CHAR(6) NOT NULL,
                "ACTDESC" VARCHAR(20) NOT NULL
        );
CREATE TABLE "HR"."DEPARTMENT" (
                "DEPTNO" CHAR(3) NOT NULL,
                "DEPTNAME" VARCHAR(36) NOT NULL,
                "MGRNO" CHAR(6),
                "ADMRDEPT" CHAR(3) NOT NULL,
                "LOCATION" CHAR(16)
        );
```

```
CREATE TABLE "HR"."EMPLOYEE" (
                "EMPNO" CHAR(6) NOT NULL,
                "FIRSTNAME" VARCHAR(12) NOT NULL,
                "MIDINIT" CHAR(1),
                "LASTNAME" VARCHAR(15) NOT NULL,
                "WORKDEPT" CHAR(3),
                "PHONENO" CHAR(4),
                "HIREDATE" DATE,
                "JOB" CHAR(8),
                "EDLEVEL" SMALLINT NOT NULL,
                "SEX" CHAR(1),
                "BIRTHDATE" DATE,
                "SALARY" DECIMAL(9 , 2),
                "BONUS" DECIMAL(9 , 2),
                "COMM" DECIMAL(9 , 2)
        );
CREATE TABLE "HR"."EMPPROJACT" (
                "EMPNO" CHAR(6) NOT NULL,
                "PROJNO" CHAR(6) NOT NULL,
                "ACTNO" SMALLINT NOT NULL,
                "EMPTIME" DECIMAL(5 , 2),
                "EMSTDATE" VARCHAR(10) NOT NULL,
                "EMENDATE" VARCHAR(10)
        );
CREATE TABLE "HR"."EMP_PHOTO" (
                "EMPNO" CHAR(6) NOT NULL,
                "PHOTO_FORMAT" VARCHAR(10) NOT NULL,
                "PICTURE" BLOB(102400) INLINE LENGTH 140
        );
CREATE TABLE "HR"."EMP_RESUME" (
                "EMPNO" CHAR(6) NOT NULL,
                "RESUME_FORMAT" VARCHAR(10) NOT NULL,
                "RESUME" CLOB(5120) INLINE LENGTH 92
        );
CREATE TABLE "HR"."PROJECT" (
                "PROJNO" CHAR(6) NOT NULL,
                "PROJNAME" VARCHAR(24) NOT NULL DEFAULT ",
                "DEPTNO" CHAR(3) NOT NULL,
                "RESPEMP" CHAR(6) NOT NULL,
                "PRSTAFF" DECIMAL(5 , 2),
                "PRSTDATE" VARCHAR(10),
                "PRENDATE" VARCHAR(10),
                "MAJPROJ" CHAR(6)
        );
```

```
CREATE TABLE "HR"."PROJECT_ACTIVITY" (
                "PROJNO" CHAR(6) NOT NULL,
                "ACTNO" SMALLINT NOT NULL,
                "ACSTAFF" DECIMAL(5 , 2),
                "ACSTDATE" VARCHAR(10) NOT NULL,
                "ACENDATE" VARCHAR(10)
        );
CREATE INDEX "HR"."XDEPT2"
        ON "HR"."DEPARTMENT"
        ("MGRNO"            ASC)
        PCTFREE 10
        ALLOW REVERSE SCANS;
CREATE INDEX "HR"."XDEPT3"
        ON "HR"."DEPARTMENT"
        ("ADMRDEPT"                 ASC)
        PCTFREE 10
        ALLOW REVERSE SCANS;
CREATE INDEX "HR"."XEMP2"
        ON "HR"."EMPLOYEE"
        ("WORKDEPT"                 ASC)
        PCTFREE 10
        ALLOW REVERSE SCANS;
CREATE INDEX "HR"."XPROJ2"
        ON "HR"."PROJECT"
        ("RESPEMP"          ASC)
        PCTFREE 10
        ALLOW REVERSE SCANS;
-- ---------------------
ALTER TABLE "HR"."ACTIVITY" ADD CONSTRAINT
"ACTIVITY_ACTKWD_UN" UNIQUE
        ("ACTKWD");
ALTER TABLE "HR"."ACTIVITY" ADD CONSTRAINT "PK_ACT" PRIMARY KEY
        ("ACTNO");
ALTER TABLE "HR"."DEPARTMENT" ADD CONSTRAINT
"DEPARTMENT_DEPTNAME_UN" UNIQUE
        ("DEPTNAME");
ALTER TABLE "HR"."DEPARTMENT" ADD CONSTRAINT "PK_DEPARTMENT"
PRIMARY KEY
        ("DEPTNO");
ALTER TABLE "HR"."EMPLOYEE" ADD CONSTRAINT "NUMBER" CHECK
(PHONENO >= '0000' AND PHONENO <= '9999');
ALTER TABLE "HR"."EMPLOYEE" ADD CONSTRAINT "PK_EMPLOYEE"
PRIMARY KEY
        ("EMPNO");
```

```
ALTER TABLE "HR"."EMPPROJACT" ADD CONSTRAINT "EMPPROJACT_PK"
PRIMARY KEY
        ("EMPNO",
         "PROJNO",
         "ACTNO",
         "EMSTDATE");
ALTER TABLE "HR"."EMP_PHOTO" ADD CONSTRAINT "PK_EMP_PHOTO"
PRIMARY KEY
        ("EMPNO",
         "PHOTO_FORMAT");
ALTER TABLE "HR"."EMP_RESUME" ADD CONSTRAINT "PK_EMP_RESUME"
PRIMARY KEY
        ("EMPNO",
         "RESUME_FORMAT");
ALTER TABLE "HR"."PROJECT" ADD CONSTRAINT "PK_PROJECT" PRIMARY
KEY
        ("PROJNO");
ALTER TABLE "HR"."PROJECT_ACTIVITY" ADD CONSTRAINT "PK_PROJACT"
PRIMARY KEY
        ("PROJNO",
         "ACTNO",
         "ACSTDATE");
ALTER TABLE "HR"."DEPARTMENT" ADD CONSTRAINT "RDE" FOREIGN KEY
        ("MGRNO")
        REFERENCES "HR"."EMPLOYEE"
        ("EMPNO")
        ON DELETE SET NULL;
ALTER TABLE "HR"."DEPARTMENT" ADD CONSTRAINT "ROD" FOREIGN
KEY
        ("ADMRDEPT")
        REFERENCES "HR"."DEPARTMENT"
        ("DEPTNO")
        ON DELETE CASCADE;
ALTER TABLE "HR"."EMPLOYEE" ADD CONSTRAINT "RED" FOREIGN KEY
        ("WORKDEPT")
        REFERENCES "HR"."DEPARTMENT"
        ("DEPTNO")
        ON DELETE SET NULL;
ALTER TABLE "HR"."EMPPROJACT" ADD CONSTRAINT
"EMPPROJACT_EMPLOYEE_FK1" FOREIGN KEY
        ("EMPNO")
        REFERENCES "HR"."EMPLOYEE"
        ("EMPNO")
        ON DELETE CASCADE;
```

```
ALTER TABLE "HR"."EMPPROJACT" ADD CONSTRAINT "REPAPA" FOREIGN
KEY
        ("PROJNO",
         "ACTNO",
         "EMSTDATE")
        REFERENCES "HR"."PROJECT_ACTIVITY"
        ("PROJNO",
         "ACTNO",
         "ACSTDATE")
        ON DELETE RESTRICT;
ALTER TABLE "HR"."EMP_PHOTO" ADD CONSTRAINT "FK_EMP_PHOTO"
FOREIGN KEY
        ("EMPNO")
        REFERENCES "HR"."EMPLOYEE"
        ("EMPNO")
        ON DELETE RESTRICT;
ALTER TABLE "HR"."EMP_RESUME" ADD CONSTRAINT "FK_EMP_RESUME"
FOREIGN KEY
        ("EMPNO")
        REFERENCES "HR"."EMPLOYEE"
        ("EMPNO")
        ON DELETE RESTRICT;
ALTER TABLE "HR"."PROJECT" ADD CONSTRAINT "FK_PROJECT_1"
FOREIGN KEY
        ("DEPTNO")
        REFERENCES "HR"."DEPARTMENT"
        ("DEPTNO")
        ON DELETE RESTRICT;
ALTER TABLE "HR"."PROJECT" ADD CONSTRAINT "FK_PROJECT_2"
FOREIGN KEY
        ("RESPEMP")
        REFERENCES "HR"."EMPLOYEE"
        ("EMPNO")
        ON DELETE RESTRICT;
ALTER TABLE "HR"."PROJECT" ADD CONSTRAINT "RPP" FOREIGN KEY
        ("MAJPROJ")
        REFERENCES "HR"."PROJECT"
        ("PROJNO")
        ON DELETE CASCADE;
ALTER TABLE "HR"."PROJECT_ACTIVITY" ADD CONSTRAINT
"PROJACT_ACT_FK" FOREIGN KEY
        ("ACTNO")
        REFERENCES "HR"."ACTIVITY"
        ("ACTNO")
```

```
        ON DELETE CASCADE;
ALTER TABLE "HR"."PROJECT_ACTIVITY" ADD CONSTRAINT "RPAP"
FOREIGN KEY
        ("PROJNO")
        REFERENCES "HR"."PROJECT"
        ("PROJNO")
        ON DELETE RESTRICT;

COMMENT ON TABLE "HR"."ACTIVITY" IS
'Activities carried out in projects';
COMMENT ON TABLE "HR"."DEPARTMENT" IS
'Departments in this enterprise';
COMMENT ON TABLE "HR"."EMPLOYEE" IS
'Employees in this enterprise';
COMMENT ON TABLE "HR"."EMPPROJACT" IS
'Activities performed by an employee in a project on a given start date.';
COMMENT ON TABLE "HR"."EMP_PHOTO" IS
'Employee Photos. An Employee can have at most one photo of the same format. ';
COMMENT ON TABLE "HR"."EMP_RESUME" IS
'Employee Resumes. An employee can have at most one resume of the same format';
COMMENT ON TABLE "HR"."PROJECT" IS
'Projects carried out by this enterprise. A project can be part of a larger project.';
COMMENT ON TABLE "HR"."PROJECT_ACTIVITY" IS
'Activities performed in a project with a given start/end date';
```

1.2. DML Script for IBM DB2 Sample RDB Instance:

```
SET SCHEMA HR;

-- clean up
ALTER TABLE department ALTER FOREIGN KEY rde NOT ENFORCED;
ALTER TABLE department ALTER FOREIGN KEY rod NOT ENFORCED;
ALTER TABLE employee   ALTER FOREIGN KEY red NOT ENFORCED;
ALTER TABLE empprojact ALTER FOREIGN KEY empprojact_employee_fk1 NOT
ENFORCED;
ALTER TABLE empprojact ALTER FOREIGN KEY repapa               NOT ENFORCED;
ALTER TABLE emp_photo   ALTER FOREIGN KEY fk_emp_photo NOT ENFORCED;
ALTER TABLE emp_resume  ALTER FOREIGN KEY fk_emp_resume NOT ENFORCED;
ALTER TABLE project ALTER FOREIGN KEY fk_project_1 NOT ENFORCED;
ALTER TABLE project ALTER FOREIGN KEY fk_project_2 NOT ENFORCED;
ALTER TABLE project ALTER FOREIGN KEY rpp NOT ENFORCED;
ALTER TABLE project_activity ALTER FOREIGN KEY projact_act_fk NOT ENFORCED;
ALTER TABLE project_activity ALTER FOREIGN KEY rpap NOT ENFORCED;
```

```
DELETE FROM activity;
DELETE FROM department;
DELETE FROM employee;
DELETE FROM empprojact;
DELETE FROM emp_photo;
DELETE FROM emp_resume;
DELETE FROM project;
DELETE FROM project_activity;

-- Table: Activity
INSERT INTO activity (actno, actkwd, actdesc) VALUES
(10,'MANAGE','MANAGE/ADVISE'),
(20,'ECOST ','ESTIMATE COST'),
(30,'DEFINE','DEFINE SPECS'),
(40,'LEADPR','LEAD PROGRAM/DESIGN'),
(50,'SPECS ','WRITE SPECS'),
(60,'LOGIC ','DESCRIBE LOGIC'),
(70,'CODE  ','CODE PROGRAMS'),
(80,'TEST  ','TEST PROGRAMS'),
(90,'ADMQS ','ADM QUERY SYSTEM'),
(100,'TEACH ','TEACH CLASSES'),
(110,'COURSE','DEVELOP COURSES'),
(120,'STAFF ','PERS AND STAFFING'),
(130,'OPERAT','OPER COMPUTER SYS'),
(140,'MAINT ','MAINT SOFTWARE SYS'),
(150,'ADMSYS','ADM OPERATING SYS'),
(160,'ADMDB ','ADM DATA BASES'),
(170,'ADMDC ','ADM DATA COMM'),
(180,'DOC   ','DOCUMENT'),
--
(190,'INFRA ','INFRASRUCTURE SUPORT'),
(200,'ENABLE','ENABLEMENT SUPPORT')
;
-- Table: DEPARTMENT
INSERT INTO department (deptno, deptname, mgrno, admrdept, location) VALUES
('A00','SPIFFY COMPUTER SERVICE DIV.','000010','A00',NULL),
('B01','PLANNING','000020','A00',NULL),
('C01','INFORMATION CENTER','000030','A00',NULL),
('D01','DEVELOPMENT CENTER', '300010','A00',NULL),
('D11','MANUFACTURING SYSTEMS','000060','D01',NULL),
('D21','ADMINISTRATION SYSTEMS','000070','D01',NULL),
('E01','SUPPORT SERVICES','000050','A00',NULL),
('E11','OPERATIONS','000090','E01',NULL),
('E21','SOFTWARE SUPPORT','000100','E01',NULL),
('F22','BRANCH OFFICE F2', '300080','E01',NULL),
('G22','BRANCH OFFICE G2', '300090','E01',NULL),
('H22','BRANCH OFFICE H2', NULL,'E01',NULL),
('I22','BRANCH OFFICE I2', NULL,'E01',NULL),
```

```
('J22','BRANCH OFFICE J2', NULL,'E01',NULL),
--
('B11','INFORMATION PLANNING', '300020','B01',NULL),
('B21','DEVELOPMENT PLANNING', '300030','B01',NULL),
('B31','SUPPORT PLANNING', '300040','B01',NULL),
('C11','INFRASTRUCTURE', '300050','C01',NULL),
('C21','DATA STORAGE', '300060','C01',NULL),
('C31','DATA PRESENTATION', '300070','C01',NULL)
;
-- Table: EMPLOYEE
INSERT INTO employee (empno, firstname, midinit, lastname, workdept, phoneno, hiredate, job,
edlevel, sex, birthdate, salary, bonus, comm) VALUES
('000010','CHRISTINE','I','HAAS','A00','3978',to_date ('19950101','YYYYMMDD'),'PRES
',18,'F',to_date ('19630824','YYYYMMDD'),+0152750.00,+0001000.00,+0004220.00),
('000020','MICHAEL'  ,'L','THOMPSON','B01','3476',to_date
('20031010','YYYYMMDD'),'MANAGER ',18,'M',to_date
('19780202','YYYYMMDD'),+0094250.00,+0000800.00,+0003300.00),
('000030','SALLY'    ,'A','KWAN','C01','4738',to_date ('20050405','YYYYMMDD'),'MANAGER
',20,'F',to_date ('19710511','YYYYMMDD'),+0098250.00,+0000800.00,+0003060.00),
('000050','JOHN'     ,'B','GEYER','E01','6789',to_date ('19790817','YYYYMMDD'),'MANAGER
',16,'M',to_date ('19550915','YYYYMMDD'),+0080175.00,+0000800.00,+0003214.00),
('000060','IRVING'   ,'F','STERN','D11','6423',to_date ('20030914','YYYYMMDD'),'MANAGER
',16,'M',to_date ('19750707','YYYYMMDD'),+0072250.00,+0000500.00,+0002580.00),
('000070','EVA'      ,'D','PULASKI','D21','7831',to_date
('20050930','YYYYMMDD'),'MANAGER ',16,'F',to_date
('20030526','YYYYMMDD'),+0096170.00,+0000700.00,+0002893.00),
('000090','EILEEN'   ,'W','HENDERSON','E11','5498',to_date
('20000815','YYYYMMDD'),'MANAGER ',16,'F',to_date
('19710515','YYYYMMDD'),+0089750.00,+0000600.00,+0002380.00),
('000100','THEODORE' ,'Q','SPENSER','E21','0972',to_date
('20000619','YYYYMMDD'),'MANAGER ',14,'M',to_date
('19801218','YYYYMMDD'),+0086150.00,+0000500.00,+0002092.00),
('000110','VINCENZO' ,'G','LUCCHESSI','A00','3490',to_date
('19880516','YYYYMMDD'),'SALESREP',19,'M',to_date
('19591105','YYYYMMDD'),+0066500.00,+0000900.00,+0003720.00),
('000120','SEAN'     ,NULL,'OCONNELL','A00','2167',to_date
('19931205','YYYYMMDD'),'CLERK   ',14,'M',to_date
('19721018','YYYYMMDD'),+0049250.00,+0000600.00,+0002340.00),
('000130','DELORES'  ,'M','QUINTANA','C01','4578',to_date
('20010728','YYYYMMDD'),'ANALYST ',16,'F',to_date
('19550915','YYYYMMDD'),+0073800.00,+0000500.00,+0001904.00),
('000140','HEATHER'  ,'A','NICHOLLS','C01','1793',to_date
('20061215','YYYYMMDD'),'ANALYST ',18,'F',to_date
('19760119','YYYYMMDD'),+0068420.00,+0000600.00,+0002274.00),
('000150','BRUCE'    ,NULL,'ADAMSON','D11','4510',to_date
('20020212','YYYYMMDD'),'DESIGNER',16,'M',to_date
('19770517','YYYYMMDD'),+0055280.00,+0000500.00,+0002022.00),
```

('000160','ELIZABETH','R','PIANKA','D11','3782',to_date
('20061011','YYYYMMDD'),'DESIGNER',17,'F',to_date
('19800412','YYYYMMDD'),+0062250.00,+0000400.00,+0001780.00),
('000170','MASATOSHI','J','YOSHIMURA','D11','2890',to_date
('19990915','YYYYMMDD'),'DESIGNER',16,'M',to_date
('19810105','YYYYMMDD'),+0044680.00,+0000500.00,+0001974.00),
('000180','MARILYN' ,'S','SCOUTTEN','D11','1682',to_date
('20030707','YYYYMMDD'),'DESIGNER',17,'F',to_date
('19790221','YYYYMMDD'),+0051340.00,+0000500.00,+0001707.00),
('000190','JAMES' ,'H','WALKER','D11','2986',to_date
('20040726','YYYYMMDD'),'DESIGNER',16,'M',to_date
('19820625','YYYYMMDD'),+0050450.00,+0000400.00,+0001636.00),
('000200','DAVID' ,NULL,'BROWN','D11','4501',to_date
('20020303','YYYYMMDD'),'DESIGNER',16,'M',to_date
('19710529','YYYYMMDD'),+0057740.00,+0000600.00,+0002217.00),
('000210','WILLIAM' ,'T','JONES','D11','0942',to_date
('19980411','YYYYMMDD'),'DESIGNER',17,'M',to_date
('20030223','YYYYMMDD'),+0068270.00,+0000400.00,+0001462.00),
('000220','JENNIFER' ,'K','LUTZ','D11','0672',to_date
('19980829','YYYYMMDD'),'DESIGNER',18,'F',to_date
('19780319','YYYYMMDD'),+0049840.00,+0000600.00,+0002387.00),
('000230','JAMES' ,'J','JEFFERSON','D21','2094',to_date
('19961121','YYYYMMDD'),'CLERK ',14,'M',to_date
('19800530','YYYYMMDD'),+0042180.00,+0000400.00,+0001774.00),
('000240','SALVATORE','M','MARINO','D21','3780',to_date
('20041205','YYYYMMDD'),'CLERK ',17,'M',to_date
('20020331','YYYYMMDD'),+0048760.00,+0000600.00,+0002301.00),
('000250','DANIEL' ,'S','SMITH','D21','0961',to_date ('19991030','YYYYMMDD'),'CLERK
',15,'M',to_date ('19691112','YYYYMMDD'),+0049180.00,+0000400.00,+0001534.00),
('000260','SYBIL' ,'P','JOHNSON','D21','8953',to_date ('20050911','YYYYMMDD'),'CLERK
',16,'F',to_date ('19761005','YYYYMMDD'),+0047250.00,+0000300.00,+0001380.00),
('000270','MARIA' ,'L','PEREZ','D21','9001',to_date ('20060930','YYYYMMDD'),'CLERK
',15,'F',to_date ('20030526','YYYYMMDD'),+0037380.00,+0000500.00,+0002190.00),
('000280','ETHEL' ,'R','SCHNEIDER','E11','8997',to_date
('19970324','YYYYMMDD'),'OPERATOR',17,'F',to_date
('19760328','YYYYMMDD'),+0036250.00,+0000500.00,+0002100.00),
('000290','JOHN' ,'R','PARKER','E11','4502',to_date
('20060530','YYYYMMDD'),'OPERATOR',12,'M',to_date
('19850709','YYYYMMDD'),+0035340.00,+0000300.00,+0001227.00),
('000300','PHILIP' ,NULL,'SMITH','E11','2095',to_date
('20020619','YYYYMMDD'),'OPERATOR',14,'M',to_date
('19761027','YYYYMMDD'),+0037750.00,+0000400.00,+0001420.00),
('000310','MAUDE' ,'F','SETRIGHT','E11','3332',to_date
('19940912','YYYYMMDD'),'OPERATOR',12,'F',to_date
('19610421','YYYYMMDD'),+0035900.00,+0000300.00,+0001272.00),
('000320','RAMLAL' ,'V','MEHTA','E21','9990',to_date
('19950707','YYYYMMDD'),'FIELDREP',16,'M',to_date
('19620811','YYYYMMDD'),+0039950.00,+0000400.00,+0001596.00),

('000330','WING'   ,NULL,'LEE','E21','2103',to_date
('20060223','YYYYMMDD'),'FIELDREP',14,'M',to_date
('19710718','YYYYMMDD'),+0045370.00,+0000500.00,+0002030.00),
('000340','JASON'   ,'R','GOUNOT','E21','5698',to_date
('19770505','YYYYMMDD'),'FIELDREP',16,'M',to_date
('19560517','YYYYMMDD'),+0043840.00,+0000500.00,+0001907.00),
('200010','DIAN'   ,'J','HEMMINGER','A00','3978',to_date
('19950101','YYYYMMDD'),'SALESREP',18,'F',to_date
('19730814','YYYYMMDD'),+0046500.00,+0001000.00,+0004220.00),
('200120','GREG'   ,NULL,'ORLANDO','A00','2167',to_date
('20020505','YYYYMMDD'),'CLERK   ',14,'M',to_date
('19721018','YYYYMMDD'),+0039250.00,+0000600.00,+0002340.00),
('200140','KIM'   ,'N','NATZ','C01','1793',to_date ('20061215','YYYYMMDD'),'ANALYST
',18,'F',to_date ('19760119','YYYYMMDD'),+0068420.00,+0000600.00,+0002274.00),
('200170','KIYOSHI' ,NULL,'YAMAMOTO','D11','2890',to_date
('20050915','YYYYMMDD'),'DESIGNER',16,'M',to_date
('19810105','YYYYMMDD'),+0064680.00,+0000500.00,+0001974.00),
('200220','REBA'   ,'K','JOHN','D11','0672',to_date
('20050829','YYYYMMDD'),'DESIGNER',18,'F',to_date
('19780319','YYYYMMDD'),+0069840.00,+0000600.00,+0002387.00),
('200240','ROBERT'   ,'M','MONTEVERDE','D21','3780',to_date
('20041205','YYYYMMDD'),'CLERK   ',17,'M',to_date
('19840331','YYYYMMDD'),+0037760.00,+0000600.00,+0002301.00),
('200280','EILEEN'   ,'R','SCHWARTZ','E11','8997',to_date
('19970324','YYYYMMDD'),'OPERATOR',17,'F',to_date
('19660328','YYYYMMDD'),+0046250.00,+0000500.00,+0002100.00),
('200310','MICHELLE' ,'F','SPRINGER','E11','3332',to_date
('19940912','YYYYMMDD'),'OPERATOR',12,'F',to_date
('19610421','YYYYMMDD'),+0035900.00,+0000300.00,+0001272.00),
('200330','HELENA'   ,NULL,'WONG','E21','2103',to_date
('20060223','YYYYMMDD'),'FIELDREP',14,'F',to_date
('19710718','YYYYMMDD'),+0035370.00,+0000500.00,+0002030.00),
('200340','ROY'   ,'R','ALONZO','E21','5698',to_date
('19970705','YYYYMMDD'),'FIELDREP',16,'M',to_date
('19560517','YYYYMMDD'),+0031840.00,+0000500.00,+0001907.00),
--
('300010','DAVE' ,'A','TBD','D01','1001',to_date ('20110501','YYYYMMDD'),'MANAGER
',20,'M',to_date ('19800101','YYYYMMDD'),+0086150.00,+0000500.00,+0002092.00),
('300020','MAVE' ,'B','TBD','B11','1002',to_date ('20110502','YYYYMMDD'),'MANAGER
',20,'F',to_date ('19800102','YYYYMMDD'),+0086150.00,+0000500.00,+0002092.00),
('300030','SALLY','C','TBD','B21','1003',to_date ('20110503','YYYYMMDD'),'MANAGER
',20,'F',to_date ('19800103','YYYYMMDD'),+0086150.00,+0000500.00,+0002092.00),
('300040','MOLLY','D','TBD','B31','1004',to_date ('20110504','YYYYMMDD'),'MANAGER
',20,'F',to_date ('19800104','YYYYMMDD'),+0086150.00,+0000500.00,+0002092.00),
('300050','KALLY','E','TBD','C11','1005',to_date ('20110505','YYYYMMDD'),'MANAGER
',20,'F',to_date ('19800105','YYYYMMDD'),+0086150.00,+0000500.00,+0002092.00),
('300060','JIM' ,'F','TBD','C21','1006',to_date ('20110506','YYYYMMDD'),'MANAGER
',20,'M',to_date ('19800106','YYYYMMDD'),+0086150.00,+0000500.00,+0002092.00),

('300070','KIM ','G','TBD','C31','1007',to_date ('20110507','YYYYMMDD'),'MANAGER
',20,'F',to_date ('19800107','YYYYMMDD'),+0086150.00,+0000500.00,+0002092.00),
('300080','SIM ','H','TBD','F22','1008',to_date ('20110508','YYYYMMDD'),'MANAGER
',20,'M',to_date ('19800108','YYYYMMDD'),+0086150.00,+0000500.00,+0002092.00),
('300090','TIM ','I','TBD','G22','1009',to_date ('20110509','YYYYMMDD'),'MANAGER
',20,'M',to_date ('19800109','YYYYMMDD'),+0086150.00,+0000500.00,+0002092.00)
;
-- Table: EMPPROJACT
INSERT INTO empproject (empno, projno, actno, emptime, emstdate, emendate) VALUES
('000010','AD3100',10,+000.50,'20020101','20020701'),
('000070','AD3110',10,+001.00,'20020101','20030201'),
('000230','AD3111',60,+001.00,'20020101','20020315'),
('000230','AD3111',60,+000.50,'20020315','20020415'),
('000230','AD3111',70,+000.50,'20020315','20021015'),
('000230','AD3111',80,+000.50,'20020415','20021015'),
('000230','AD3111',180,+000.50,'20021015','20030101'),
('000240','AD3111',70,+001.00,'20020215','20020915'),
('000240','AD3111',80,+001.00,'20020915','20030101'),
('000250','AD3112',60,+001.00,'20020101','20020201'),
('000250','AD3112',60,+000.50,'20020201','20020315'),
('000250','AD3112',60,+001.00,'20030101','20030201'),
('000250','AD3112',70,+000.50,'20020201','20020315'),
('000250','AD3112',70,+001.00,'20020315','20020815'),
('000250','AD3112',70,+000.25,'20020815','20021015'),
('000250','AD3112',80,+000.25,'20020815','20021015'),
('000250','AD3112',80,+000.50,'20021015','20021201'),
('000250','AD3112',180,+000.50,'20020815','20030101'),
('000260','AD3113',70,+000.50,'20020615','20020701'),
('000260','AD3113',70,+001.00,'20020701','20030201'),
('000260','AD3113',80,+001.00,'20020101','20020301'),
('000260','AD3113',80,+000.50,'20020301','20020415'),
('000260','AD3113',180,+000.50,'20020301','20020415'),
('000260','AD3113',180,+001.00,'20020415','20020601'),
('000260','AD3113',180,+001.00,'20020601','20020701'),
('000270','AD3113',60,+000.50,'20020301','20020401'),
('000270','AD3113',60,+001.00,'20020401','20020901'),
('000270','AD3113',60,+000.25,'20020901','20021015'),
('000270','AD3113',70,+000.75,'20020901','20021015'),
('000270','AD3113',70,+001.00,'20021015','20030201'),
('000270','AD3113',80,+001.00,'20020101','20020301'),
('000270','AD3113',80,+000.50,'20020301','20020401'),
('000030','IF1000',10,+000.50,'20020601','20030101'),
('000130','IF1000',90,+001.00,'20021001','20030101'),
('000130','IF1000',100,+000.50,'20021001','20030101'),
('000140','IF1000',90,+000.50,'20021001','20030101'),
('000030','IF2000',10,+000.50,'20020101','20030101'),
('000140','IF2000',100,+001.00,'20020101','20020301'),
('000140','IF2000',100,+000.50,'20020301','20020701'),

('000140','IF2000',110,+000.50,'20020301','20020701'),
('000140','IF2000',110,+000.50,'20021001','20030101'),
('000010','MA2100',10,+000.50,'20020101','20021101'),
('000110','MA2100',20,+001.00,'20020101','20030301'),
('000010','MA2110',10,+001.00,'20020101','20030201'),
('000200','MA2111',50,+001.00,'20020101','20020615'),
('000200','MA2111',60,+001.00,'20020615','20030201'),
('000220','MA2111',40,+001.00,'20020101','20030201'),
('000150','MA2112',60,+001.00,'20020101','20020715'),
('000150','MA2112',180,+001.00,'20020715','20030201'),
('000170','MA2112',60,+001.00,'20020101','20030601'),
('000170','MA2112',70,+001.00,'20020601','20030201'),
('000190','MA2112',70,+001.00,'20020101','20021001'),
('000190','MA2112',80,+001.00,'20021001','20031001'),
('000160','MA2113',60,+001.00,'20020715','20030201'),
('000170','MA2113',80,+001.00,'20020101','20030201'),
('000180','MA2113',70,+001.00,'20020401','20020615'),
('000210','MA2113',80,+000.50,'20021001','20030201'),
('000210','MA2113',180,+000.50,'20021001','20030201'),
('000050','OP1000',10,+000.25,'20020101','20030201'),
('000090','OP1010',10,+001.00,'20020101','20030201'),
('000280','OP1010',130,+001.00,'20020101','20030201'),
('000290','OP1010',130,+001.00,'20020101','20030201'),
('000300','OP1010',130,+001.00,'20020101','20030201'),
('000310','OP1010',130,+001.00,'20020101','20030201'),
('000050','OP2010',10,+000.75,'20020101','20030201'),
('000100','OP2010',10,+001.00,'20020101','20030201'),
('000320','OP2011',140,+000.75,'20020101','20030201'),
('000320','OP2011',150,+000.25,'20020101','20030201'),
('000330','OP2012',140,+000.25,'20020101','20030201'),
('000330','OP2012',160,+000.75,'20020101','20030201'),
('000340','OP2013',140,+000.50,'20020101','20030201'),
('000340','OP2013',170,+000.50,'20020101','20030201'),
('000020','PL2100',30,+001.00,'20020101','20020915')
;
-- Table: EMP_PHOTO
INSERT INTO emp_photo (empno, photo_format) VALUES
('000130','bitmap'),
('000130','gif'),
('000140','bitmap'),
('000140','gif'),
('000150','bitmap'),
('000150','gif'),
('000190','bitmap'),
('000190','gif')
;
-- Table: EMP_RESUME
INSERT INTO emp_resume (empno, resume_format) VALUES

```
('000130','ascii'),
('000130','html'),
('000140','ascii'),
('000140','html'),
('000150','ascii'),
('000150','html'),
('000190','ascii'),
('000190','html')
;
-- Table: PROJECT
INSERT INTO project (projno, projname, deptno, respemp, prstaff, prstdate, prendate, majproj)
VALUES
('AD3100','ADMIN SERVICES','D01','000010',+006.50,'20020101','20030201',NULL),
('AD3110','GENERAL ADMIN
SYSTEMS','D21','000070',+006.00,'20020101','20030201','AD3100'),
('AD3111','PAYROLL
PROGRAMMING','D21','000230',+002.00,'20020101','20030201','AD3110'),
('AD3112','PERSONNEL
PROGRAMMING','D21','000250',+001.00,'20020101','20030201','AD3110'),
('AD3113','ACCOUNT
PROGRAMMING','D21','000270',+002.00,'20020101','20030201','AD3110'),
('IF1000','QUERY SERVICES','C01','000030',+002.00,'20020101','20030201',NULL),
('IF2000','USER EDUCATION','C01','000030',+001.00,'20020101','20030201',NULL),
('MA2100','WELD LINE
AUTOMATION','D01','000010',+012.00,'20020101','20030201',NULL),
('MA2110','W L PROGRAMMING','D11','000060',+009.00,'20020101','20030201','MA2100'),
('MA2111','W L PROGRAM
DESIGN','D11','000220',+002.00,'20020101','19821201','MA2110'),
('MA2112','W L ROBOT DESIGN','D11','000150',+003.00,'20020101','19821201','MA2110'),
('MA2113','W L PROD CONT
PROGS','D11','000160',+003.00,'20020215','19821201','MA2110'),
('OP1000','OPERATION SUPPORT','E01','000050',+006.00,'20020101','20030201',NULL),
('OP1010','OPERATION','E11','000090',+005.00,'20020101','20030201','OP1000'),
('OP2000','GEN SYSTEMS SERVICES','E01','000050',+005.00,'20020101','20030201',NULL),
('OP2010','SYSTEMS SUPPORT','E21','000100',+004.00,'20020101','20030201','OP2000'),
('OP2011','SCP SYSTEMS SUPPORT','E21','000320',+001.00,'20020101','20030201','OP2010'),
('OP2012','APPLICATIONS
SUPPORT','E21','000330',+001.00,'20020101','20030201','OP2010'),
('OP2013','DB/DC SUPPORT','E21','000340',+001.00,'20020101','20030201','OP2010'),
('PL2100','WELD LINE PLANNING','B01','000020',+001.00,'20020101','20020915','MA2100')
;
-- Table: PROJECT_ACTIVITY
INSERT INTO project_activity(projno, actno, acstaff,acstdate, acendate) VALUES
('AD3100',10, NULL,'20020101',NULL),
('AD3110',10, NULL,'20020101',NULL),
('AD3111',60, NULL,'20020101',NULL),
('AD3111',60, NULL,'20020315',NULL),
('AD3111',70, NULL,'20020315',NULL),
```

('AD3111',80, NULL,'20020415',NULL),
('AD3111',180, NULL,'20021015',NULL),
('AD3111',70, NULL,'20020215',NULL),
('AD3111',80, NULL,'20020915',NULL),
('AD3112',60, NULL,'20020101',NULL),
('AD3112',60, NULL,'20020201',NULL),
('AD3112',60, NULL,'20030101',NULL),
('AD3112',70, NULL,'20020201',NULL),
('AD3112',70, NULL,'20020315',NULL),
('AD3112',70, NULL,'20020815',NULL),
('AD3112',80, NULL,'20020815',NULL),
('AD3112',80, NULL,'20021015',NULL),
('AD3112',180, NULL,'20020815',NULL),
('AD3113',70, NULL,'20020615',NULL),
('AD3113',70, NULL,'20020701',NULL),
('AD3113',80, NULL,'20020101',NULL),
('AD3113',80, NULL,'20020301',NULL),
('AD3113',180, NULL,'20020301',NULL),
('AD3113',180, NULL,'20020415',NULL),
('AD3113',180, NULL,'20020601',NULL),
('AD3113',60, NULL,'20020301',NULL),
('AD3113',60, NULL,'20020401',NULL),
('AD3113',60, NULL,'20020901',NULL),
('AD3113',70, NULL,'20020901',NULL),
('AD3113',70, NULL,'20021015',NULL),
('IF1000',10, NULL,'20020601',NULL),
('IF1000',90, NULL,'20021001',NULL),
('IF1000',100, NULL,'20021001',NULL),
('IF2000',10, NULL,'20020101',NULL),
('IF2000',100, NULL,'20020101',NULL),
('IF2000',100, NULL,'20020301',NULL),
('IF2000',110, NULL,'20020301',NULL),
('IF2000',110, NULL,'20021001',NULL),
('MA2100',10, NULL,'20020101',NULL),
('MA2100',20, NULL,'20020101',NULL),
('MA2110',10, NULL,'20020101',NULL),
('MA2111',50, NULL,'20020101',NULL),
('MA2111',60, NULL,'20020615',NULL),
('MA2111',40, NULL,'20020101',NULL),
('MA2112',60, NULL,'20020101',NULL),
('MA2112',180, NULL,'20020715',NULL),
('MA2112',70, NULL,'20020601',NULL),
('MA2112',70, NULL,'20020101',NULL),
('MA2112',80, NULL,'20021001',NULL),
('MA2113',60, NULL,'20020715',NULL),
('MA2113',80, NULL,'20020101',NULL),
('MA2113',70, NULL,'20020401',NULL),
('MA2113',80, NULL,'20021001',NULL),

```
('MA2113',180, NULL,'20021001',NULL),
('OP1000',10, NULL,'20020101',NULL),
('OP1010',10, NULL,'20020101',NULL),
('OP1010',130, NULL,'20020101',NULL),
('OP2010',10, NULL,'20020101',NULL),
('OP2011',140, NULL,'20020101',NULL),
('OP2011',150, NULL,'20020101',NULL),
('OP2012',140, NULL,'20020101',NULL),
('OP2012',160, NULL,'20020101',NULL),
('OP2013',140, NULL,'20020101',NULL),
('OP2013',170, NULL,'20020101',NULL),
('PL2100',30, NULL,'20020101',NULL)
;
ALTER TABLE department ALTER FOREIGN KEY rde ENFORCED;
ALTER TABLE department ALTER FOREIGN KEY rod ENFORCED;
ALTER TABLE employee   ALTER FOREIGN KEY red ENFORCED;
ALTER TABLE empproject ALTER FOREIGN KEY empproject_employee_fk1 ENFORCED;
ALTER TABLE empproject ALTER FOREIGN KEY repapa              ENFORCED;
ALTER TABLE emp_photo   ALTER FOREIGN KEY fk_emp_photo ENFORCED;
ALTER TABLE emp_resume  ALTER FOREIGN KEY fk_emp_resume ENFORCED;
ALTER TABLE project ALTER FOREIGN KEY fk_project_1 ENFORCED;
ALTER TABLE project ALTER FOREIGN KEY fk_project_2 ENFORCED;
ALTER TABLE project ALTER FOREIGN KEY rpp ENFORCED;
ALTER TABLE project_activity ALTER FOREIGN KEY project_act_fk ENFORCED;
ALTER TABLE project_activity ALTER FOREIGN KEY rpap ENFORCED;
```

2. Case-Study Two:

In this case-study, I used as a source the RDB instance that was packaged with MS SQL Server. This sample RDB instance models an e-commerce business. Below are the DDL and DML scripts used:

2.1. DDL Script for MS SQL Server Sample RDB Instance:

```
SET SCHEMA ECOMM;

-- Clean up
ALTER TABLE Invoices DROP CONSTRAINT Invoice_Status_Codes_Invoices;
ALTER TABLE Invoices DROP CONSTRAINT Orders_Invoices;
ALTER TABLE Orders DROP CONSTRAINT Order_Status_Codes_Orders;
ALTER TABLE Orders DROP CONSTRAINT Customers_1_Orders;
ALTER TABLE Shipments DROP CONSTRAINT Orders_Shipments;
ALTER TABLE Shipments DROP CONSTRAINT Invoices_Shipments;
ALTER TABLE Shipment_Items DROP CONSTRAINT Shipments_Shipment_Items;
```

```
ALTER TABLE Shipment_Items DROP CONSTRAINT Order_Items_Shipment_Items;
ALTER TABLE Order_Items DROP CONSTRAINT Order_Item_Status_Order_Items;
ALTER TABLE Order_Items DROP CONSTRAINT Products_Order_Items;
ALTER TABLE Order_Items DROP CONSTRAINT Orders_Order_Items;
ALTER TABLE Products DROP CONSTRAINT Ref_Product_Types_Products;
ALTER TABLE Customer_Payment_Methods DROP CONSTRAINT
Customers_Customer_Payment_Methods;
ALTER TABLE Customer_Payment_Methods DROP CONSTRAINT
Ref_Payment_Methods_Customer_Payment_Methods;
ALTER TABLE Payments DROP CONSTRAINT Invoices_Payments;
ALTER TABLE Ref_Product_Types DROP CONSTRAINT
Ref_Product_Types_Ref_Product_Types;

DROP TABLE Invoices;
DROP TABLE Orders;
DROP TABLE Ref_Order_Status_Codes ;
DROP TABLE Ref_Order_Item_Status_Codes ;
DROP TABLE Shipments ;
DROP TABLE Shipment_Items ;
DROP TABLE Order_Items;
DROP TABLE Products;
DROP TABLE Customers;
DROP TABLE Ref_Payment_Methods;
DROP TABLE Customer_Payment_Methods;
DROP TABLE Ref_Invoice_Status_Codes;
DROP TABLE Payments;
DROP TABLE Ref_Product_Types ;
-- ----------------------
-- CREATE TABLES
-- ----------------------
CREATE TABLE Invoices (
   invoice_number      INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
(START WITH 1, INCREMENT BY 1), -- KA: Replaced "IDENTITY(1,1)" with
GENERATED ...
   order_id            INTEGER NOT NULL,
   invoice_status_code CHAR(10) NOT NULL,
   invoice_date        DATE,  -- KA: was DATETIME
   invoice_details     VARCHAR(255),
   CONSTRAINT PK_Invoices PRIMARY KEY (invoice_number)
);
CREATE TABLE Orders (
   order_id         INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
(START WITH 1, INCREMENT BY 1), -- KA: Replaced "IDENTITY(1,1)" with
GENERATED ...
```

```
   customer_id       INTEGER NOT NULL,
   order_status_code CHAR(10) NOT NULL,
   date_order_placed DATE NOT NULL,  -- KA: was DATETIME
   order_details     VARCHAR(255),
   CONSTRAINT PK_Orders PRIMARY KEY (order_id)
);
CREATE TABLE Ref_Order_Status_Codes (
   order_status_code        CHAR(10) NOT NULL,
   order_status_description VARCHAR(80),  -- eg. Cancelled, Completed
   CONSTRAINT PK_Ref_Order_Status_Codes PRIMARY KEY (order_status_code)
);
CREATE TABLE Ref_Order_Item_Status_Codes (
   order_item_status_code        CHAR(10) NOT NULL,
   order_item_status_description VARCHAR(80),  -- eg Delivered, Out of Stock
   CONSTRAINT PK_Ref_Order_Item_Status_Codes PRIMARY KEY
(order_item_status_code)
);
CREATE TABLE Shipments (
   shipment_id             INTEGER NOT NULL GENERATED ALWAYS AS
IDENTITY (START WITH 1, INCREMENT BY 1), -- KA: Replaced "IDENTITY(1,1)"
with GENERATED ...
   order_id                INTEGER NOT NULL,
   invoice_number          INTEGER NOT NULL,
   shipment_tracking_number VARCHAR(80),
   shipment_date           DATE, -- KA: was DATETIME
   other_shipment_details   VARCHAR(255),
   CONSTRAINT PK_Shipments PRIMARY KEY (shipment_id)
);
CREATE TABLE Shipment_Items (
   shipment_id   INTEGER NOT NULL,
   order_item_id INTEGER NOT NULL,
   CONSTRAINT PK_Shipment_Items PRIMARY KEY (shipment_id, order_item_id)
);
CREATE TABLE Order_Items (
   order_item_id           INTEGER NOT NULL GENERATED ALWAYS AS
IDENTITY (START WITH 1, INCREMENT BY 1), -- KA: Replaced "IDENTITY(1,1)"
with GENERATED ...
   product_id              INTEGER NOT NULL,
   order_id                INTEGER NOT NULL,
   order_item_status_code   CHAR(10) NOT NULL,
   order_item_quantity      VARCHAR(50),
   order_item_price         DECIMAL,  -- KA: was MONEY
   other_order_item_details VARCHAR(255),
   CONSTRAINT PK_Order_Items PRIMARY KEY (order_item_id)
```

```
);
CREATE TABLE Products (
   product_id           INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
(START WITH 1, INCREMENT BY 1), -- KA: Replaced "IDENTITY(1,1)" with
GENERATED ...
   product_type_code     CHAR(15) NOT NULL,
   product_name          VARCHAR(80),
   product_price         DECIMAL,  -- KA: was MONEY,
   product_color         VARCHAR(20),
   product_size          VARCHAR(20),
   product_description   VARCHAR(255),
   other_product_details VARCHAR(255),
   CONSTRAINT PK_Products PRIMARY KEY (product_id)
);
CREATE TABLE Customers (
   customer_id           INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
(START WITH 1, INCREMENT BY 1), -- KA: Replaced "IDENTITY(1,1)" with
GENERATED ...
   organisation_or_person CHAR(20),
   organisation_name      VARCHAR(40),
   gender                CHAR(1),
   first_name            VARCHAR(50),
   middle_initial        CHAR(1),
   last_name             VARCHAR(50),
   email_address         VARCHAR(255),
   login_name            VARCHAR(80),
   login_password        VARCHAR(20),
   phone_number          VARCHAR(255),
   address_line_1        VARCHAR(255),
   address_line_2        VARCHAR(255),
   address_line_3        VARCHAR(255),
   address_line_4        VARCHAR(80),
   town_city             VARCHAR(50),
   county                VARCHAR(50),
   country               VARCHAR(50),
   CONSTRAINT PK_Customers PRIMARY KEY (customer_id)
);
CREATE TABLE Ref_Payment_Methods (
   payment_method_code       CHAR(10) NOT NULL,
   payment_method_description VARCHAR(80),  -- eg CC=Credit Card.
   CONSTRAINT PK_Ref_Payment_Methods PRIMARY KEY
(payment_method_code)
);
CREATE TABLE Customer_Payment_Methods (
```

```
   customer_payment_id    INTEGER NOT NULL GENERATED ALWAYS AS
IDENTITY (START WITH 1, INCREMENT BY 1), -- KA: Replaced "IDENTITY(1,1)"
with GENERATED ...
   customer_id            INTEGER NOT NULL,
   payment_method_code    CHAR(10) NOT NULL,
   credit_card_number     VARCHAR(40),
   payment_method_details CHAR(50) NULL,
   CONSTRAINT PK_Customer_Payment_Methods PRIMARY KEY
(customer_payment_id)
);
CREATE TABLE Ref_Invoice_Status_Codes (
   invoice_status_code        CHAR(10) NOT NULL,
   invoice_status_description VARCHAR(80), -- eg Issued, Paid.
   CONSTRAINT PK_Ref_Invoice_Status_Codes PRIMARY KEY
(invoice_status_code)
);
CREATE TABLE Payments (
   payment_id      INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY
(START WITH 1, INCREMENT BY 1), -- KA: Replaced "IDENTITY(1,1)" with
GENERATED ...
   invoice_number  INTEGER NOT NULL,
   payment_date    DATE,    -- KA: Was DATETIME,
   payment_amount  DECIMAL, -- KA: Was MONEY,
   CONSTRAINT PK_Payments PRIMARY KEY (payment_id)
);
CREATE TABLE Ref_Product_Types (
   product_type_code        CHAR(15) NOT NULL,
   parent_product_type_code CHAR(15),
   product_type_description VARCHAR(80), --e.g. Book, CD
   CONSTRAINT PK_Ref_Product_Types PRIMARY KEY (product_type_code)
);
-----------------------------------------------------------------------
-- Foreign key constraints
-----------------------------------------------------------------------
ALTER TABLE Invoices ADD CONSTRAINT Invoice_Status_Codes_Invoices
   FOREIGN KEY (invoice_status_code) REFERENCES Ref_Invoice_Status_Codes
(invoice_status_code);
ALTER TABLE Invoices ADD CONSTRAINT Orders_Invoices
   FOREIGN KEY (order_id) REFERENCES Orders (order_id);
ALTER TABLE Orders ADD CONSTRAINT Order_Status_Codes_Orders
   FOREIGN KEY (order_status_code) REFERENCES Ref_Order_Status_Codes
(order_status_code);
ALTER TABLE Orders ADD CONSTRAINT Customers_1_Orders
   FOREIGN KEY (customer_id) REFERENCES Customers (customer_id);
```

```
ALTER TABLE Shipments ADD CONSTRAINT Orders_Shipments
   FOREIGN KEY (order_id) REFERENCES Orders (order_id);
ALTER TABLE Shipments ADD CONSTRAINT Invoices_Shipments
   FOREIGN KEY (invoice_number) REFERENCES Invoices (invoice_number);
ALTER TABLE Shipment_Items ADD CONSTRAINT Shipments_Shipment_Items
   FOREIGN KEY (shipment_id) REFERENCES Shipments (shipment_id);
ALTER TABLE Shipment_Items ADD CONSTRAINT Order_Items_Shipment_Items
   FOREIGN KEY (order_item_id) REFERENCES Order_Items (order_item_id);
ALTER TABLE Order_Items ADD CONSTRAINT Order_Item_Status_Order_Items
   FOREIGN KEY (order_item_status_code) REFERENCES
Ref_Order_Item_Status_Codes (order_item_status_code);
ALTER TABLE Order_Items ADD CONSTRAINT Products_Order_Items
   FOREIGN KEY (product_id) REFERENCES Products (product_id);
ALTER TABLE Order_Items ADD CONSTRAINT Orders_Order_Items
   FOREIGN KEY (order_id) REFERENCES Orders (order_id);
ALTER TABLE Products ADD CONSTRAINT Ref_Product_Types_Products
   FOREIGN KEY (product_type_code) REFERENCES Ref_Product_Types
(product_type_code);
ALTER TABLE Customer_Payment_Methods ADD CONSTRAINT
Customers_Customer_Payment_Methods
   FOREIGN KEY (customer_id) REFERENCES Customers (customer_id);
ALTER TABLE Customer_Payment_Methods ADD CONSTRAINT
Ref_Payment_Methods_Customer_Payment_Methods
   FOREIGN KEY (payment_method_code) REFERENCES Ref_Payment_Methods
(payment_method_code);
ALTER TABLE Payments ADD CONSTRAINT Invoices_Payments
   FOREIGN KEY (invoice_number) REFERENCES Invoices (invoice_number);
ALTER TABLE Ref_Product_Types ADD CONSTRAINT
Ref_Product_Types_Ref_Product_Types
   FOREIGN KEY (parent_product_type_code) REFERENCES Ref_Product_Types
(product_type_code);
```

2.2. DML Script for MS SQL Server Sample RDB Instance:

```
SET SCHEMA ECOMM;

----------------------------------------
-- Clean up
----------------------------------------
ALTER TABLE Invoices ALTER FOREIGN KEY Invoice_Status_Codes_Invoices NOT
ENFORCED;
ALTER TABLE Invoices ALTER FOREIGN KEY Orders_Invoices NOT ENFORCED;
ALTER TABLE Orders ALTER FOREIGN KEY Order_Status_Codes_Orders NOT
ENFORCED;
```

```
ALTER TABLE Orders ALTER FOREIGN KEY Customers_1_Orders NOT ENFORCED;
ALTER TABLE Shipments ALTER FOREIGN KEY Orders_Shipments NOT ENFORCED;
ALTER TABLE Shipments ALTER FOREIGN KEY Invoices_Shipments NOT ENFORCED;
ALTER TABLE Shipment_Items ALTER FOREIGN KEY Shipments_Shipment_Items NOT
ENFORCED;
ALTER TABLE Shipment_Items ALTER FOREIGN KEY Order_Items_Shipment_Items NOT
ENFORCED;
ALTER TABLE Order_Items ALTER FOREIGN KEY Order_Item_Status_Order_Items NOT
ENFORCED;
ALTER TABLE Order_Items ALTER FOREIGN KEY Products_Order_Items NOT
ENFORCED;
ALTER TABLE Order_Items ALTER FOREIGN KEY Orders_Order_Items NOT ENFORCED;
ALTER TABLE Products ALTER FOREIGN KEY Ref_Product_Types_Products NOT
ENFORCED;
ALTER TABLE Customer_Payment_Methods ALTER FOREIGN KEY
Customers_Customer_Payment_Methods NOT ENFORCED;
ALTER TABLE Customer_Payment_Methods ALTER FOREIGN KEY
Ref_Payment_Methods_Customer_Payment_Methods NOT ENFORCED;
ALTER TABLE Payments ALTER FOREIGN KEY Invoices_Payments NOT ENFORCED;
ALTER TABLE Ref_Product_Types ALTER FOREIGN KEY
Ref_Product_Types_Ref_Product_Types NOT ENFORCED;

DELETE FROM Invoices;
DELETE FROM Orders;
DELETE FROM Ref_Order_Status_Codes ;
DELETE FROM Ref_Order_Item_Status_Codes ;
DELETE FROM Shipments ;
DELETE FROM Shipment_Items ;
DELETE FROM Order_Items;
DELETE FROM Products;
DELETE FROM Customers;
DELETE FROM Ref_Payment_Methods;
DELETE FROM Customer_Payment_Methods;
DELETE FROM Ref_Invoice_Status_Codes;
DELETE FROM Payments;
DELETE FROM Ref_Product_Types ;

-- Restart IDENTITY Cols
ALTER TABLE Invoices ALTER COLUMN invoice_number RESTART with 1;
ALTER TABLE Orders ALTER COLUMN order_id RESTART with 1;
ALTER TABLE Shipments ALTER COLUMN shipment_id RESTART with 1;
ALTER TABLE Order_Items ALTER COLUMN order_item_id RESTART with 1;
ALTER TABLE Products ALTER COLUMN product_id RESTART with 1;
ALTER TABLE Customers ALTER COLUMN customer_id RESTART with 1;
ALTER TABLE Customer_Payment_Methods ALTER COLUMN customer_payment_id
RESTART with 1;
ALTER TABLE Payments ALTER COLUMN payment_id RESTART with 1;
----------------------------------------
```

```
-- Insert data in tables
----------------------------------------
INSERT INTO Ref_Invoice_Status_Codes
 (invoice_status_code,invoice_status_description) VALUES
 ('Issued'          ,'Issued')
,('Paid'            ,'Paid')
--
,('Pre-Issue'       ,'Pre-Issue State')
,('Sent1'           ,'Sent for the 1st time')
,('Sent2'           ,'Sent for the 2nd time - 2nd notice')
,('Sent3'           ,'Sent for the 3ed time - 3ed and final notice')
,('Contested'       ,'Contensted by the client')
,('Under-Rev'       ,'Under-Review after being contested')
,('Reviewed'        ,'Review Completed')
,('Re-issued'       ,'Re-issued after review')
,('Re-sent1'        ,'Re-sent for the 1st time')
,('Re-sent2'        ,'Re-sent for the 2nd time - 2nd notice')
,('Re-sent3'        ,'Re-sent for the 3ed time - 3ed and final notice')
,('Collection'      ,'Went to collection')
,('Collected'       ,'Payment collected through collection')
,('Canceled'        ,'Invoce Canceled')
,('OTHER'           ,'None of the above.')
;
INSERT INTO Ref_Order_Item_Status_Codes
 (order_Item_status_code,order_Item_status_description) VALUES
 ('DEL'             ,'Delivered')
,('OUT'             ,'Out of Stock')
,('ROUTE'           ,'En Route')
,('WAIT'            ,'Waiting')
--
,('READY'           ,'Ready to be shipped')
,('BACK-ORDER'      ,'Back-Order')
,('RETURNED'        ,'Retruned')
,('LOST'            ,'Declared Lost')
,('UNKNOWN'         ,'Currently Unknown')
,('CANCELED'        ,'Cancelded')
;
INSERT INTO Ref_Order_Status_Codes
 (order_status_code,order_status_description) VALUES
 ('CANC'          ,'Cancelled')
,('COMPL'         ,'Completed')
,('OPEN'          ,'Open - eg just placed')
,('PROV'          ,'Provisional')
;
INSERT INTO Ref_Payment_Methods
 (payment_method_code,payment_method_description) VALUES
 ('AMEX'          ,'American Express')
,('CASH'          ,'Cash')
```

```
,('DD'              ,'Direct Debit')
--
,('VISA'            ,'Visa')
,('MC'              ,'Master-Card')
,('DISC'            ,'Discover')
,('DINERS'          ,'Diners Club Card')
,('CHECK'           ,'Check')
,('MONEY-ORD'       ,'Money Order')
;
INSERT INTO Ref_Product_Types
 (product_type_code,parent_product_type_code,product_type_description) VALUES
 ('Book'          ,NULL                  ,'Book'                )
,('Camera'        ,NULL                  ,'Camera'              )
,('Digital Camera' ,'Camera'             ,'Digital Camera'      )
,('CD'            ,NULL                  ,'CD'                  )
--
,('EBook'         ,'Book'                ,'Electronic Book'     )
,('Tape'          ,NULL                  ,'TAPE'                )
,('DVD'           ,NULL                  ,'DVD'                 )
,('Phone'         ,NULL                  ,'Phone'               )
,('Cell-Phone'    ,'Phone'               ,'Cell Phone'          )
,('Smart-Phone'   ,'Phone'               ,'Smart Phone'         )
,('PDA'           ,NULL                  ,'PDA'                 )
,('Computer'      ,NULL                  ,'Computer'            )
,('Notebook'      ,'Computer'            ,'Notebook'            )
,('Netbook'       ,'Computer'            ,'Netbook'             )
,('Desktop'       ,'Computer'            ,'Desktop'             )
,('Server'        ,'Computer'            ,'Server'              )
,('Storage'       ,NULL                  ,'Storage Device'      )
,('IHD'           ,'Storage'             ,'Internal Hard Disk'  )
,('EHD'           ,'Storage'             ,'External Hard Disk'  )
,('FLASH'         ,'Storage'             ,'USB Flash Drive'     )
,('SAN'           ,'Storage'             ,'SAN Box'             )
,('Toy'           ,NULL                  ,'Toys Category'       )
,('Toy1'          ,'Toy'                 ,'Toy 001'             )
,('Toy2'          ,'Toy'                 ,'Toy 002'             )
,('Toy3'          ,'Toy'                 ,'Toy 003'             )
,('Toy4'          ,'Toy'                 ,'Toy 004'             )
,('Toy5'          ,'Toy'                 ,'Toy 005'             )
,('Toy6'          ,'Toy'                 ,'Toy 006'             )
;
INSERT INTO Products
(product_type_code,product_name
,product_price,product_color,product_size,product_description          ,other_product_details)
VALUES
 ('Digital Camera' ,'Olympus Camedia C-170' ,64.97       ,'Silver'    ,NULL       ,'Olympus C-170
Digital Camera'  ,NULL    )
```

```
,('Digital Camera' ,'Pentax Opto 50L'       ,89.97       ,'Rose'       ,NULL       ,'Pentax Opto 50L
Digital Camera',NULL    )
,('Digital Camera' ,'Nikon Coolpix L3 Black',99.99       ,'Black'      ,NULL       ,'Nikon Coolpix
L3 Black'       ,NULL    )
--
,('Book'          ,'Book Name1'          ,09.99       ,'N/A'       ,'50 Pages'  ,'Book Name1 Desc.'
,NULL    )
,('Book'          ,'Book Name2'          ,19.99       ,'N/A'       ,'100 Pages' ,'Book Name2 Desc.'
,NULL    )
,('Book'          ,'Book Name3'          ,29.99       ,'N/A'       ,'150 Pages' ,'Book Name3 Desc.'
,NULL    )
,('EBook'         ,'EBook Name1'         ,09.99       ,'N/A'       ,'50 MB'     ,'Book Name1 Desc.'
,NULL    )
,('EBook'         ,'EBook Name2'         ,19.99       ,'N/A'       ,'100 MB'    ,'Book Name2 Desc.'
,NULL    )
,('EBook'         ,'EBook Name3'         ,29.99       ,'N/A'       ,'150 MB'    ,'Book Name3 Desc.'
,NULL    )
,('Tape'          ,'Tape Name1'          ,09.99       ,'N/A'       ,'90 Min'    ,'Tape Name1 Desc.'
,NULL    )
,('DVD'           ,'DVD Name1'           ,19.99       ,'N/A'       ,'90 Min'    ,'DVD Name1 Desc.'
,NULL    )
,('Phone'         ,'Phone Name1'         ,19.99       ,'White'     ,'10x6 inches','Phone Name1 Desc.'
,NULL    )
,('Phone'         ,'Phone Name2'         ,29.99       ,'Black'     ,'7x7 inches','Phone Name2 Desc.'
,NULL    )
,('Cell-Phone'    ,'Cell Phone Name1'    ,99.99       ,'White'     ,'Slim'      ,'Cell Phone Name1
Desc.'       ,NULL    )
,('Cell-Phone'    ,'Cell Phone Name2'    ,99.99       ,'Silver'    ,'Slim'      ,'Cell Phone Name2
Desc.'       ,NULL    )
,('Cell-Phone'    ,'Cell Phone Name3'    ,99.99       ,'Black'     ,'Slim'      ,'Cell Phone Name3
Desc.'       ,NULL    )
,('Smart-Phone'   ,'Smart Phone Name1'   ,199.99      ,'White'     ,'Slim'      ,'Smart Phone
Name1 Desc.'      ,NULL    )
,('Smart-Phone'   ,'Smart Phone Name2'   ,299.99      ,'Silver'    ,'Slim'      ,'Smart Phone Name2
Desc.'      ,NULL    )
,('Smart-Phone'   ,'Smart Phone Name3'   ,399.99      ,'Black'     ,'Slim'      ,'Smart Phone Name3
Desc.'      ,NULL    )
,('PDA'           ,'PDA Name1'           ,99.99       ,'Black'     ,'Slim'      ,'PDA Name1 Desc.'
,NULL    )
,('Notebook'      ,'Notebook Name1'      ,399.99      ,'White'     ,'Slim'      ,'Notebook Name1
Desc.'       ,NULL    )
,('Notebook'      ,'Notebook Name2'      ,499.99      ,'Black'     ,'Slim'      ,'Notebook Name2
Desc.'       ,NULL    )
,('Netbook'       ,'Netbook Name1'       ,399.99      ,'White'     ,'Slim'      ,'Notebook Name1
Desc.'       ,NULL    )
,('Netbook'       ,'Netbook Name2'       ,499.99      ,'Black'     ,'Slim'      ,'Notebook Name2 Desc.'
,NULL    )
```

```
,('Desktop'       ,'Desktop Name1'        ,399.99      ,'White'     ,'Slim'     ,'Desktop Name1 Desc.'
,NULL    )
,('Desktop'       ,'Desktop Name2'        ,499.99      ,'Black'     ,'Slim'     ,'Desktop Name2 Desc.'
,NULL    )
,('IHD'           ,'Internal HD Name1'    ,99.99       ,'N/A'       ,'1 TB'     ,'Internal HD Name1
Desc.'      ,NULL    )
,('EHD'           ,'External HD Name1'    ,99.99       ,'N/A'       ,'1 TB'     ,'External HD Name1
Desc.'      ,NULL    )
,('FLASH'         ,'FLASH Name1'          ,49.99       ,'Red'       ,'10 GB'    ,'Flash Name1 Desc.'
,NULL    )
;
INSERT INTO Customers
 --(first_name, middle_initial, last_name, email_address                    , address_line_1            ,
address_line_2            , address_line_3, address_line_4, town_city, county, country) VALUES
 --('John'    , NULL         , 'Doe'    , 'john.doe@fictitiousmail.com'     , '1500 E MAIN AVE STE
201'  , 'SPRINGFIELD VA 22162-1010', NULL          , NULL          , NULL     , NULL  , NULL
)
 --('Joe '    , NULL         , 'Bloggs' , 'joe.bloggs@fictitiousmail.com'    , '1776 New Cavendish
Street', 'Marylebone'              , NULL          , 'W11X 5BY'     , 'London'     , 'Greater London',
'UK')
 (organisation_or_person, organisation_name, gender, first_name, middle_initial, last_name ,
email_address                , login_name, login_password, address_line_1            , address_line_2
, address_line_3, address_line_4, town_city, county, country) VALUES
 ('Person'            , NULL            , 'M'   , 'John'    , NULL          , 'Doe'    ,
'john.doe@fictitiousmail.com'  , 'user01'  , 'john_pwd'    , '1500 E MAIN AVE STE 201'  , ' VA
22162-1010'          , NULL          , NULL          , 'SPRING' , NULL  , 'USA')
,('Person'            , NULL            , 'M'   , 'Joe '    , NULL          , 'Bloggs' ,
'joe.bloggs@fictitiousmail.com', 'user02'  , 'joe_pwd'     , '1776 New Cavendish Street',
'Marylebone'              , NULL          , 'W11X 5BY'    , 'London' , 'G.L.', 'UK')
,('Person'            , NULL            , 'M'   , 'F-Name1' , 'A'           , 'L-Name-A',
'fname1@fictitiousmail.com'   , 'user03'  , 'fname01_pwd' , 'F-Name01 address_line1' , 'F-Name1
address_line2'     , NULL          , NULL          , 'City1'  , NULL  , 'USA'  )
,('Person'            , NULL            , 'F'   , 'F-Name2' , 'B'           , 'L-Name-B',
'fname2@fictitiousmail.com'   , 'user04'  , 'fname02_pwd' , 'F-Name02 address_line1' , 'F-Name2
address_line2'     , NULL          , NULL          , 'City2'  , NULL  , 'USA'  )
,('Person'            , NULL            , 'M'   , 'F-Name3' , 'C'           , 'L-Name-C',
'fname3@fictitiousmail.com'   , 'user05'  , 'fname03_pwd' , 'F-Name03 address_line1' , 'F-Name3
address_line2'     , NULL          , NULL          , 'City3'  , NULL  , 'USA'  )
,('Person'            , NULL            , 'F'   , 'F-Name4' , 'D'           , 'L-Name-D',
'fname4@fictitiousmail.com'   , 'user06'  , 'fname04_pwd' , 'F-Name04 address_line1' , 'F-Name4
address_line2'     , NULL          , NULL          , 'City4'  , NULL  , 'USA'  )
,('Person'            , NULL            , 'M'   , 'F-Name5' , 'E'           , 'L-Name-E',
'fname5@fictitiousmail.com'   , 'user07'  , 'fname05_pwd' , 'F-Name05 address_line1' , 'F-Name5
address_line2'     , NULL          , NULL          , 'City5'  , NULL  , 'USA'  )
,('Person'            , NULL            , 'F'   , 'F-Name6' , 'F'           , 'L-Name-F',
'fname6@fictitiousmail.com'   , 'user08'  , 'fname06_pwd' , 'F-Name06 address_line1' , 'F-Name6
address_line2'     , NULL          , NULL          , 'City6'  , NULL  , 'USA'  )
```

```
,('Person'          , NULL          , 'M'   , 'F-Name7' , 'G'          , 'L-Name-G',
'fname7@fictitiousmail.com'   , 'user09'  , 'fname07_pwd' , 'F-Name07 address_line1' , 'F-Name7
address_line2'     , NULL         , NULL         , 'City7'  , NULL  , 'USA'  )
,('Person'          , NULL          , 'F'   , 'F-Name8' , 'H'          , 'L-Name=H',
'fname8@fictitiousmail.com'   , 'user10'  , 'fname08_pwd' , 'F-Name08 address_line1' , 'F-Name8
address_line2'     , NULL         , NULL         , 'City8'  , NULL  , 'USA'  )
,('Person'          , NULL          , 'M'   , 'F-Name9' , 'I'          , 'L-Name-I',
'fname9@fictitiousmail.com'   , 'user11'  , 'fname09_pwd' , 'F-Name09 address_line1' , 'F-Name9
address_line2'     , NULL         , NULL         , 'City9'  , NULL  , 'USA'  )
,('Person'          , NULL          , 'F'   , 'F-Name10', 'J'          , 'L-Name-J',
'fname10@fictitiousmail.com'  , 'user12'  , 'fname10_pwd' , 'F-Name10 address_line1' , 'F-
Name10 address_line2'    , NULL         , NULL         , 'City10' , NULL  , 'USA'  )
,('Person'          , NULL          , 'M'   , 'F-Name10', 'A'          , 'L-Name-A',
'fname11@fictitiousmail.com'  , 'user13'  , 'fname10_pwd' , 'F-Name10 address_line1' , 'F-
Name10 address_line2'    , NULL         , NULL         , 'City11' , NULL  , 'USA'  )
,('Person'          , NULL          , 'F'   , 'F-Name9' , 'B'          , 'L-Name-B',
'fname12@fictitiousmail.com'  , 'user14'  , 'fname09_pwd' , 'F-Name09 address_line1' , 'F-
Name09 address_line2'    , NULL         , NULL         , 'City12' , NULL  , 'USA'  )
,('Person'          , NULL          , 'F'   , 'F-Name8' , 'C'          , 'L-Name-C',
'fname13@fictitiousmail.com'  , 'user15'  , 'fname09_pwd' , 'F-Name08 address_line1' , 'F-
Name08 address_line2'    , NULL         , NULL         , 'City13' , NULL  , 'USA'  )
,('Person'          , NULL          , 'F'   , 'F-Name7' , 'D'          , 'L-Name-D',
'fname14@fictitiousmail.com'  , 'user16'  , 'fname07_pwd' , 'F-Name07 address_line1' , 'F-
Name07 address_line2'    , NULL         , NULL         , 'City14' , NULL  , 'USA'  )
,('Person'          , NULL          , 'F'   , 'F-Name6' , 'E'          , 'L-Name-E',
'fname15@fictitiousmail.com'  , 'user17'  , 'fname06_pwd' , 'F-Name06 address_line1' , 'F-
Name06 address_line2'    , NULL         , NULL         , 'City15' , NULL  , 'USA'  )
,('Person'          , NULL          , 'F'   , 'F-Name5' , 'F'          , 'L-Name-F',
'fname16@fictitiousmail.com'  , 'user18'  , 'fname05_pwd' , 'F-Name05 address_line1' , 'F-
Name05 address_line2'    , NULL         , NULL         , 'City16' , NULL  , 'USA'  )
,('Person'          , NULL          , 'F'   , 'F-Name4' , 'G'          , 'L-Name-G',
'fname17@fictitiousmail.com'  , 'user19'  , 'fname04_pwd' , 'F-Name04 address_line1' , 'F-
Name04 address_line2'    , NULL         , NULL         , 'City17' , NULL  , 'USA'  )
,('Person'          , NULL          , 'F'   , 'F-Name3' , 'H'          , 'L-Name-H',
'fname18@fictitiousmail.com'  , 'user20'  , 'fname03_pwd' , 'F-Name03 address_line1' , 'F-
Name03 address_line2'    , NULL         , NULL         , 'City18' , NULL  , 'USA'  )
,('Person'          , NULL          , 'F'   , 'F-Name3' , 'I'          , 'L-Name-I',
'fname19@fictitiousmail.com'  , 'user21'  , 'fname02_pwd' , 'F-Name02 address_line1' , 'F-
Name02 address_line2'    , NULL         , NULL         , 'City19' , NULL  , 'USA'  )
,('Person'          , NULL          , 'F'   , 'F-Name1' , 'J'          , 'L-Name-J',
'fname20@fictitiousmail.com'  , 'user22'  , 'fname01_pwd' , 'F-Name01 address_line1' , 'F-
Name01 address_line2'    , NULL         , NULL         , 'City20' , NULL  , 'USA'  )
,('Org.'            , 'Org Name1'     , NULL  , NULL      , NULL         , NULL      ,
'sales1@fictitiousmail.com'   , 'user23'  , 'org01_pwd'   , 'Org Name1 address_line1', 'Org Name1
address_line2'   , NULL         , NULL         , 'City1'  , NULL  , 'USA'  )
,('Org.'            , 'Org Name2'     , NULL  , NULL      , NULL         , NULL      ,
'sales2@fictitiousmail.com'   , 'user24'  , 'org02_pwd'   , 'Org Name2 address_line1', 'Org Name2
address_line2'   , NULL         , NULL         , 'City2'  , NULL  , 'USA'  )
```

```
;
INSERT INTO Customer_Payment_Methods
 (customer_id,payment_method_code,credit_card_number,payment_method_details        )
VALUES
 (1        ,'AMEX'           ,'123456'         ,'From 01/01/2004 to 01/01/2008')
,(2        ,'CASH'           ,NULL             ,NULL                )
--
,(1        ,'DD'             ,NULL             ,'From 01/01/2008 to 01/01/2011')
,(2        ,'AMEX'           ,'0123'           ,'From 01/01/2009 to 01/01/2011')
,(3        ,'CASH'           ,NULL             ,NULL)
,(4        ,'DD'             ,NULL             ,'From 01/01/2010 to 01/01/2011')
,(5        ,'VISA'           ,'1234'           ,NULL)
,(6        ,'MC'             ,'2345'           ,'From 01/01/2010 to 01/01/2011')
,(7        ,'DISC'           ,'3456'           ,'From 01/07/2009 to 01/01/2011')
,(8        ,'DINERS'         ,'4567'           ,'From 01/07/2009 to 01/01/2011')
,(9        ,'CHECK'          ,NULL             ,NULL)
,(10       ,'MONEY-ORD'      ,NULL             ,NULL)
,(11       ,'DD'             ,NULL             ,'From 01/07/2008 to 01/01/2011')
,(12       ,'AMEX'           ,'0124'           ,'From 01/07/2008 to 01/01/2011')
,(13       ,'CASH'           ,NULL             ,NULL)
,(14       ,'DD'             ,NULL             ,'From 01/07/2010 to 01/01/2011')
,(15       ,'VISA'           ,'1235'           ,NULL)
,(16       ,'MC'             ,'2346'           ,'From 01/01/2010 to 01/01/2011')
,(17       ,'DISC'           ,'3457'           ,'From 01/01/2009 to 01/01/2010')
,(18       ,'DINERS'         ,'4568'           ,'From 01/07/2009 to 01/01/2010')
,(19       ,'CHECK'          ,NULL             ,NULL)
,(20       ,'MONEY-ORD'      ,NULL             ,NULL)
;
INSERT INTO Orders
 (customer_id,order_status_code,date_order_placed,order_details                ) VALUES
 (1        ,'OPEN'           ,'01/01/2007'     ,'First Order from a new Customer')
 --
 ,(1       ,'CANC'           ,'01/01/2007'     ,'Duplicate Order from a new Customer')
 ,(2       ,'COMPL'          ,'01/01/2006'     ,'Completed Order from a new Customer')
 ,(2       ,'OPEN'           ,'01/01/2007'     ,'Order from an existing Customer')
 ,(3       ,'OPEN'           ,'01/01/2007'     ,'First Order from a new Customer')
 ,(4       ,'OPEN'           ,'01/01/2007'     ,'First Order from a new Customer')
 ,(5       ,'OPEN'           ,'01/01/2007'     ,'First Order from a new Customer')
 ,(6       ,'OPEN'           ,'01/01/2007'     ,'First Order from a new Customer')
 ,(7       ,'OPEN'           ,'01/01/2007'     ,'First Order from a new Customer')
 ,(8       ,'OPEN'           ,'01/01/2007'     ,'First Order from a new Customer')
 ,(9       ,'OPEN'           ,'01/01/2007'     ,'First Order from a new Customer')
 ,(10      ,'OPEN'           ,'01/01/2007'     ,'First Order from a new Customer')
 ,(11      ,'CANC'           ,'01/06/2006'     ,'Canceled Order from a new Customer')
 ,(12      ,'CANC'           ,'01/07/2006'     ,'Canceled Order from an existing Customer')
 ,(13      ,'COMPL'          ,'01/08/2006'     ,'Completed Order from a new Customer')
 ,(14      ,'PROV'           ,'12/31/2006'     ,'Completed Order from a new Customer')
 ,(15      ,'PROV'           ,'12/31/2006'     ,'Prov. Order from an existing Customer')
```

```
 ,(16      ,'PROV'       ,'12/31/2006'   ,'Prov. Order from an existing Customer')
 ,(17      ,'PROV'       ,'12/31/2006'   ,'Prov. Order from an existing Customer')
 ,(18      ,'PROV'       ,'12/31/2006'   ,'Prov. Order from an existing Customer')
 ,(19      ,'PROV'       ,'12/31/2006'   ,'Prov. Order from an existing Customer')
 ,(20      ,'PROV'       ,'12/31/2006'   ,'Prov. Order from an existing Customer')
 ,(21      ,'OPEN'       ,'01/01/2007'   ,'Order from an existing Customer')
 ,(22      ,'OPEN'       ,'01/01/2007'   ,'Order from an existing Customer')
 ,(23      ,'OPEN'       ,'01/01/2007'   ,'Order from an existing Customer')
 ,(24      ,'CANC'       ,'01/01/2006'   ,'Canceled Order from an existing Customer')
 ;
INSERT INTO Order_Items
 (product_id,order_id,
order_item_status_code,order_item_quantity,order_item_price,other_order_item_details)
VALUES
 (1       ,1      ,'DEL'             ,1              ,100.00       ,NULL                )
,(2       ,1      ,'ROUTE'           ,2              ,200.00       ,'A Rare Groove'     )
,(3       ,1      ,'WAIT'            ,3              ,300.00       ,'The usual Order'   )
--
,(1       ,2      ,'CANCELED'        ,1              ,100.00       ,NULL                )
,(2       ,2      ,'CANCELED'        ,2              ,200.00       ,'A Rare Groove'     )
,(3       ,2      ,'CANCELED'        ,3              ,300.00       ,'The usual Order'   )
,(4       ,3      ,'DEL'             ,5              ,100.00       ,'Something 1'       )
,(5       ,3      ,'DEL'             ,5              ,200.00       ,'Something 2'       )
,(6       ,4      ,'WAIT'            ,2              ,300.00       ,'Something 1 again' )
,(7       ,4      ,'READY'           ,2              ,200.00       ,'Something 2 again' )
,(8       ,5      ,'OUT'             ,4              ,500.00       ,'Something 1'       )
,(9       ,6      ,'BACK-ORDER'      ,6              ,600.00       ,'Something 1'       )
,(10      ,6      ,'RETURNED'        ,8              ,800.00       ,'Something 2'       )
,(11      ,7      ,'LOST'            ,4              ,400.00       ,'Something 1'       )
,(12      ,8      ,'UNKNOWN'         ,1              ,100.00       ,'Something 1'       )
,(13      ,9      ,'ROUTE'           ,1              ,100.00       ,'Something 1'       )
,(14      ,10     ,'ROUTE'           ,2              ,100.00       ,'Something 1'       )
,(15      ,11     ,'ROUTE'           ,2              ,100.00       ,'Something 1'       )
,(16      ,12     ,'ROUTE'           ,2              ,100.00       ,'Something 1'       )
,(17      ,13     ,'READY'           ,1              ,100.00       ,'Something 1'       )
,(18      ,14     ,'READY'           ,1              ,100.00       ,'Something 1'       )
,(19      ,15     ,'DEL'             ,1              ,100.00       ,'Something 1'       )
,(20      ,16     ,'DEL'             ,1              ,100.00       ,'Something 1'       )
,(21      ,16     ,'DEL'             ,1              ,100.00       ,'Something 2'       )
,(20      ,17     ,'RETURNED'        ,8              ,300.00       ,'Something 1'       )
,(21      ,18     ,'LOST'            ,4              ,200.00       ,'Something 1'       )
,(22      ,19     ,'UNKNOWN'         ,1              ,100.00       ,'Something 1'       )
,(23      ,20     ,'ROUTE'           ,1              ,100.00       ,'Something 1'       )
,(24      ,21     ,'ROUTE'           ,2              ,100.00       ,'Something 1'       )
,(25      ,22     ,'ROUTE'           ,2              ,100.00       ,'Something 1'       )
,(26      ,23     ,'ROUTE'           ,2              ,100.00       ,'Something 1'       )
,(27      ,24     ,'ROUTE'           ,1              ,100.00       ,'Something 1'       )
,(28      ,25     ,'ROUTE'           ,1              ,100.00       ,'Something 1'       )
```

```
,(28      ,26     ,'CANCELED'         ,1              ,100.00         ,'Something 1'          )
,(29      ,26     ,'CANCELED'         ,1              ,100.00         ,'Something 2'          )
;
INSERT INTO Invoices
  (order_id, invoice_status_code, invoice_date, invoice_details                 ) VALUES
  (1      ,'Paid'           ,'01/01/2007' , 'Single Invoice for the complete Order')
 --
 ,(1      ,'Issued'          ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(1      ,'Sent1'           ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(2      ,'Canceled'          ,'01/01/2007' , 'Canceled Invoice for the canceled Order')
 ,(3      ,'Issued'          ,'01/01/2007' , 'Combined Invoice for the complete Order')
 ,(3      ,'Sent1'           ,'01/01/2007' , 'Combined Invoice for the complete Order')
 ,(3      ,'Paid'           ,'01/01/2007' , 'Combined Invoice for the complete Order')
 ,(4      ,'Pre-Issue'         ,'01/01/2007' , 'Partial Invoice for the complete Order')
 ,(5      ,'Issued'          ,'01/01/2007' , 'Partial Invoice for the complete Order')
 ,(5      ,'Sent1'           ,'01/01/2007' , 'Partial Invoice for the complete Order')
 ,(5      ,'Paid'           ,'01/10/2007' , 'Partial Invoice for the complete Order')
 ,(6      ,'Pre-Issue'         ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(7      ,'Issued'          ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(7      ,'Sent1'           ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(7      ,'Sent2'           ,'02/01/2007' , 'Single Invoice for the complete Order')
 ,(7      ,'Sent3'           ,'03/01/2007' , 'Single Invoice for the complete Order')
 ,(7      ,'Paid'           ,'03/15/2007' , 'Single Invoice for the complete Order')
 ,(8      ,'Issued'          ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(9      ,'Issued'          ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(10     ,'Issued'          ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(10     ,'Sent1'           ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(10     ,'Contested'         ,'01/15/2007' , 'Contested Invoice for the complete Order')
 ,(11     ,'Issued'          ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(11     ,'Sent1'           ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(11     ,'Contested'         ,'01/15/2007' , 'Contested Invoice for the complete Order')
 ,(11     ,'Under-Rev'         ,'02/01/2007' , 'Under Review Invoice for the complete Order')
 ,(11     ,'Reviewed'          ,'02/15/2007' , 'Reviewed Invoice for the complete Order')
 ,(11     ,'Re-issued'         ,'02/15/2007' , 'Re-issued new Invoice for the complete Order')
 ,(11     ,'Re-sent1'          ,'02/15/2007' , 'Re-sent Invoice for the complete Order')
 ,(11     ,'Paid'           ,'03/10/2007' , 'Re-sent Invoice for the complete Order')
 ,(12     ,'Issued'          ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(13     ,'Issued'          ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(14     ,'Pre-Issue'         ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(15     ,'Issued'          ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(16     ,'Pre-Issue'         ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(17     ,'Issued'          ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(18     ,'Issued'          ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(19     ,'Issued'          ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(20     ,'Pre-Issue'         ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(21     ,'Pre-Issue'         ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(22     ,'Pre-Issue'         ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(23     ,'Issued'          ,'01/01/2007' , 'Single Invoice for the complete Order')
```

```
 ,(23      ,'Sent1'           ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(23      ,'Paid'            ,'01/05/2007' , 'Single Invoice for the complete Order')
 ,(24      ,'Pre-Issue'       ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(25      ,'OTHER'           ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(26      ,'Issued'          ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(26      ,'Sent1'           ,'01/01/2007' , 'Single Invoice for the complete Order')
 ,(26      ,'Contested'       ,'01/15/2007' , 'Contested Invoice for the complete Order')
 ,(26      ,'Under-Rev'       ,'02/01/2007' , 'Under Review Invoice for the complete Order')
 ,(26      ,'Reviewed'        ,'02/15/2007' , 'Reviewed Invoice for the complete Order')
 ,(26      ,'Re-issued'       ,'02/15/2007' , 'Re-issued new Invoice for the complete Order')
 ,(26      ,'Re-sent1'        ,'02/15/2007' , 'Re-sent (1st attempt) Invoice for the complete Order')
 ,(26      ,'Re-sent2'        ,'03/15/2007' , 'Re-sent (2nd attempt) Invoice for the complete Order')
 ,(26      ,'Re-sent3'        ,'04/15/2007' , 'Re-sent (3ed attempt) Invoice for the complete Order')
 ,(26      ,'Collection'      ,'05/15/2007' , 'Collection-Action for Invoice for the complete Order')
 ,(26      ,'Collected'       ,'06/01/2007' , 'Collected invoice for the complete Order')
;
INSERT INTO Payments
 (invoice_number, payment_date, payment_amount) VALUES
 (1          ,'01/01/2007' , 600.00       )
--
,(7          ,'01/01/2007' , 50.00       )
,(7          ,'01/02/2007' , 100.00      )
,(7          ,'01/03/2007' , 100.00      )
,(7          ,'01/04/2007' , 50.00       )
,(11         ,'01/01/2007' , 100.00      )
,(11         ,'01/02/2007' , 200.00      )
,(11         ,'01/03/2007' , 75.00       )
,(11         ,'01/04/2007' , 75.00       )
,(17         ,'01/15/2007' , 100.00      )
,(17         ,'01/25/2007' , 200.00      )
,(17         ,'01/30/2007' , 75.00       )
,(33         ,'01/01/2007' , 10.00       )
,(33         ,'01/02/2007' , 20.00       )
,(33         ,'01/03/2007' , 7.50        )
,(33         ,'01/04/2007' , 7.50        )
,(44         ,'02/01/2007' , 100.00      )
,(44         ,'02/02/2007' , 200.00      )
,(44         ,'02/03/2007' , 75.00       )
,(44         ,'02/04/2007' , 75.00       )
;
INSERT INTO Shipments
 (order_id, invoice_number, shipment_tracking_number, shipment_date,other_shipment_details)
VALUES
 (1      , 1          ,'123456'              , '01/01/2007' , NULL              )
--
,(3      , 5          ,'3-123457'            , '01/01/2007' , NULL              )
,(6      , 12         ,'6-123457'            , '01/01/2007' , NULL              )
,(7      , 13         ,'7-123457'            , '01/02/2007' , NULL              )
```

```
,(9      , 19          ,'9-123457'             , '01/02/2007' , NULL              )
,(10     , 20          ,'10-123457'            , '01/03/2007' , NULL              )
,(11     , 23          ,'11-123457'            , '01/03/2007' , NULL              )
,(12     , 31          ,'12-123457'            , '01/04/2007' , NULL              )
,(15     , 34          ,'15-123457'            , '01/04/2007' , NULL              )
,(16     , 35          ,'16-123457'            , '01/05/2007' , NULL              )
,(16     , 35          ,'16-123457-1'          , '02/05/2007' , NULL              )
,(17     , 36          ,'17-123457'            , '01/06/2007' , NULL              )
,(18     , 37          ,'18-123457'            , '01/07/2007' , NULL              )
,(19     , 38          ,'19-123457'            , '01/08/2007' , NULL              )
,(20     , 39          ,'20-123457'            , '01/09/2007' , NULL              )
,(21     , 40          ,'21-123457'            , '01/10/2007' , NULL              )
,(22     , 41          ,'22-123457'            , '01/11/2007' , NULL              )
,(23     , 42          ,'23-123457'            , '01/12/2007' , NULL              )
,(24     , 45          ,'24-123457'            , '01/13/2007' , NULL              )
,(25     , 46          ,'25-123457'            , '01/14/2007' , NULL              )
;
INSERT INTO Shipment_Items
 (shipment_id,order_item_id) VALUES
 (1         ,1          )
,(1         ,2          )
--
,(2         ,7          )  -- order_id: 3
,(2         ,8          )  -- order_id: 3
,(3         ,13         )  -- order_id: 6
,(4         ,14         )  -- order_id: 7
,(5         ,16         )  -- order_id: 9
,(6         ,17         )  -- order_id: 10
,(7         ,18         )  -- order_id: 11
,(8         ,19         )  -- order_id: 12
,(9         ,22         )  -- order_id: 15
,(10        ,13         )  -- order_id: 16
,(11        ,14         )  -- order_id: 16
,(12        ,25         )  -- order_id: 17
,(13        ,26         )  -- order_id: 18
,(14        ,27         )  -- order_id: 19
,(15        ,28         )  -- order_id: 20
,(16        ,29         )  -- order_id: 21
,(17        ,30         )  -- order_id: 22
,(18        ,31         )  -- order_id: 23
,(19        ,32         )  -- order_id: 24
,(20        ,33         )  -- order_id: 25
;
-----------------------------------------
-- Re-enable FKs
-----------------------------------------
ALTER TABLE Invoices ALTER FOREIGN KEY Invoice_Status_Codes_Invoices
ENFORCED;
```

```
ALTER TABLE Invoices ALTER FOREIGN KEY Orders_Invoices ENFORCED;
ALTER TABLE Orders ALTER FOREIGN KEY Order_Status_Codes_Orders ENFORCED;
ALTER TABLE Orders ALTER FOREIGN KEY Customers_1_Orders ENFORCED;
ALTER TABLE Shipments ALTER FOREIGN KEY Orders_Shipments ENFORCED;
ALTER TABLE Shipments ALTER FOREIGN KEY Invoices_Shipments ENFORCED;
ALTER TABLE Shipment_Items ALTER FOREIGN KEY Shipments_Shipment_Items
ENFORCED;
ALTER TABLE Shipment_Items ALTER FOREIGN KEY Order_Items_Shipment_Items
ENFORCED;
ALTER TABLE Order_Items ALTER FOREIGN KEY Order_Item_Status_Order_Items
ENFORCED;
ALTER TABLE Order_Items ALTER FOREIGN KEY Products_Order_Items ENFORCED;
ALTER TABLE Order_Items ALTER FOREIGN KEY Orders_Order_Items ENFORCED;
ALTER TABLE Products ALTER FOREIGN KEY Ref_Product_Types_Products ENFORCED;
ALTER TABLE Customer_Payment_Methods ALTER FOREIGN KEY
Customers_Customer_Payment_Methods ENFORCED;
ALTER TABLE Customer_Payment_Methods ALTER FOREIGN KEY
Ref_Payment_Methods_Customer_Payment_Methods ENFORCED;
ALTER TABLE Payments ALTER FOREIGN KEY Invoices_Payments ENFORCED;
-- ALTER TABLE Product_Prices ALTER FOREIGN KEY Products_Product_Prices
ENFORCED;
-- ALTER TABLE Product_Prices ALTER FOREIGN KEY Ref_Art_Types_Product_Prices
ENFORCED;
ALTER TABLE Ref_Product_Types ALTER FOREIGN KEY
Ref_Product_Types_Ref_Product_Types ENFORCED;
```

APPENDIX D: Settings in DM2ONT and DataMaster

This appendix contains the property file used to control DM2ONT behavior, and the settings used when running DataMaster.

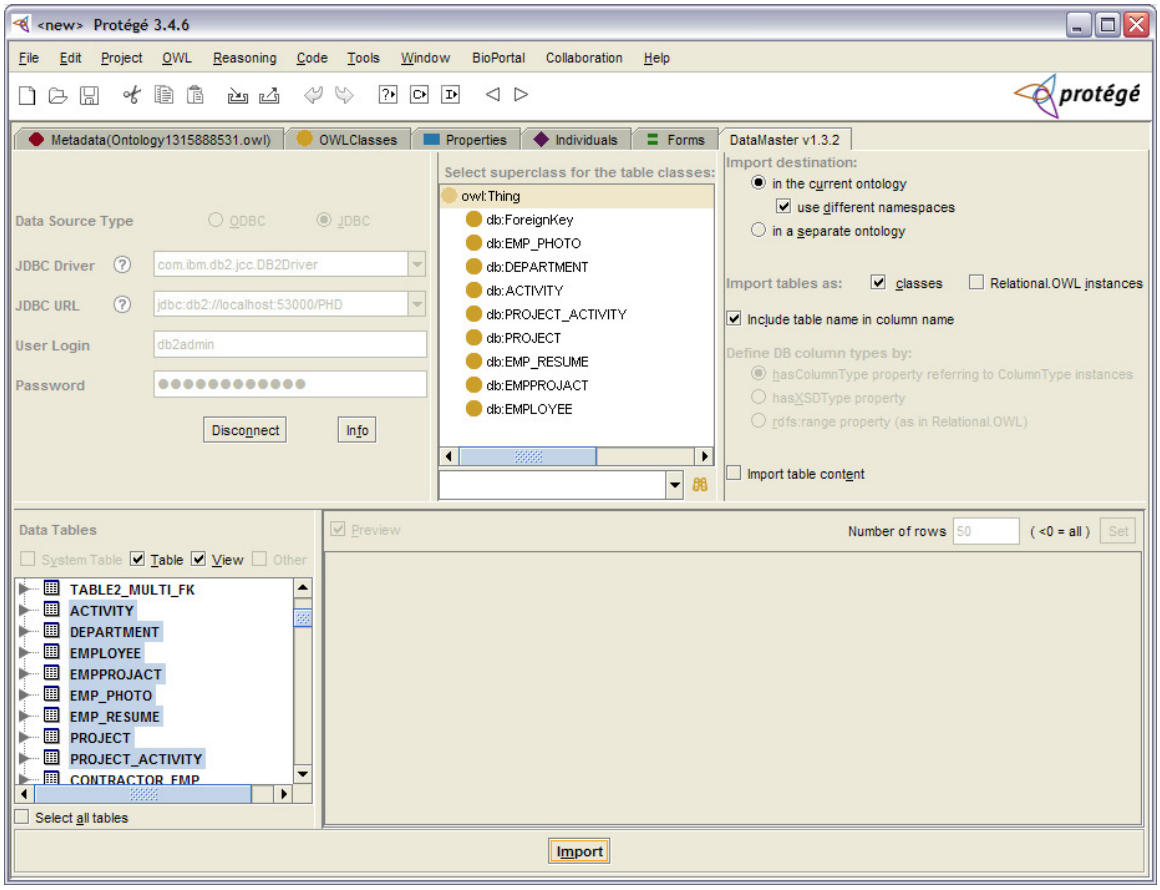
1. DM2ONT Property File:

```
[DM2ONT GENERAL INPUT PROPERTIES]
trace_on=true
[RDB_ORDB INPUT PROPERTIES]
  [DB2 SETTINGS]
jdbc_string=jdbc:db2://localhost:53000/PHD
db_username=<BLOCKED>
db_password=<BLOCKED>
db_schema=ECOMM
  [RDB_ORDB - GENERAL SETTINGS]
isa_type1=true
isa_type1_threshold=0.6
isa_type2=true
isa_type2_threshold=0.6
isa_type2_common_cols=3
  [RDB_ORDB - DATA ANALYSIS SETTINGS]
data_analysis=true
null_data_analysis=true
unique_data_analysis=true
sparse_data_analysis=true
rel_cardinality_data_analysis=true
rel_symmetric_data_analysis=true
rel_transitive_data_analysis=true
sparse_value_threshold=5
confidence_threshold=0.2

[OWLConvertor INPUT PROPERTIES]

[Ontology Generator INPUT PROPERTIES]
-output_file_name=C:\\khalid\\java\\phd\\output_files\\validation\\owl_model_ecomm.owl
--owl_model_namespace=dm
```

2. DataMaster Settings:



APPENDIX E: Domain Requirements and Recall/Precision

This appendix shows the domain requirements used in case-study one and case-study two, and how the ontology axioms generated by DM2ONT and DataMaster map to these requirements. A value of "1" in the tools' column indicates that the ontology model generated by the tool addressed the requirement listed in the row under the Domain Requirements column (i.e. true-positive). A value of "0" indicates that the tool missed the requirement (i.e. false-negative). Axioms that are generated by the tool without corresponding domain requirements (i.e. false-positive) are listed below as well. The information in the tables below is the base for the recall/precision computation conducted in my validation.

1. Case-Study One:

Domain Requirements (DRS)			DRS. (Count)	DM2ONT	DataMaster
Class: Activity	A class describing the activities carried out in the organization. Has the following properties		1	1	1
	ACTNO	Activity ID/Number (Data Property).	1	1	1
		Uniquely identifies an activity	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	ACTKWD	Activity Keyword (Data Property).	1	1	1
		Uniquely identifies an activity	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	ACTDESC	Activity Description (Data Property).	1	1	0
		Uniquely identifies an activity	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	Associated Projects	Identifies the projects associated with an activity (Object Property).	1	1	0
		May have 0 or more projects	1	1	0
Class: Department	A class describing the departments found in the organization. Has the following properties		1	1	1

	DEPTNO	Department ID/Number (Data Property).	1	1	1
		Uniquely identifies a Department	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	DEPTNAME	Department Name (Data Property).	1	1	1
		Uniquely identifies a Department	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	LOCATION	Department Location (Data Property).	1	1	1
		May not have a value (null-able)	1	0	0
		Has one value at most (atomic)	1	1	1
	ROD	Identifies the department to which this department reports to (Object Property).	1	1	1
		This relationships is transitive	1	1	0
		Each department must report to another deprtment (at least 1)	1	1	0
		Each department can report to one department at most	1	1	1
	ROD.Inverse	Identifies the departments reporting into this department (Object Property).	1	1	0
		A department might not have other deprtments reporting into it	1	1	0
	RDE	Identifies the manager (an employee) of this department (Object Property).	1	1	1
		A department might not have a manager (null-able)	1	1	0
		Can have one value at most	1	1	1
	RED.Inverse	Identifies the employees working for this department (Object Property).	1	1	0
		A department might not have employees reporting to it (null-able)	1	1	0
	FK_PROJECT_1_Inverse	Identifies the projects this department is responsible for (Object Property).	1	1	0
		A department might not have projects assigned to it (null-able)	1	1	0
Class: Employee	A class describing the employees working in the organization. Has the following properties		1	1	1
	EMPNO	Employee ID/Number (Data Property).	1	1	1
		Uniquely identifies an employee	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	FIRSTNAME	Employee First-Name (Data Property).	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	MIDINIT	Employee Middle-Initials (Data Property).	1	1	1
		May not have a value (null-able)	1	1	0
		Has one value at most (atomic)	1	1	1
	LASTNAME	Employee Last Name (Data Property).	1	1	1
		Has one value at least (not null)	1	1	0

	Has one value at most (atomic)	1	1	1
PHONENO	Employee Phone Number (Data Property).	1	1	1
	Has one value at least (not null)	1	1	0
	Has one value at most (atomic)	1	1	1
HIREDATE	Employee Hire Date (Data Property).	1	1	1
	Has one value at least (not null)	1	1	0
	Has one value at most (atomic)	1	1	1
JOB	Employee Job Name (Data Property).	1	1	1
	Has one value at least (not null)	1	1	0
	Has one value at most (atomic)	1	1	1
EDLEVEL	Employee Education Level (Data Property).	1	1	1
	Has one value at least (not null)	1	1	0
	Has one value at most (atomic)	1	1	1
SEX	Employee Sex/Gender (Data Property).	1	1	1
	Has one value at least (not null)	1	1	0
	Has one value at most (atomic)	1	1	1
	Can have one of two values only: either 'M' for Male or 'F' for Female	1	1	0
BIRTHDATE	Employee Birth Date (Data Property).	1	1	1
	Has one value at least (not null)	1	1	0
	Has one value at most (atomic)	1	1	1
SALARY	Employee Salary (Data Property).	1	1	1
	Has one value at least (not null)	1	1	0
	Has one value at most (atomic)	1	1	1
BOUNS	Employee Bonus (Data Property).	1	1	1
	Has one value at least (not null)	1	1	0
	Has one value at most (atomic)	1	1	1
COMM	Employee Commission (Data Property).	1	1	1
	Has one value at least (not null)	1	1	0
	Has one value at most (atomic)	1	1	1
RDE.Inverse	Identifies the department managed by this employee	1	1	0
	Each employee may manage a department (at least 0)	1	1	0
	Each employee can manage one department at most	1	1	0
RED	Identifies the department an employee works in	1	1	1
	An employee must works in a department (at least 1)	1	1	0
	An employee can work for one department at most	1	1	1
EMPPROJACT_EMPLOYEE_FK1.Inverse	Project-Activities this employee worked on	1	1	0
	An employee may work in 0 or more project activity (at least 0)	1	1	0
FK_EMP_PHOTO.Inverse	Photos associated with this employee	1	1	0
	An employee may have 0 or more photos (at least 0)	1	0	0

	FK_EMP_RESUME.Inverse	Resumes associated with this employee	1	1	0
		An employee may have 0 or more resume (at least 0)	1	0	0
	FK_PROJECT_2.Inverse	Projects this employee is responsible for	1	1	0
		An employee can be responsible for 0 or more projects (at least 0)	1	1	0
Class: EmpProjAct	A class describing employees' project activities. Has the following properties		1	1	1
	EMPTIME	Employee Time Allocated (Data Property).	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
		Can have one of 4 values only: either '0.25', '0.5', '0.75', or '1.0'.	1	1	0
	EMPENDATE	Emp Assignment End Date (Data Property).	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	EMPPROJACT_EMPLOYEE_FK1	The employee assigned to this project activity	1	1	1
		Must have one employee at least	1	1	0
		Must have one employee at most	1	1	1
	REPAPA	The project activity assigned to this emp	1	1	1
		Must have one project activity at least	1	1	0
		Must have one project activity at most	1	1	1
Class: Emp_Photo	A class containing employees' photos. Has the following properties		1	1	1
	PHOTO_FORMAT	Format of the picture (Data Property).	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	PICTURE	Employee Picture/Image (Data Property).	1	1	1
		Has one value at least (not null)	1	0	0
		Has one value at most (atomic)	1	1	1
	FK_EMP_PHOTO	The employee associated with this photo	1	1	1
		Must have one employee at least	1	1	0
		Must have one employee at most	1	1	1
Class: Emp_Resume	A class containing employees' resumes. Has the following properties		1	1	1
	RESUME_FORMAT	Format of the Resume (Data Property).	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	RESUME	Employee Resume (Data Property).	1	1	1
		Has one value at least (not null)	1	0	0
		Has one value at most (atomic)	1	1	1
	FK_EMP_RESUME	The employee associated with this resume	1	1	1
		Must have one employee at least	1	1	0
		Must have one employee at most	1	1	1
Class: PROJECT	A class describing the projects in the organization. Has the following properties		1	1	1
	PROJNO	Project Number (Data Property).	1	1	1

		Uniquely identifies a Project	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	PROJNAME	Project Name (Data Property).	1	1	1
		Uniquely identifies a Project	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	PRSTAFF	Project Staffing Requirement (Data Property).	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	PRSTDATE	Project Start Date (Data Property).	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	PRENDATE	Project End Date (Data Property).	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	FK_PROJECT_1	The department responsible for this project	1	1	1
		Each project must have a department responsible for it (at least 1)	1	1	0
		Each project can have only one department responsible for it (at most 1)	1	1	1
	FK_PROJECT_2	The employee responsible for this project	1	1	1
		Each project must have an employee responsible for it (at least 1)	1	1	0
		Each project can have only one employee responsible for it (at most 1)	1	1	1
	RPP	The major project to which this project belong to	1	1	1
		A project can be part of a major project (at least 0)	1	1	0
		A project can be part of only one major project (at most 1)	1	1	1
		The project to major-project relationship is transitive	1	1	0
	RPP.Inverse	The sub-project of this project (inverse of major project relationship)	1	1	0
		A project can have sub-projects (at least 0)	1	1	0
	RPAP.Inverse	Activities associated with this project	1	1	0
		A project can be associated with 0 or more projects (at least 0)	1	1	0
Class: Project_Activity	A class describing the activities associated with the projects. Has the following properties		1	1	1
	ACSTAFF	Project/Activity Staffing Needs (Data Property)	1	1	1
		May not have a value (null-able)	1	0	0
		Has one value at most (atomic)	1	1	1
	ACSTDATE	Project/Activity start date (Data Property)	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	ACENDATE	Project/Activity end date (Data Property)	1	1	1

		May not have a value (null-able)	1	0	0
		Has one value at most (atomic)	1	1	1
	REPAPA.Inverse	Employees associated with this project-activity	1	1	0
		Each project/activity pair can have 1 or more employees (at least 1)	1	1	0
	PROJACT_ACT_FK	Activity associated with this project/activity pair	1	1	1
		Each project/activity pair must have an activity associated with it (at least 1)	1	1	0
		A project/activity pair can have one activity at most	1	1	1
	RPAP	Project associated with this project/activity pair	1	1	1
		Each project/activity pair must have a project associated with it (at least 1)	1	1	0
		A project/activity pair can have one project at most	1	1	1
Total # of Relevant/Valid Statements			**180**	**173**	**97**
Recall (relv axioms retrieved / relv axioms in ref Onto)			**n/a**	**0.961**	**0.539**

Invalid assertions by DM2ONT and DataMaster

DM2ONT					
Class: PROJECT	PRSTDATE	Project Start Date has sparse values	0	1	0
	PRENDATE	Project End Date has sparse values	0	1	0
DataMaster					
Class: Department	MGRNO	Department Manager (Data Property).	0	0	1
		Has one value at most (atomic)	0	0	1
	ADMRDEPT	Department to which this department reports to (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
Class: Employee	WORKDEPT	Identifies the department an employee works in (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
Class: EmpProjAct	EMPNO	Employee Number/ID (Data Property).	0	0	1
		Has one value at most (atomic)	0	0	1
	ACTNO	Activity Number/ID (Data Property).	0	0	1
		Has one value at most (atomic)	0	0	1
	EMSTDATE	Employee Project-Activity Start Date (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
	PROJNO	Project Number (Data Property).	0	0	1
		Has one value at most (atomic)	0	0	1
	EMSTDATE_INSTANCE	A 2nd relationship to Project-Activity class	0	0	1
		Has one value at most (atomic)	0	0	1
	PROJNO_INSTANCE	A 3ed relationship to Project-Activity class	0	0	1
		Has one value at most (atomic)	0	0	1
Class:	EMPNO	Employee Number/ID (Data Property).	0	0	1

Emp_Photo		Has one value at most (atomic)	0	0	1
Class: Emp_Resume	EMPNO	Employee Number/ID (Data Property).	0	0	1
		Has one value at most (atomic)	0	0	1
Class: PROJECT	DEPTNO	Department ID/Number (Data Property).	0	0	1
		Has one value at most (atomic)	0	0	1
	MAJPROJ	Major project to which this project is part of (Data Propery)	0	0	1
		Has one value at most (atomic)	0	0	1
	RESPEMP	Employee responsible for this project (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
Class: Project_Activity	ACTNO	Activity Number/ID (Data Property).	0	0	1
		Has one value at most (atomic)	0	0	1
	PROJNO	Project Number (Data Property).	0	0	1
		Has one value at most (atomic)	0	0	1
Total # of invalid Statements			**n/a**	**2**	**32**
Precision (relv axioms retrieved / retrieved axioms)				**0.989**	**0.752**

2. Case-Study Two:

Domain Requirements (DRS)			DRS (Count)	DM2ONT	DataMaster
Class: CUSTOMERS	A class describing the customers transacting with the organization. Has the following properties		1	1	1
	CUSTOMER_ID	Customer ID (Data Property).	1	1	1
		Uniquely identifies a customer	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	ORGANIZATION_OR_CUSTOMER	Indicates whether the customer is an Organization or a Person (Data Property).	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
		can have one of two values: "Org" or "Person".	1	1	0
	ORGANIZATION_NAME	Identifies the Organization's name	1	1	1
		May not have a value (null-able)	1	0	0
		Has one value at most (atomic)	1	1	1
	GENDER	Indicates whether the customer is Female or Male (Data Property).	1	1	1
		May not have a value (null-able)	1	1	0
		Has one value at most (atomic)	1	1	1
		Can have one of two values: "F" for Female or "M" for Male	1	1	0
	FIRST_NAME	Customer First-Name (Data Property).	1	1	1
		May not have a value (null-able)	1	1	0
		Has one value at most (atomic)	1	1	1
	MIDDLE_INITIAL	Customer Middle-Initials (Data Property).	1	1	1
		May not have a value (null-able)	1	1	0

	Has one value at most (atomic)	1	1	1
LAST_NAME	Customer Last Name (Data Property).	1	1	1
	May not have a value (null-able)	1	1	0
	Has one value at most (atomic)	1	1	1
EMAIL_ADDRESS	Customer email address (Data Property).	1	1	1
	Uniquely identifies a customer	1	1	0
	Has one value at least (not null)	1	1	0
	Has one value at most (atomic)	1	1	1
LOGIN_NAME	Customer login name (Data Property).	1	1	1
	Uniquely identifies a customer	1	1	0
	Has one value at least (not null)	1	1	0
	Has one value at most (atomic)	1	1	1
LOGIN_PASSWORD	Customer login password (Data Property).	1	1	1
	Has one value at least (not null)	1	1	0
	Has one value at most (atomic)	1	1	1
PHONE_NUMBER	Customer Phone Number (Data Property).	1	1	1
	May not have a value (null-able)	1	0	0
	Has one value at most (atomic)	1	1	1
ADDRESS_LINE_1	Customer address line 1 (Data Property).	1	1	1
	Has one value at least (not null)	1	1	0
	Has one value at most (atomic)	1	1	1
ADDRESS_LINE_2	Customer address line 2 (Data Property).	1	1	1
	Has one value at least (not null)	1	1	0
	Has one value at most (atomic)	1	1	1
ADDRESS_LINE_3	Customer address line 3 (Data Property).	1	1	1
	May not have a value (null-able)	1	0	0
	Has one value at most (atomic)	1	1	1
ADDRESS_LINE_4	Customer address line 4 (Data Property).	1	1	1
	May not have a value (null-able)	1	0	0
	Has one value at most (atomic)	1	1	1
TOWN_CITY	Customer Town/City (Data Property).	1	1	1
	Has one value at least (not null)	1	1	0
	Has one value at most (atomic)	1	1	1
COUNTY	Customer County (Data Property).	1	1	1
	May not have a value (null-able)	1	0	0
	Has one value at most (atomic)	1	1	1
COUNTRY	Customer Country (Data Property).	1	1	1
	Has one value at least (not null)	1	1	0
	Has one value at most (atomic)	1	1	1
CUSTOMERS_CUSTOMER_PAYMENT_METHODS.Inverse	The payment methods used by this customer (Object Property)	1	1	0
	May have 0 or more payment methods	1	1	0
CUSTOMERS_1_ORDERS.Inverse	The orders placed by this customer (Object Property)	1	1	0

		Each customer must have one order at least	1	1	0
Class: CUSTOMER_PAYMENT_METHODS	A class describing the payment methods associate with (made by) a customer. Has the following properties		1	1	1
	CUSTOMER_PAYMENT_ID	Customer Payment ID (Data Property)	1	1	1
		Uniquely identifies a payment method for a customer	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	CREDIT_CARD_NUMBER	Customer Credit Card Number (Data Property)	1	1	1
		May not have a value (null-able)	1	0	0
		Has one value at most (atomic)	1	1	1
	PAYMENT_METHOD_DETAILS	Details/Description for payment method (Data Property)	1	1	1
		May not have a value (null-able)	1	1	0
		Has one value at most (atomic)	1	1	1
	CUSTOMERS_CUSTOMER_PAYMENT_METHODS	The customer associated with this payment method (Object Property)	1	1	1
		A customer payment method must be associated with a customer (at least 1)	1	1	0
		Can have one value at most	1	1	1
	REF_PAYMENT_METHODS_CUSTOMER_PAYMENT_METHODS	The reference payment method associated with this customer payment method (Object Property)	1	1	1
		A customer payment method must be associated with a ref payment method (at least 1)	1	1	0
		Can have one value at most	1	1	1
Class:INVOICES	A class describing all the invoices issued by the organization for every order. Has the following properties		1	1	1
	INVOICE_NUMBER	Invoice Number (Data Property)	1	1	1
		Uniquely identifies an invoice	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	INVOICE_DATE	The date the invoice was issued in (Data Property)	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	INVOICE_DETAILS	Invoice Details (Data Property)	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	ORDERS_INVOICES	The order associated with this invoice (Object Property)	1	1	1
		Each invoice is associaed with one order at least	1	1	0
		Each invoice is associaed with one order at most	1	1	1
	INVOICE_STATUS_CODES_INVOICES	The status code associated with this invoice (Object Property)	1	1	1
		Each invoice is associaed with one invoice status code at least	1	1	0
		Each invoice is associaed with one invoice status code at most	1	1	1
	INVOICES_PAYMENTS.Inverse	The payments associated with this invoice (Object Property)	1	1	0
		May have 0 or more payments (at least 0)	1	1	0
	INVOICES_SHIP				
		The shipments associated with this invoice	1	1	0

	MENTS.Inverse	(Object Property)			
		May have 0 or more shipments (at least 0)	1	1	0
Class: ORDERS	A class describing the orders placed by customers in the origanization. Has the following properties		1	1	1
	ORDER_ID	Order ID (Data Property)	1	1	1
		Uniquely identifies an order within the organization	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	DATE_ORDER_PLACED	Date the order was placed (Data Property)	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	ORDER_DETAILS	Details pertaining to the order (Data Property)	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	ORDERS_INVOICES.Inverse	The invoices associated with this order (Object Property)	1	1	0
		Each order must have one invoice associated with it (at least 1)	1	1	0
	CUSTOMERS_1_ORDERS	The customer associated with this order (Object Property)	1	1	1
		Each order must have at least one customer associated with it	1	1	0
		Each order must have at most one customer associated with it	1	1	1
	ORDER_STATUS_CODES_ORDERS	The order status code associated with this order (Object Property)	1	1	1
		Each order must have at least one order status code associated with it	1	1	0
		Each order must have at most one order status code associated with it	1	1	1
	ORDERS_ORDER_ITEMS.Inverse	The order items associated with this order (Object Property)	1	1	0
		Each order must have at least one order item associated with it	1	1	0
	ORDERS_SHIPMENTS.Inverse	The shipments associated with this order (Object Property)	1	1	0
		An order may have 0 or more shipments associated with it (at least 0)	1	1	0
Class: ORDER_ITEMS	A class describing the order items within an order. Has the following properties		1	1	1
	ORDER_ITEM_ID	Order Item ID (Data Property)	1	1	1
		Uniquely identifies an order item within the organization	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	ORDER_ITEM_QUANTITY	Quantity requested for the order item (Data Property)	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	ORDER_ITEM_PRICE	Order Item Price (Data Property)	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1

	OTHER_ORDER_ITEM_DETAILS	Order Item Details (Data Property)	1	1	1
		May have a value (null-ablel)	1	1	0
		Has one value at most (atomic)	1	1	1
	ORDERS_ORDER_ITEMS	The order associated with this order item (Object Property)	1	1	1
		Each order item is associated with at least one order	1	1	0
		Each order item is associated with at most one order	1	1	1
	PRODUCTS_ORDER_ITEMS	The product associated with this order item (Object Property)	1	1	1
		Each order item is associated with at least one product	1	1	0
		Each order item is associated with at most one product	1	1	1
	ORDER_ITEM_STATUS_ORDER_ITEMS	The order item status associated with this order item (Object Property)	1	1	1
		Each order item is associated with at least one order item status	1	1	0
		Each order item is associated with at most one order item status	1	1	1
	SHIPMENTS_SHIPMENT_ITEMS	The shipments associated with this order item (Object Property)	1	1	0
		Each order item is associated with at least one shipment	1	1	0
Class: PAYMENTS	A class describing the payments associated with an invoice. Has the following properties		1	1	1
	PAYMENT_ID	Payment ID (Data Property)	1	1	1
		Uniquely identifies a payment within the organization	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	PAYMENT_DATE	The date the payment was made (Data Property)	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	PAYMENT_AMOUNT	The amount associated with this payment (Data Property)	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	INVOICES_PAYMENTS	The invoices associated with this payment (Object Property)	1	1	1
		Each payment is associated with at least one invoice	1	1	0
		Each payment is associated with at most one invoice	1	1	1
Class: PRODUCTS	A class describing the products carried out by the organization. Has the following properties		1	1	1
	PRODUCT_ID	Product ID (Data Property)	1	1	1
		Uniquely identifies a product within the organization	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	PRODUCT_NAME	Product Name (Data Property)	1	1	1
		Uniquely identifies a product within the	1	1	0

		organization			
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	PRODUCT_PRICE	Product Price (Data Property)	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	PRODUCT_COLOR	Product Color (Data Property)	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	PRODUCT_SIZE	Product size (Data Property)	1	1	1
		May have a value (null-able)	1	1	0
		Has one value at most (atomic)	1	1	1
	PRODUCT_DESCRIPTION	Product Description (Data Property)	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	OTHER_PRODUCT_DETAILS	Other Product Details (Data Property)	1	1	1
		May have a value (null-able)	1	0	0
		Has one value at most (atomic)	1	1	1
	PRODUCTS_ORDER_ITEMS.Inverse	The order items associated with this product (Object Property)	1	1	0
		Each product is associated with at least one order item	1	1	0
	REF_PRODUCT_TYPES_PRODUCTS	The product type associated with this product (Object Property)	1	1	1
		Each product is associated with at least one product type	1	1	0
		Each product is associated with at most one product type	1	1	1
Class: REF_INVOICE_STATUS_CODES	A class desribing the Invoice Status Codes found in the organization. Has the following properties		1	1	1
	INVOICE_STATUS_CODE	Invoice Status Code (Data Property)	1	1	1
		Uniquely identifies an invoice status within the organization	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	INVOICE_STATUS_DESCRIPTION	Invoice Status Description (Data Property)	1	1	1
		Uniquely identifies an invoice status within the organization	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	INVOICE_STATUS_CODES_INVOICES.Inverse	The invoices associated with this invoice status (Object Property)	1	1	0
		Each invoice status is associated with at least one invoice	1	1	0
Class: REF_ORDER_ITEM_STATUS_CODES	A class desribing the Order Item Status Codes found in the organization. Has the following properties		1	1	1
	ORDER_ITEM_STATUS_CODE	Order Item Status Code (Data Property)	1	1	1
		Uniquely identifies an order item status within the organization	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	ORDER_ITEM_S	Order Item Status Description (Data Property)	1	1	1

	TATUS_DESCRIPTION	Uniquely identifies an order item status within the organization	1	0	0
		Has one value at least (not null)	1	0	0
		Has one value at most (atomic)	1	1	1
	ORDER_ITEM_STATUS_ORDER_ITEMS.Inverse	The order items associated with this order item status (Object Property)	1	1	0
		Each order item status is associated with at least one order item	1	1	0
Class: REF_ORDER_STATUS_CODES	A class desribing the Order Status Codes found in the organization. Has the following properties		1	1	1
	ORDER_STATUS_CODE	Order Status Code (Data Property)	1	1	1
		Uniquely identifies an order status within the organization	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	ORDER_ITEM_DESCRIPTION	Order Status Description (Data Property)	1	1	1
		Uniquely identifies an order status within the organization	1	0	0
		Has one value at least (not null)	1	0	0
		Has one value at most (atomic)	1	1	1
	ORDER_STATUS_CODES_ORDERS.Inverse	The orders associated with this order status (Object Property)	1	1	0
		Each order status is associated with at least one order	1	1	0
Class: REF_PAYMENT_METHODS	A class describing the payment methods accpeted by the organization.Has the following properties		1	1	1
	PAYMENT_METHOD_CODE	Payment Method Code (Data Property)	1	1	1
		Uniquely identifies a payment method within the organization	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	PAYMENT_METHOD_DESCRIPTION	Payment Method Description (Data Property)	1	1	1
		Uniquely identifies a payment method within the organization	1	0	0
		Has one value at least (not null)	1	0	0
		Has one value at most (atomic)	1	1	1
	REF_PAYMENT_METHODS_CUSTOMER_PAYMENT_METHODS.Inverse	Identifies the customer payment methods associated with this payment method (Object Property)	1	1	0
		Each ref payment method is associated with at least one customer payment method	1	1	0
Class: REF_PRODUCT_TYPES	A class describing the product types found in the organization. Has the following properties		1	1	1
	PRODUCT_TYPE_CODE	Product Type Code (Data Property)	1	1	1
		Uniquely identifies a product type within the organization	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	PRODUCT_TYPE_DESCRIPTION	Product Type Description (Data Property)	1	1	1
		Uniquely identifies a product type within the organization	1	1	0
		Has one value at least (not null)	1	1	0

		Has one value at most (atomic)	1	1	1
	REF_PRODUCT_TYPES_PRODUCTS.Inverse	Identifies the products associated with this product type (Object Property)	1	1	0
		Each product type is associated with zero or more products (at least 0)	1	1	0
	REF_PRODUCT_TYPES_REF_PRODUCT_TYPES	Identifies the product (super-)type associated with this product type (Object Property)	1	1	1
		This relationships is transitive	1	1	0
		Each product type might be associated with 0 or more product type (at least 0)	1	1	0
		Each product type can be associated with at most 1 product type	1	1	1
	REF_PRODUCT_TYPES_REF_PRODUCT_TYPES.Inverse	Identifies the product (sub-)types associated with this product type (Object Property)	1	1	0
		Each product type is associated with 0 or more sub-types (at least 0)	1	1	0
Class: SHIPMENTS	A class describing the shipments made by the organization. Has the following properties		1	1	1
	SHIPMENT_ID	SHIPMENT ID (Data Property)	1	1	1
		Uniquely identifies a shipment within the organization	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	SHIPMENT_TRACKING_NUMBER	Shipment tracking number for a specific shipment (Data Property)	1	1	1
		Uniquely identifies a shipment within the organization	1	1	0
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	SHIPMENT_DATE	The date the shipment was made (Data Property)	1	1	1
		Has one value at least (not null)	1	1	0
		Has one value at most (atomic)	1	1	1
	OTHER_SHIPMENT_DETAILS	Additional Shipment Details (Data Property)	1	1	1
		May have a value (null-ablel)	1	0	0
		Has one value at most (atomic)	1	1	1
	INVOICES_SHIPMENTS	The invoice associated with this shipment (Object Property)	1	1	1
		Each shipment is assoc. with at least 1 invoice	1	1	0
		Each shipment is assoc. with at most 1 invoice	1	1	1
	ORDERS_SHIPMENTS	The order associated with this shipment (Object Property)	1	1	1
		Each shipment is associated with at least 1 order	1	1	0
		Each shipment is associated with at most 1 order	1	1	1
	ORDER_ITEMS_SHIPMENT_ITEMS	The order items associated with this shipment (Object Property)	1	1	0
		Each shipment is associated with at least one order item	1	1	0
Total # of Relevant/Valid Statements			**277**	**263**	**151**
Recall (relv axioms retrieved / relv axioms in ref Onto)			**n/a**	**0.949**	**0.545**

Invalid assertions by DM2ONT and DataMaster

DM2ONT					
Class: CUSTOMERS	COUNTRY	Can have one of two values: "USA" or "UK".	0	1	0
DataMaster					
Class: CUSTOMER_PAYMENT_METHODS	CUSTOMER_ID	Customer ID (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
	PAYMENT_METHOD_CODE	Payment Method Code (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
Class: INVOICES	INVOICE_STATUS_CODE	Invoice Status Code (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
	ORDER_ID	Order ID (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
Class: ORDERS	CUSTOMER_ID	Customer ID (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
	ORDER_STATUS_CODE	Order Status Code (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
Class: ORDER_ITEMS	ORDER_ID	Order ID (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
	ORDER_ITEM_STATUS_CODE	Order Item Status Code (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
	PRODUCT_ID	Product ID (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
Class: PAYMENTS	INVOICE_NUMBER	Invoice Number/ID (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
Class: PRODUCTS	PRODUCT_TYPE_CODE	Product Type Code (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
Class: REF_PRODUCT_TYPE	PARENT_PRODUCT_TYPE_CODE	Parent Product Type Code (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
Class: SHIPMENTS	INVOICE_NUMBER	Invoice Number/ID (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
	ORDER_ID	Order ID (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
Class: SHIPMENT_ITEMS	Class associating the shipments with the order items they contain (a many-to-many rel). Has the following properties		0	0	1
	ORDER_ITEM_ID	Order Item ID (Data Property)		0	1
		Has one value at most (atomic)	0	0	1
	SHIPMENT_ID	Shipment ID (Data Property)	0	0	1
		Has one value at most (atomic)	0	0	1
	ORDER_ITEM_ID_INSTANCE	Order Item associated with a shipment (Object Property)	0	0	1
		Has one value at most (atomic)	0	0	1
	SHIPMENT_ID_I				
		Shipment associated with an order item (Object	0	0	1

	NSTANCE	Property)			
		Has one value at most (atomic)	0	0	1
Total # of invalid Statements			**n/a**	**1**	**37**
Precision (relv axioms retrieved / retrieved axioms)				**0.996**	**0.803**

APPENDIX F: Explicitness Measurement Methodology

For each case-study, the results from the matching phase of the explicitness measurement methodology were sets of Matched Class Pairs (MCP), Matched Data-type property Pairs (MDP), and Matched Object property Pairs (MOP). This appendix lists the sets generated in each case-study with entities from DM2ONT designated as anchors. Since most of the entities generated by DM2ONT and DataMaster share the same name, and to avoid ambiguity and preserve space, entities from DM2ONT were suffixed with "1" below (e.g. $Name_1$) while those generated by DataMaster were suffixed with 2 (e.g. $Name_2$).

1. Case-Study One:

- MCP(om_1 ,om_2) = { ($Activity_1$, $Activity_2$), ($Department_1$, $Department_2$),
  ($Employee_1$, $Employee_2$), ($EmpProjAct_1$, $EmpProjAct_2$),
  (Emp_Photo_1, Emp_Photo_2), (Emp_Resume_1, Emp_Resume_2),
  ($Project_1$, $Project_2$), ($Project\_Activity_1$, $Project\_Activity_2$) }
- MDP($Activity_1$, $Activity_2$) = {($ACTNO_1$, $ACTNO_2$), ($ACTKWD_1$, $ACTKWD_2$),
  ($ACTDESC_1$, $ACTDESC_2$)}
- MOP($Activity_1$, $Activity_2$) = {($PROJACT\_ACT\_FK.Inverse_1$, null)}
- MDP($Department_1$, $Department_2$) = { ($DEPTNO_1$, $DEPTNO_2$) ,
  ($DEPTNAME_1$, $DEPTNAME_2$), ($LOCATION_1$, $LOCATION_2$)}
- MOP($Department_1$, $Department_2$) = { (ROD_1, $ADMR\_DEPT\_INST_2$),
  ($ROD.Inverse_1$, null), (RDE_1, $MGRNO\_INST_2$), ($RED.Inverse_1$, null),
  ($FK.PROJECT\_1.Inverse_1$, null) }
- MDP($Employee_1$, $Employee_2$) = { ($EMPNO_1$, $EMPNO_2$),
  ($FIRSTNAME_1$, $FIRSTNAME_2$), ($MIDINIT_1$, $MIDINIT_2$),
  ($LASTNAME_1$, $LASTNAME_2$), ($PHONENO_1$, $PHONENO_2$),
  ($HIREDATE_1$, $HIREDATE_2$), (JOB_1, JOB_2), ($EDLEVEL_1$, $EDLEVEL_2$),
  (SEX_1, SEX_2), ($BIRTHDATE_1$, $BIRTHDATE_2$), ($SALARY_1$, $SALARY_2$),
  ($BONUS_1$, $BONUS_2$), ($COMM_1$, $COMM_2$)}
- MOP($Employee_1$, $Employee_2$) = {($RDE.Inverse_1$, null),
  (RED_1, $WORKDEPT\_INSTANCE_2$),
  ($EMPPROJACT\_EMPLOYEE\_FK1.Inverse_1$, null),

(FK_EMP_PHOTO.Inverse$_1$, null), (FK_EMP_RESUME.Inverse$_1$, null),
(FK_PROJECT_2.Inverse$_1$, null)}

- MDP(EmpProjAct$_1$, EmpProjAct$_2$) = { (EMPTIME$_1$, EMPTIME$_2$),
(EMPENDATE$_1$, EMPENDATE$_2$)}
- MOP(EmpProjAct$_1$, EmpProjAct$_2$) = {
(EMPPROJACT_EMPLOYEE_FK1$_1$, EMPNO_INSTANCE$_2$)
(REPAPA$_1$, <ACTNO,…>_INSTANCE$_2$)}
- MDP(Emp_Photo$_1$, Emp_Photo$_2$) = { (PHOTO_FORMAT$_1$, PHOTO_FORMAT$_2$),
(PICTURE$_1$, PICTURE$_2$)}
- MOP(Emp_Photo$_1$, Emp_Photo$_2$) = {(FK_EMP_PHOTO$_1$, EMPNO_INSTANCE$_2$)}
- MDP(Emp_Resume$_1$, Emp_Resume$_2$) = {
(RESUME_FORMAT$_1$, RESUME_FORMAT$_2$) ,
(RESUME$_1$, RESUME$_2$)}
- MOP(Emp_Resume$_1$, Emp_Resume$_2$) = {
(FK_EMP_RESUME$_1$, EMPNO_INSTANCE$_2$)}
- MDP(Project$_1$, Project$_2$) = { (PROJNO$_1$, PROJNO$_2$) , (PROJNAME$_1$, PROJNAME$_2$),
(PRSTAFF$_1$, PRSTAFF$_2$), (PRSTDATE$_1$, PRSTDATE$_2$),
(PRENDATE$_1$, PRENDATE$_2$)}
- MOP(Project$_1$, Project$_2$) = {(FK_PROJECT_1$_1$, DEPTNO_INSTANCE$_2$),
(FK_PROJECT_2$_1$, RESPEMP_INSTANCE$_2$),
(RPP$_1$, MAJPROJ_INSTANCE$_2$), (RPP.Inverse$_1$, null),
(RPAP.Inverse$_1$, null)}
- MDP(Project_Activity$_1$, Project_Activity$_2$) = { (ACSTAFF$_1$, ACSTAFF$_2$),
(ACSTDATE$_1$, ACSTDATE$_2$), (ACENDATE$_1$, ACENDATE$_2$)}
- MOP(Project_Activity$_1$, Project_Activity$_2$) = {(REPAPA.Inverse$_1$, null),
(PROJACT_ACT_FK$_1$, ACTNO_INSTANCE$_2$),
(RPAP$_1$, PROJNO_INSTANCE$_2$)}

2. Case-Study Two:

- MCP(om$_1$,om$_2$) = { (Customers$_1$, Customers$_2$),
(Customer_Payment_Methods$_1$, Customer_Payment_Methods$_2$),
(Invoices$_1$,Invoices$_2$), (Orders$_1$,Orders$_2$), (Order_Items$_1$,Order_Items$_2$),
(Payments$_1$,Payments$_2$), (Products$_1$, Products$_2$),
(Ref_Invoice_Status_Codes$_1$, Ref_Invoice_Status_Codes$_2$),
(Ref_Order_Item_Status_Codes$_1$, Ref_Order__Item_Status_Codes$_2$),
(Ref_Order_Status_Codes$_1$, Ref_Order_Status_Codes$_2$),
(Ref_Payment_Methods$_1$, Ref_Payment_Methods$_2$),
(Ref_Product_Types$_1$, Ref_Product_Types$_2$), (Shipments$_1$, Shipments$_2$) }
- MDP(Customers$_1$, Customers$_2$) = { (CUSTOMER_ID$_1$, CUSTOMER_ID$_2$),
(ORGANIZATION_OR_CUSTOMER$_1$, ORGANIZATION_OR_CUSTOMER$_2$),

(ORGANIZATION_NAME$_1$, ORGANIZATION_NAME$_2$),
(GENDER$_1$, GENDER$_2$), (FIRST_NAME$_1$, FIRST_NAME$_2$),
(MIDDLE_INITIAL$_1$, MIDDLE_INITIAL$_2$), (LAST_NAME$_1$, LAST_NAME$_2$),
(EMAIL_ADDRESS$_1$, EMAIL_ADDRESS$_2$),
(LOGIN_NAME$_1$, LOGIN_NAME$_2$),
(LOGIN_PASSWORD$_1$, LOGIN_PASSWORD$_2$),
(PHONE_NUMBER$_1$, PHONE_NUMBER$_2$),
(ADDRESS_LINE_1$_1$, ADDRESS_LINE_1$_2$),
(ADDRESS_LINE_2$_1$, ADDRESS_LINE_2$_2$),
(ADDRESS_LINE_3$_1$, ADDRESS_LINE_3$_2$),
(ADDRESS_LINE_4$_1$, ADDRESS_LINE_4$_2$),
(TOWN_CITY$_1$, TOWN_CITY$_2$), (COUNTY$_1$, COUNTY$_2$),
(COUNTRY$_1$, COUNTRY$_2$) }

- MOP(Customers$_1$, Customers$_2$) = {
  (CUSTOMERS_CUSTOMER_PAYMENT_METHODS.Inverse$_1$, null),
  (CUSTOMERS_1_ORDERS.Inverse$_1$, null) }
- MDP(Customer_Payment_Methods$_1$, Customer_Payment_Methods$_2$) = {
  (CUSTOMER_PAYMENT_ID$_1$, CUSTOMER_PAYMENT_ID$_2$),
  (CREDIT_CARD_NUMBER$_1$, CREDIT_CARD_NUMBER$_2$),
  (PAYMENT_METHOD_DETAILS$_1$, PAYMENT_METHOD_DETAILS$_2$) }
- MOP(Customer_Payment_Methods$_1$, Customer_Payment_Methods$_2$) = {
  (CUSTOMERS_CUSTOMER_PAYMENT_METHODS$_1$,
  CUSTOMER_ID_INSTANCE$_2$),
  (REF_PAYMENT_METHODS_CUSTOMER_PAYMENT_METHODS$_1$,
  PAYMENT_METHOD_CODE_INSTANCE$_2$) }
- MDP(Invoices$_1$,Invoices$_2$) = { (INVOICE_NUMBER$_1$, INVOICE_NUMBER$_2$),
  (INVOICE_DATE$_1$, INVOICE_DATE$_2$),
  (INVOICE_DETAILS$_1$, INVOICE_DETAILS$_2$) }
- MOP(Invoices$_1$,Invoices$_2$) = { (ORDERS_INVOICES$_1$, ORDER_ID_INSTANCE$_2$),
  (INVOICE_STATUS_CODES_INVOICES$_1$,
  INVOICE_STATUS_CODE_INSTANCE$_2$),
  (INVOICES_PAYMENTS.Inverse$_1$, null),
  (INVOICES_SHIPMENTS.Inverse$_1$, null) }
- MDP(Orders$_1$,Orders$_2$) = { (ORDER_ID$_1$, ORDER_ID$_2$),
  (DATE_ORDER_PLACED$_1$, DATE_ORDER_PLACED$_2$),
  (ORDER_DETAILS$_1$, ORDER_DETAILS$_2$) }
- MOP(Orders$_1$,Orders$_2$) = { (ORDERS_INVOICES.Inverse$_1$, null),
  (CUSTOMERS_1_ORDERS$_1$, CUSTOMER_ID_INSTANCE$_2$),

(ORDER_STATUS_CODES_ORDERS$_1$,
ORDER_STATUS_CODE_INSTANCE$_2$),
(ORDERS_ORDER_ITEMS.Inverse$_1$, null),
(ORDERS_SHIPMENTS.Inverse$_1$, null) }

- MDP(Order_Items$_1$,Order_Items$_2$) = { (ORDER_ITEM_ID$_1$, ORDER_ITEM_ID$_2$),
(ORDER_ITEM_QUANTITY$_1$, ORDER_ITEM_QUANTITY$_2$),
(ORDER_ITEM_PRICE$_1$, ORDER_ITEM_PRICE$_2$),
(OTHER_ORDER_ITEM_DETAILS$_1$, OTHER_ORDER_ITEM_DETAILS$_2$) }
- MOP(Order_Items$_1$,Order_Items$_2$) = {
(ORDERS_ORDER_ITEMS$_1$, ORDER_ID_INSTANCE$_2$),
(PRODUCTS_ORDER_ITEMS$_1$, PRODUCT_ID_INSTANCE$_2$),
(ORDER_ITEM_STATUS_ORDER_ITEMS$_1$,
ORDER_ITEM_STATUS_CODE_INSTANCE$_2$),
(SHIPMENTS_SHIPMENT_ITEMS$_1$, null) }
- MDP(Payments$_1$,Payments$_2$) = { (PAYMENT_ID$_1$, PAYMENT_ID$_2$),
(PAYMENT_DATE$_1$, PAYMENT_DATE$_2$),
(PAYMENT_AMOUNT$_1$, PAYMENT_AMOUNT$_2$)}
- MOP(Payments$_1$,Payments$_2$) = {
(INVOICES_PAYMENTS$_1$, INVOICE_NUMBER_INSTANCE$_2$) }
- MDP(Products$_1$, Products$_2$) = { (PRODUCT_ID$_1$, PRODUCT_ID$_2$),
(PRODUCT_NAME$_1$, PRODUCT_NAME$_2$),
(PRODUCT_PRICE$_1$, PRODUCT_PRICE$_2$),
(PRODUCT_COLOR$_1$, PRODUCT_COLOR$_2$),
(PRODUCT_SIZE$_1$, PRODUCT_SIZE$_2$),
(PRODUCT_DESCRIPTION$_1$, PRODUCT_DESCRIPTION$_2$),
(OTHER_PRODUCT_DETAILS$_1$, OTHER_PRODUCT_DETAILS$_2$) }
- MOP(Products$_1$, Products$_2$) = { (PRODUCTS_ORDER_ITEMS.Inverse$_1$, null),
(REF_PRODUCT_TYPES_PRODUCTS$_1$,
PRODUCT_TYPE_CODE_INSTANCE$_2$) }
- MDP(Ref_Invoice_Status_Codes$_1$, Ref_Invoice_Status_Codes$_2$) = {
(INVOICE_STATUS_CODE$_1$, INVOICE_STATUS_CODE$_2$),
(INVOICE_STATUS_DESCRIPTION$_1$,
INVOICE_STATUS_DESCRIPTION$_2$) }
- MOP(Ref_Invoice_Status_Codes$_1$, Ref_Invoice_Status_Codes$_2$) = {
(INVOICE_STATUS_CODES_INVOICES.Inverse$_1$, null)}
- MDP(Ref_Order_Item_Status_Codes$_1$, Ref_Order__Item_Status_Codes$_2$) = {
(ORDER_ITEM_STATUS_CODE$_1$, ORDER_ITEM_STATUS_CODE$_2$),
(ORDER_ITEM_STATUS_DESCRIPTION$_1$,

ORDER_ITEM_STATUS_DESCRIPTION$_2$) }

- MOP(Ref_Order_Item_Status_Codes$_1$, Ref_Order__Item_Status_Codes$_2$) = {
  (ORDER_ITEM_STATUS_ORDER_ITEMS.Inverse$_1$, null) }
- MDP(Ref_Order_Status_Codes$_1$, Ref_Order_Status_Codes$_2$) = {
  (ORDER_STATUS_CODE$_1$, ORDER_STATUS_CODE$_2$),
  (ORDER_STATUS_DESCRIPTION$_1$, ORDER_STATUS_DESCRIPTION$_2$) }
- MOP(Ref_Order_Status_Codes$_1$, Ref_Order_Status_Codes$_2$) = {
  (ORDER_STATUS_CODES_ORDERS.Inverse$_1$, null) }
- MDP(Ref_Payment_Methods$_1$, Ref_Payment_Methods$_2$) = {
  (PAYMENT_METHOD_CODE$_1$, PAYMENT_METHOD_CODE$_2$),
  (PAYMENT_METHOD_DESCRIPTION$_1$,
  PAYMENT_METHOD_DESCRIPTION$_2$) }
- MOP(Ref_Payment_Methods$_1$, Ref_Payment_Methods$_2$) = {
  (REF_PAYMENT_METHODS_CUSTOMER_PAYMENT_METHODS.Inverse$_1$,
  null) }
- MDP(Ref_Product_Types$_1$, Ref_Product_Types$_2$) = {
  (PRODUCT_TYPE_CODE$_1$, PRODUCT_TYPE_CODE$_2$),
  (PRODUCT_TYPE_DESCRIPTION$_1$, PRODUCT_TYPE_DESCRIPTION$_2$) }
- MOP(Ref_Product_Types$_1$, Ref_Product_Types$_2$) = {
  (REF_PRODUCT_TYPES_PRODUCTS.Inverse$_1$, null),
  (REF_PRODUCT_TYPES_REF_PRODUCT_TYPES$_1$,
  PARENT_PRODUCT_TYPE_CODE_INSTANCE$_2$),
  (REF_PRODUCT_TYPES_REF_PRODUCT_TYPES.Inverse$_1$, null) }
- MDP(Shipments$_1$, Shipments$_2$) = { (SHIPMENT_ID$_1$, SHIPMENT_ID$_2$),
  (SHIPMENT_TRACKING_NUMBER$_1$,
  SHIPMENT_TRACKING_NUMBER$_2$),
  (SHIPMENT_DATE$_1$, SHIPMENT_DATE$_2$),
  (OTHER_SHIPMENT_DETAILS$_1$, OTHER_SHIPMENT_DETAILS$_2$) }
- MOP(Shipments$_1$, Shipments$_2$) = {
  (INVOICES_SHIPMENTS$_1$, INVOICE_NUMBER_INSTANCE$_2$),
  (ORDERS_SHIPMENTS$_1$, ORDER_ID_INSTANCE$_2$)
  (ORDER_ITEMS_SHIPMENT_ITEMS$_1$, null) }

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Agrawal, Rakesh et al. 1989. Efficient management of transitive relationships in large data and knowledge bases. ACM.

[2] Albarrak, Khalid and Sibley, Edgar. 2009. Translating relational & object-relational database models into OWL models. Proceedings of the 10th IEEE international conference on Information Reuse & Integration (IRI 2009). Las Vegas, Nevada, USA. August 10-12, 2009.

[3] Albarrak, Khalid and Sibley, Edgar. 2010. An Extensible Framework for Generating Ontology Models from Data Models. Journal of the International Transactions on Systems Science and Applications (ITSSA), Vol. 6, No. 2/3, August 2010, pp. 97-112.

[4] Albarrak, Khalid and Sibley, Edgar. 2011. A survey of methods that transform data models into Ontology models. Proceedings of the 12th IEEE international conference on Information Reuse & Integration (IRI 2011). Las Vegas, Nevada, USA. August 3-5, 2011.

[5] Albarrak, Khalid and Sibley, Edgar. 2012. Measuring Expressivity between Ontology Models. Proceedings of the 11th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases (AIKED '12).

[6] Alhajj, Reda. 2003. Extracting the extended entity-relationship model from a legacy relational database. Information Systems, v.28 n.6, p.597-618. Elsevier Science.

[7] ANSI/ISO/IEC International Standard. 1999. ISO/IEC 9075-2:1999 - Database Language SQL - Part 2: Foundation (SQL/Foundation). September 1999.

[8] Antoniou, Grigoris and Van Harmelen, Frank. 2008. A Semantic Web Premier, 2nd Edition. The MIT Press.

[9] Astrova, Irina. 2004. Reverse Engineering of Relational Databases to Ontologies. The Semantic Web: Research and Applications; Volume 3053/2004. Springer Berlin/Heidelberg.

[10] Astrova, Irina. 2005. Toward the Semantic Web - An Approach to Reverse Engineering of Relational Databases to Ontologies. Communications of the Ninth East-European Conference on Advances in Databases and Information Systems, ADBIS-2005. Tallinn, Estonia, September 12-15, 2005. pp. 111-122

[11] Astrova, Irina. 2008. Rules for Mapping SQL Relational Databases to OWL Ontologies. Metadata and Semantics. Springer US.

[12] Baeza-Yates, Ricardo and Ribeiro-Neto, Berthier. 1999. *Modern Information Retrieval*. ACM Press, New York, 1999.

[13] Blaha, Michael. 2010. Patterns of Data Modeling. CRC Press.

[14] Bohring, Hannes and Auer, Sören 2005. Mapping XML to OWL Ontologies. Leipziger Informatik-Tage, volume 72 of LNI (2005), pp. 147-156

[15] Brank, Janez et al. 2005. A Survey of Ontology Evaluation Techniques. In Proc. Of the Conf. on Data Mining and Data Warehouses (SiKDD 2005), Ljubljana, Slovenia

[16] Brewster, Christopher et al. 2004. Data Driven Ontology Evaluation. Proceedings of Int. Conf. on Language Resources and Evaluation, Lisbon, Portugal.

[17] Bruijn, Jos de. 2003. Enabling Knowledge Sharing and Reuse on the Semantic Web. Digital Enterprise Research Institute (DERI) Technical Report DERI-2003-10-29.

[18] Buccella, Agustina et al. 2004. From Relational Databases to OWL Ontologies. Proceeding of the 6th National Russian Conference on Digital Libraries (RCDL), Pushchino, Russia, September 29th - October 1st, 2004.

[19] Burgun, Anita. 2006. Desiderata for domain reference ontologies in biomedicine. Journal of Biomedical Informatics, 39 (2006) 307-313. Elsevie

[20] Burton-Jones, Andrew et al. 2005. A semiotic Metrics Suite for Assessing the Quality of Ontologies. Data Knowledge & Engineering 55 (2005) 84-102. Elsevier.

[21] CA, Inc. Database Design & Modeling; CA ERwin Data Modeler. http://www.ca.com/us/database-design (accessed March 2011).

[22] Cardoso, Jorge. 2007. Semantic Web Services: Theory, Tools and Applications. Idea Group. e-Book:978-1-59904-047-9.

[23] Castano, S. et al. 1998. Conceptual Schema Analysis- Techniques and Applications. ACM Transactions on Database Systems (TODS). Volume 23, Issue 3, September 1998.

[24] Cerbah, Farid. 2008. Mining the Content of Relational Databases to Learn Ontologies with Deeper Taxonomies. IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI'08 & IAT'08). Sydney, Australia. December 9-12, 2008.

[25] Cerbah, Farid. 2008. Learning Highly Structured Semantic Repositories from Relational Databases: The RDBToOnto Tool. The Semantic Web: Research and Applications, Volume 5021/2008, pp 777-781. Springer Berlin/Heidelberg.

[26] Chen, Jia et al. 2009. Rules Driven Object-Relational Databases Ontology Learning. IEEE.

[27] Chiang, Roger H.L. et al. 1994. Reverse engineering of relational databases: extraction of an EER model from a relational database. Data Knowledge Eng. 12, 2 (Mar. 1994). Elsevier Science.

[28] Cullot, Nadine et al. 2007. DB2OWL: A Tool for Automatic Database-to-Ontology Mapping. In Proceedings of the 15th Italian Symposium on Advanced Database Systems (SEBD 2007). Torre Canne di Fasano (BR), Italy, June 2007.

[29] DAML. 2003. About the DAML Language. http://www.daml.org/about.html (accessed on Dec. 22, 2011)

[30] Date, Chris J. 1995. Introduction to Database Systems, 6th edition. Addison-Wesley

[31] Date, Chris J. 2003. On Various Types of Relations. http://www.dbdebunk.com/page/page/622108.htm (accessed on Dec 5th, 2011)

[32] Dey, Debabrata et al. 1999. Improving database design through the analysis of relationships. ACM Transactions on Database Systems (TODS), Volume 24 Issue 4, Dec. 1999.

[33] Doan, AnHai and Halevy, Alon. 2005. Semantic Integration Research in the Database Community: A Brief Survey. AI Magazine

[34] Doan, AnHai. 2002. Learning to Map between Structured Representations of Data. PhD Thesis. Chapters 1 and 5. http://pages.cs.wisc.edu/~anhai/thesis/anhai-thesis.pdf (accessed on April 28, 2010).

[35] Euzenat, Jerome and Valtchev, Petko. 2004. Similarity-based Ontology Alignment in OWL-Lite. In The 16th European Conf. on Artificial Intelligence (ECAI-04), Valencia, Spain.

[36] Fensel, Dieter et al. 2001. OIL: An Ontology Infrastructure for the Semantic Web. IEEE Intelligent Systems, March/April 2001, Vol. 16, no. 2.

[37] Ferdinand, Matthias et al. 2004. Lifting XML Schema to OWL. Web Engineering. Volume 3140/2004. Springer Berlin / Heidelberg.

[38] Fisher, Maydene, Jon Ellis and Jonathan Bruce. 2004. JDBC API Tutorial and Reference, Third Edition. Addison-Wesley. Chapter 1 and 2 (Pages 3-112)

[39] Fonseca, Frederico and Martin, James. 2007. Learning the Differences Between Ontologies and Conceptual Schemas Through Ontology-Driven Information Systems. Journal of the Association for Information Systems (JAIS); Special Issue on Ontologies in the Context of IS. Volume 8, Issue 2. pp. 129–142,

[40] Formica, Anna. 2009. Concept Similarity by Evaluating Information Contents and Feature Vectors: A Combined Approach. Communication of the ACM. Volume 52, Issue 3, March 2009.

[41] Ghawi, Raji and Cullot, Nadine. 2007. Database-to-Ontology Mapping Generation for Semantic Interoperability. The 3rd International Workshop on Database Interoperability (InterDB); Held in conjunction with VLDB 2007. Vienna, Austria. September 24, 2007.

[42] Gillenson, Mark L. et al. 2007. Introduction to Database Management. Wiley.

[43] Gruber, Thomas R. 1993. A Translation Approach to Portable Ontology. Specifications. Knowledge Acquisition, Vol. 5, Issue 2, June 1993, pp 199-220.

[44] Guarino, Nicola and Giaretta, Pierdaniele. 1995. Ontologies and Knowledge Bases: Towards a Terminological Clarification. In Towards Very Large Knowledge Bases. N. J. I. Mars, Ed., IOS Press: 25-32

[45] Halpin, Terry. 2008. Information Modeling and Relational Databases, Second Edition. Morgan Kaufmann Publishers. Books24x7 version.

[46] He-ping, Chen et al. 2008. Research and Implementation of Ontology Automatic Construction Based on Relational Database. IEEE; International Conference on Computer Science and Software Engineering. Wuhan, Hubei, China, December 12-14, 2008.

[47] Hu, Changjun et al. 2008. Research and Implementation of Domain-Specific Ontology Building from Relational Database. IEEE; The Third ChinaGrid Annual Conference 2008 (ChinaGrid '08). August 20-22, 2008.

[48] IBM. DB2 for Linux, UNIX, and Windows – The SAMPLE database. http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.apdv.samptop.doc/doc/r0001094.html (accessed on Sept. 26, 2011).

[49] IBM. IBM InfoSphere Data Architect. http://www-01.ibm.com/software/data/optim/data-architect/ (accessed March 2011)

[50] ISRD Group. 2006. Introduction to Database Management Systems. Tata McGraw-Hill.

[51] Kalfoglou, Yannis and Schorlemmer, Marco. 2003. Ontology mapping: the state of the art. The knowledge Engineering Review, Vol. 18:1, 1-31. Cambridge Univ Press.

[52] Laborda, Cristian and Conrad, Stefan. 2005. Relational.OWL - A Data and Schema Representation Format Based on OWL. ACM; Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling - Volume 43; 2005; Newcastle, New South Wales, Australia. pp. 89-96

[53] Levenshtein, V. I. 1966. Binary Codes capable of correcting deletions, insertions, and reversals. Cybernetics and Control Theory. Soviet Physics-Doklady. Vol 10, No. 8. 10(8):707–710, 1966.

[54] Li, Man et al. 2005. Learning Ontology From Relational Database. IEEE; Proceedings of the 4th International Conference on Machine Learning and Cybernetics. Guangzhou, China, August 18-21, 2005.

[55] Li, Man et al. 2005. A Semi-automatic Ontology Acquisition Method for the Semantic Web. Advances in Web-Age Information Management; Volume 3739/2005. Springer Berlin / Heidelberg.

[56] Lin, Dekang. 1998. An information-theoretic definition of similarity. Proceedings of the 15th International Conference on Machine Learning (ICML), Madison, Wisconsin, USA. July 24–27, 1998. Morgan Kaufmann. pp. 296–304.

[57] Lubyte, Lina and Tessaris, Sergio. 2007. Extracting Ontologies from Relational Databases. Technical Report, KRDB group – Free University of Bozen-Bolzano. www.inf.unibz.it/krdb/pub/TR/KRDB07-4.pdf (accessed March 2011).

[58] Lubyte, Lina. 2007. Reusing Relational Sources for Semantic Information Access. Proceedings of the ACM first Ph.D. workshop in Conference on Information and Knowledge Management (CIKM), Lisbon, Portugal, 9/11/2007, pp. 9-16

[59] Maedche, Alexander and Staab, Steffen. 2002. Measuring Similarity between Ontologies. Proc CIKM 2-2. LNAI vol 2473.

[60] Melton, Jim. 2003. Advanced SQL: 1999 - Understanding Object-Relational and Other Advanced Features. Morgan Kaufmann.

[61] Microsoft. SQL Server – Starter Database Schemas. http://msdn.microsoft.com/en-us/express/bb403186#5 (accessed on Sept. 26, 2011).

[62] Mukhopadhyay, Debajyoti et al. 2007. A Technique for Automatic Construction of Ontology from Existing Database to Facilitate Semantic Web. IEEE; 10th International Conference on Information Technology (ICIT 2007). Rourkela, India, December 17-20, 2007.

[63] Noy, Natalya F. and McGuinness Deborah L. 2001. Ontology Development 101: A Guide to Creating Your First Ontology. Online. Available at http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness.pdf (accessed March 29, 2009)

[64] Noy, Natalya. 2004. Sematic Integration: A Survey of Ontology-Based Approaches. SIGMOD Vol 33, No. 4. ACM.

[65] Nyulas, Csongor et al. 2007. DataMaster – a Plug-in for Importing Schemas and Data from Relational Databases into Protégé. 10th International Protégé Conference. Budapest, Hungary, July 15-18, 2007.

[66] Obrst, Leo et al. 2007. The Evaluation of Ontologies - Toward Improved Semantic Interoperability. Semantic Web, Part II, 139-158, Springer US.

[67] Park, Jinsoo et al. 2010. Evaluating Ontology Extraction Tools Using a Comprehinsive Evaluation Framework. Data Knowledge & Engineering 69 (2010) 1043-1061. Elsevier.

[68] Pfeiffer, Daniel and Gehlert, Andreas. 2005. A framework for comparing conceptual models. Proceedings of the Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 2005), Klagenfurt, Austria. pp. 108-122.

[69] Princeton University. WordNet Search - 3.0. http://wordnet.princeton.edu/perl/webwn?s=chair (accessed on Sept. 1st, 2008)

[70] Qu, Z. and Tang, S. 2008. Research on Transforming Relational Database into Enriched Ontology. International Conference on Advanced Computer Theory and Engineering (ICACTE), Phuket, Thailand, 20-22/12/2008, pp. 749-753.

[71] Rahimi, Saeed K. and Haug, Frank S. 2010. Distributed Database Management Systems; A Practical Approach. Wiley.

[72] Ramakrishnan, Raghu and Gehrke, Johannes. 2003. Database Management Systems, 3ed edition. McGraw-Hill.

[73] Ross, Kenneth A and Stoyanovich, Julia. 2004. Symmetric Relations and Cardinality-Bounded Multisets in Database Systems. The 30th VLDB Conference, Toronto, Canada, 2004.

[74] Rzhetsky A, Evans JA. 2011. War of Ontology Worlds: Mathematics, Computer Code, or Esperanto? PLoS Computational Biology 7(9): e1002191. doi:10.1371/journal.pcbi.1002191

[75] Shanks, Graeme et al. 2003. Using ontology to validate conceptual models. Communications of the ACM. Volume 46, Issue 10, October 2003.

[76] Sheu, Phillip et al. 2010. Semantic Computing. Wiley-IEEE Press. pp 1-10

[77] Sheu, Phillip. 2010. Semantic Computing and Semantic Web. Institute of Semantic Computing. http://isc2010.wordpress.com/2010/03/05/semantic-computing-and-semantic-web/ (accessed on Dec 26th, 2011).

[78] Silverston, Len and Agnew, Paul. 2009. The Data Model Resource Book: Universal Patterns for Data Modeling, Volume 3. John Wiley & Sons

[79] Simsion, Graeme C. & Witt, Graham C. 2005. Data Modeling Essentials, 3ed edition. Morgan Kaufmann Publishers; Books24x7 version.

[80] Stojanovic, Ljiljana et al. 2002 - Migrating data-intensive Web Sites into the Semantic Web. ACM; Proceedings of the 2002 ACM Symposium on Applied Computing. Madrid, Spain, March 11-14, 2002. pp. 1100-1107

[81] Stuckenschmidt, Heiner and Van Harmelen, Frank. 2005. Information Sharing on the Semantic Web; Advanced Information and Knowledge Processing. Springer.

[82] Sugumaran, Vijayan and Storey, Veda. 2006. The role of domain ontologies in database design: An ontology management and conceptual modeling environment. ACM Transactions on Database Systems (TODS). Volume 31, Issue 3, September 2006.

[83] Thuy, Pham Thi Thu et al. 2008. Exploiting XML Schema for Interpreting XML Document as RDF. 2008 IEEE International Conference on Services Computing (SCC 2008). Honolulu, Hawaii, USA, July 8-11, 2008.

[84] Trinh, Quang et al. 2006. RDB2ONT - A Tool for Generating OWL Ontologies From Relational Database Systems. IEEE; Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW 2006). Guadeloupe, French Caribbean, February 19-25, 2006. pp. 170

[85] Trinkunas, Justas and Vasilecas, Olegas 2007. Building ontologies from relational databases using reverse engineering methods. ACM; In Proceedings of the 2007 international Conference on Computer Systems and Technologies. Bulgaria, June 14-15, 2007.

[86] Tsinaraki, Chrisa and Christodoulakis, Stavros. 2007. XS2OWL: A Formal Model and a System for Enabling XML Schema Applications to Interoperate with OWL-DL Domain Knowledge and Semantic Web Tools. Digital Libraries: Research and Development. Volume 4877/2007. Springer Berlin / Heidelberg.

[87] Vysniauskas, Ernestas and Nemuraite, Lina. 2006. Transforming Ontology Representation from OWL to Relational Database. Informaion technology and Control. Vol. 35, No 3A.

[88] W3C, OWL Web Ontology Language, Guide, W3C Recommendation 10 February 2004. ed. Michael Smith et al. http://www.w3.org/TR/owl-guide/

[89] W3C, OWL Web Ontology Language, Overview, W3C Recommendation 10 February 2004. ed. D McGuinness and F van Harmelen. http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.3

[90] W3C. RDF Validation Service. Eric Prud'hommeaux. http://www.w3.org/RDF/Validator/. Accessed on July 9th, 2011.

[91] W3C. W3C Semantic Web Activity http://www.w3.org/2001/sw and http://www.w3.org/2001/sw/Activity (accessed on September 1st, 2008)

[92] Wache, H., et al. 2001. Ontology-based Integration of Information - A Survey of Existing Approaches. In Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing, Seattle, WA.

[93] Wang, HongSheng et al. 2008. Text Information Extraction Based on OWL Ontologies. IEEE; Fifth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD '08). Jinan, Shandong, China, October 18-20, 2008.

[94] Weiss, Gerhard (ed), Gerhard, Michael. 1999. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence; Chapter 1 - Intelligent Agents. The MIT Press. Books24x7

[95] Wolfram Research. Mathematica 8. http://www.wolfram.com/mathematica/ (accessed on Dec 5th, 2011)

[96] WonderWeb OWL Ontology Validator. http://www.mygrid.org.uk/OWL/Validator. © University of Manchester, 2003 & © University of Karlsruhe, 2003. Accessed on July 9th, 2011.

[97] Xu, Jiuyun and Li, Weichong. 2007. Using Relational Database to Build OWL Ontology from XML Data Sources. IEEE; International Conference on Computational Intelligence and Security Workshops (CISW 2007). December 15-19, 2007.

[98] Yan, Luo and Changrui, Yu. 2007. Development Method of Domain Ontology Based on Reverse Engineering. IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI 2007). Philadelphia, PA, USA, Aug. 27-29, 2007.

[99] Zhao, Shuxin and Chang, Elizabeth. 2007. From Database to Semantic Web Ontology: An Overview. On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops. Volume 4806/2007. Springer Berlin / Heidelberg.

BIOGRAPHY

Khalid Al-Barrak received his Bachelor of Science in Computer Information Science from King Saud University, Riyadh, Saudi Arabia. Three years later, he relocated to the United States to pursue higher education. In 2000, Khalid obtained his Master of Science in Computer Science from California State University, Chico, California.

Khalid has over 16 years of experience in the IT field, 13 of which is with IBM Corporation. In 2000, he joined IBM's leading database and software lab, Santa Teresa Lab, as a Software Engineer. Four years later, Khalid transitioned to a solution architecture role serving as an IBM World-Wide (WW) Information Integration subject matter-expert for various IBM technologies where he was brought in to solve integration challenges facing Fortune-100 US companies, government entities and foreign companies. Most recently, Khalid joined the Channels Sales organization where he supports IBM business partners.