A Framework for Information Integration using

Ontological Foundations

by

© Yoones Asgharzadehsekhavat

A Thesis submitted to the

School of Graduate Studies

in partial fulfillment of the requirements for the degree of

Ph.D.

Department of Computer Science

Memorial University of Newfoundland

May 2014

St. John’s Newfoundland

ABSTRACT

With the increasing amount of data, ability to integrate information has always been a competitive advantage in information management. Semantic heterogeneity reconciliation is an important challenge of many information interoperability applications such as data exchange and data integration. In spite of a large amount of research in this area, the lack of theoretical foundations behind semantic heterogeneity reconciliation techniques has resulted in many ad-hoc approaches. In this thesis, I address this issue by providing ontological foundations for semantic heterogeneity reconciliation in information integration. In particular, I investigate fundamental semantic relations between properties from an ontological point of view and show how one of the basic and natural relations between properties – inferring implicit properties from existing properties – can be used to enhance information integration. These ontological foundations have been exploited in four aspects of information integration. First, I propose novel algorithms for semantic enrichment of schema mappings. Second, using correspondences between similar properties at different levels of abstraction, I propose a configurable data integration system, in which query rewriting techniques allows the tradeoff between accuracy and completeness in query answering. Third, to keep the semantics in data exchange, I propose an entity preserving data exchange approach that reflects source entities in the target independent of classification of entities. Finally, to improve the efficiency of the data exchange approach proposed in this thesis, I propose an extended model of the column-store model called sliced column store. Working prototypes of the techniques proposed in this thesis are implemented to show the feasibility of realizing these

techniques. Experiments that have been performed using various datasets show the techniques proposed in this thesis outperform many existing techniques in terms of ability to handle semantic heterogeneities and performance of information exchange.

ACKNOWLEDGEMENTS

This thesis could not be possible without help of a number of people. First and foremost, I would like to extend thanks to my supervisor, Prof. Jeffrey Parsons for continuous academic and support of my study and research. I really appreciate his insights and comments, motivation, immense knowledge, and expertise that added considerably to this thesis. What I have accomplished academically as a doctoral student would not have happened without him and his guidance. He provided me with encouragement, advice and sincere support I needed to complete my doctoral studies. He taught me how to rationally think big, and follow big goals. Jeff has been a caring and supportive advisor who has always believed in me, allowing me to pursue independent work. I also acknowledge his financial support which helped me pursue my graduate studies more easily.

I also would like to thank my supervisory committee members Dr. Jian Tang, and Dr. Orland Hoeber for their meticulous review of my thesis. In particular, I appreciate Dr. Hoeber's great care and insightful comments on my work. I enjoyed and learned very much working with him during a joint project. I also thank Dr. Renée Miller for providing test schemas for the experiments.

Appreciation goes out to the staff and faculty of the Computer Science who offered me assistance in many levels of my graduate studies. Special thanks go out to Dr. Adrian Fiech and Dr. Jim Wyse. Adrian's enthusiasm in holding social gatherings helped boost the sense of community among students and faculty members. Jim has been a very supportive teacher and a very good friend to me. I experienced Newfoundland boating with him, and I will never forget his kindness. I would like to thank Darlene Oliver,

Brenda Hiller, Elaine Boone, Nolan White, and Paul Price for their help and support during completion of my program. I also acknowledge the financial contributions of the School of Graduate Studies of Memorial University which supported my graduate studies.

Thanks to the fellow students in the department of Computer Science for their friendship and support. I will never forget soccer games and badminton tenements through which I have found many good friends. I also thank their comments and constructive criticism during the presentation of some parts of my research.

I express my greatest gratitude to my family, my parents, Parvin and Hasan, my brothers, Mohamad Reza and Hadi. I thank them for offering me their endless love and support through my entire life.

Finally, special recognition goes to my wife, Saeedeh, who inspired me and provided constant encouragement during the entire process. There are no words to convey how much her unconditional love has helped me to not only survive, but also enjoy this journey. I also thank her for editing my thesis.

Table of Contents

List of Tables

List of Figures

List of Symbols, Nomenclature or Abbreviations

HL7 - Health Level 7

SESM - Semantic Enrichment of Schema Mappings

CDI - Configurable Data Integration

EDEX - Entity Preserving Data Exchange

HSRC - Health System Resident Component

PHIN - Public Health Information Network

NEDSS - National Electronic Disease Surveillance System

Exp-Spc - Query expansion through specialization

Exp-Gen - Query expansion through generalization

Co-Authorship Statement

I have conducted the research project of this thesis under supervision of Dr. Jeffrey Parsons. His comments have helped me to revise the ideas and improve the quality of this thesis. In the following, I elaborate my role in the preparation of this thesis.

Design and identification of the research proposal: I have proposed the research topic of this thesis after consultation with Dr. Jeffrey Parsons. Early meetings with him and discussing theoretical foundations in information systems have inspired me to propose the main ideas. Although the general topic was constrained to deal with ontological foundations, I independently decided to research ontological foundations for data integration and data exchange.

Practical aspects of the research: The practical aspects of the research including literature review, proposing ideas, design of prototypes, development of prototypes, and performing experiments were undertaken by me. In particular, I have designed, developed and tested the working prototypes. Dr. Parsons helped me to design and select evaluation techniques in various parts of the thesis. However, I have designed test cases and experiments. In addition, I have independently conducted the experiments.

Data analysis: I analyzed the primary results of the experiments. Then, initial results were investigated in meetings with Dr. Parsons to revise the experiments and prototypes. I have conducted the final interpretation of the results and evaluating the outcomes.

Manuscript preparation: I have been the primary author of this thesis with Dr. Parsons being co-author. I wrote a first draft of each paper (chapter). Then, the papers were

revised in an interactive process based on reviews and comments of Dr. Parsons. He has assisted in improving the quality of this manuscript.

Chapter 1 Introduction and Overview

1.1 Context and Problem

The evolution of the Internet is driving data volume growth at an increasing rate. According to (Beath et al., 2012), the volume of data is expanding annually around 50 percent. Distributing this data over different sources for security, efficiency, reliability, or other purposes has resulted in creating islands of data. Such islands of data form a heterogeneous environment in which each data island may hold only part of the data that is required to fulfill information requests. Although the rapid increase of data is considered as one of the main reasons for difficulty in finding information, data fragmentation and existence of data in different heterogeneous data sources is a deeper problem (Haas, 2006)

Being able to manage and cope with distributed data requires that data be integrated for decision making and analysis. One possible option to deal with data heterogeneity is an *a priori* approach in which database designers follow existing standards in a domain for data modeling. However, as discussed in (Hentschel et al., 2009), even if such standards are used, diversity and heterogeneity still exist because data schemas are developed independently by different people in different contexts. An alternative solution can be accepting the existence of data heterogeneity and trying to reconcile it through information integration techniques. Information integration is a solution for fulfilling information requests in a heterogeneous environment by establishing connections among data islands. According to Ventana research (San, 2012),

more than 57 percent of organizations integrate six or more data sources, which is expected to reach 68 percent by the end of 2014. Information integration can be performed through *data exchange*, in which data is combined and integrated in a single data source (Fagin et al., 2005), or *data integration*, in which a query interface works as a mediator to access data while data remains in the sources. Whether data integration or data exchange is used, information integration is generally performed based on schema mappings. Such mappings represent high level relationship between data sources independent of implementation details (Bonifati et al., 2011). Generating mappings is considered the first step in information integration and requires a considerable effort (Halevy, 2010; Doan et al., 2012). A simple search on a travel website such as Expedia requires integrating tens of data sources powered by schema mapping as the glue to tie these data sources together.

Generally, information integration is performed in a heterogeneous environment involving several data sources that are designed and maintained independently by different parties. In such an environment, local users of data sources are only aware of the dictionary of the database to which they pose a query. As a result, due to heterogeneities among different data sources, the terms that a user employs to build a query may not match the terms in other data sources. Moreover, there might be values in the database that seem different but are semantically equivalent to the user terms and express the same intention of the user. To deal with this heterogeneity, considering the semantics of information content is necessary. Inability to reconcile semantic heterogeneity results in improper interpretation of concepts, terminology, and metadata, that ultimately hinders

proper communication among different information sources (Arch-Int et al., 2003). In spite of a large amount of research on information integration, semantic heterogeneity reconciliation is not adequately addressed. In this thesis I identify and address several problems in existing information integration techniques that are consequences of ignoring semantics.

First, many existing schema mapping techniques ignore the semantics of relations, which can result in many ambiguous scenarios (Bonifati et al., 2005; Marnette et al., 2011; Popa et al., 2002; ten Cate et al., 2013). In such ambiguous scenarios, it is not possible to decide which mappings should be employed to integrate data solely based on metadata. In Chapter 3, I argue that since a relational model is not expressive enough to show the semantics originally contained in a conceptual model, not all semantics are preserved in translating a conceptual model to a relational model. I contend that without high level knowledge to reconcile semantic heterogeneities, it is not possible to identify the semantics behind different types of associations in schema mapping.

Second, past research on schema mapping has largely dealt with the case where simple property correspondences between properties (representing equivalent properties) are used to generate mappings. In Chapter 4, I show why such correspondences between similar properties are not able to handle complex semantic heterogeneities. I argue that other relationships between properties and the ability to infer a property from a completely different property are largely ignored in previous work. Using indirect relationships between properties and inferring implicit properties from existing explicit

properties are key points in binding two heterogeneous data sources that have not been well studied in information integration.

Third, existing data integration techniques employ query rewriting algorithms that are lossless and consist of statements asserting that some portion of the data is equal to some other portion of the data. This approach may prevent full data integration because it ignores bindings between two similar properties that are presented at different levels of abstraction. Consequently, some potential answers that may satisfy the original query are ignored. However, an intelligent data integration system must be able to find those potential answers in addition to exact answers even if this compromises accuracy. This problem has led this thesis to a new information integration approach that provides tradeoff between accuracy and completeness in information integration. This does not mean finding exact answers is less important than finding additional, possibly incorrect, answers. Rather, I argue that this approach allows discovering some potential answers that can be pruned to find accurate answers.

Forth, one of the main problems of schema mapping based data exchange is that this approach only deals with schema level relations between source and target schemas. In spite of independent progress in schema level and data level approaches for data exchange, semantic heterogeneities are not completely resolved because of the gap between these two approaches (Haas et al., 2009). I argue that such semantic heterogeneities may result in ambiguous scenarios in schema mappings and consequently improper data exchange. In this thesis, I attribute this problem to confining schema mapping expressions in class definitions.

In this thesis I argue that, due to problems such as these, information integration has witnessed limited progress in practice. According to Ventana's research (San, 2012) in data integration, less than 15 percent of organizations who perform information integration activities use formal and automated data integration techniques. As discussed in (Haas, 2006), we lack "a deep understanding of what fundamental operations are needed to integrate information" and we are not sure if current operations for data integration through query processing are complete and precise. The lack of theoretical foundation behind semantic reconciliation techniques has been cited as the main reason for limited progress in this area (Parsons & Wand, 2003).

In this thesis, I turn to ontology to address the problem of the lack of theoretical foundations for semantic heterogeneity reconciliation. In particular, I use some fundamental notions from Mario Bunge's ontology (Bunge, 1977) for this purpose. The rationale for choosing this ontology is that it deals with systems in general, it is a comprehensive ontology covering many other ontologies, and it is well-formalized in terms of set-theory. In addition, Bunge's ontology has been widely considered as a theoretical foundation for conceptual modeling both in theoretical analysis (Parsons & Wand, 2000; Wand et al., 1999) and in empirical studies (Gemino & Wand, 2004; Parsons, 2011; Shanks et al., 2002).

1.2 Research Questions

In this thesis, the following primary research questions are addressed:

a) What are the root problems of existing information integration techniques?

b) How can ontological foundations be used to enrich semantic heterogeneity reconciliation techniques?
c) What does it mean to accurately or completely answer a query given semantically enriched relations among data items?
d) How is it possible to address the problem of ambiguous scenarios in data exchange and data integration? Can ontological foundations be used to address these problems?
e) How can we improve the performance of data exchange processes using ontological foundations?

1.3 Research objectives

The goal of this thesis is to propose an information integration framework based on ontological foundations. To attain this goal, the specific research objectives of this thesis are:

a) Study deficiencies of the current data integration and data exchange techniques to handle semantic heterogeneities.
b) Propose ontological foundations for semantic heterogeneity reconciliation.
c) Design and development of a novel data integration system that exploits rich semantic knowledge to improve the quality of data integration.
d) Design and development of a semantically enriched data exchange system to address ambiguous mappings in data exchange.
e) Improve the performance of data exchange based on ontological foundations.
f) Comparison of the proposed techniques with common existing techniques.

1.4 Research Methods

An overview of the research methodology employed in this thesis is summarized in Figure 1-1. The research started with an extensive literature review in information integration. The thesis has been conducted in three main phases including theoretical analysis of the problems in information integration, constructing prototypes, and experiments. The research methodology shown in Figure 1-1 is an incremental model that unlike many sequential models (e.g., waterfall method) allows developing the system step by step while mitigates the risk of failure by developing prototypes.

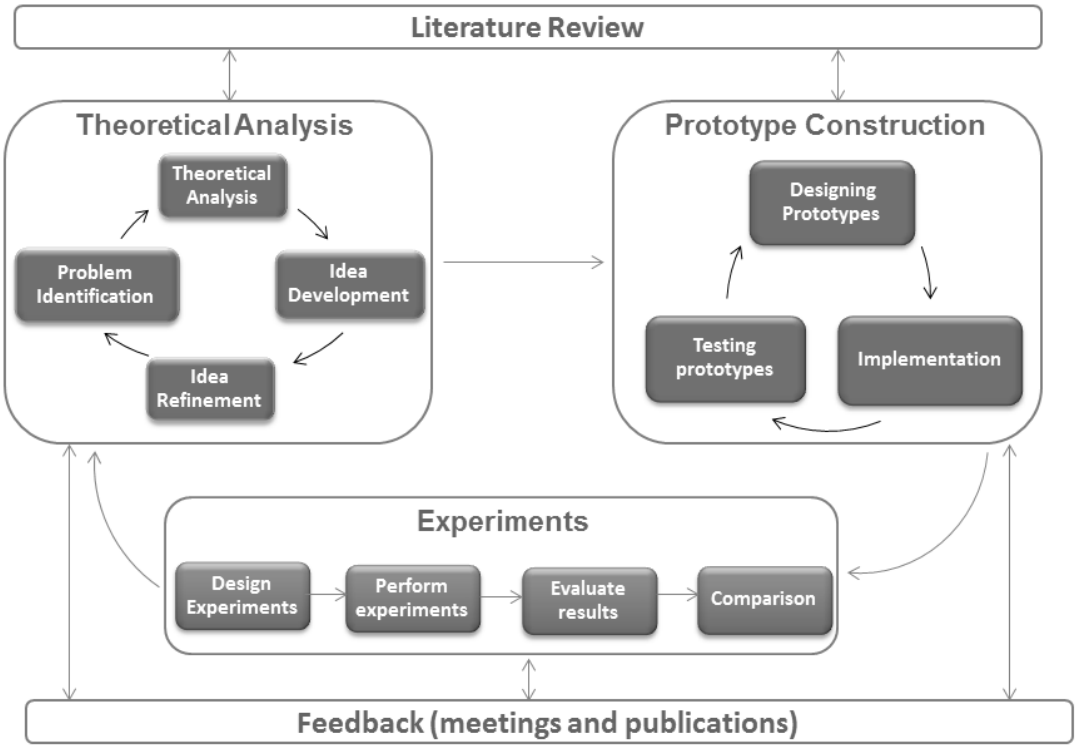


Figure 1-1: The overview of the research methodology employed in this thesis

In the phase of theoretical analysis, the current problems of information integration techniques were studied. This analysis performed with the goal of finding root

problems of semantic heterogeneity reconciliation in information integration to address the problem of the lack of theoretical foundations for semantic heterogeneity reconciliation. In particular, we used some basic notions from Mario Bunge's ontology (Bunge, 1977). Using these foundations guided the proposition of new approaches in information integration that are compatible with the basis of this ontology.

In the phase of prototype construction, working prototypes of each of various components of the information integration framework were implemented. components were developed feature by feature, and then were tested and evaluated using synthesized data. In the experimental phase, different real-world datasets from a variety of domains as well as some synthesized datasets were used. For each component, a comparison between existing techniques and the technique proposed in this thesis was conducted.

1.5 Thesis Organization

This thesis is presented in manuscript format. Specifically, Chapter 2 provides a comprehensive review of the literature in information integration. Chapter 3 is a paper published in ICDE Workshop on Data Engineering Meets Semantic Web (DESWEB'13). Chapter 4 is a an extended version of a conference paper published in IEEE International Conference on Semantic Computing (ICSC'12). Chapter 5 is an extended version of a conference paper accepted in International Conference on Data Management Technologies and Applications (DATA'13). Finally, Chapter 6 is a conference paper published in International Conference on Conceptual Modeling (ER'12). The overall architecture of the thesis proposing a framework for semantic information integration is shown in Figure 1-2. This architecture shows how different components (each component

presented as chapter) provides the semantic information integration services at different levels of abstraction. More specifically, the architecture in Figure 1-2 represents the components of the thesis from the theoretical foundations layer to the physical storage layer. The main components are Data Integration and Data Exchange components that employ enriched schema mappings and store data at physical storage layer provided by Sliced Column Store component

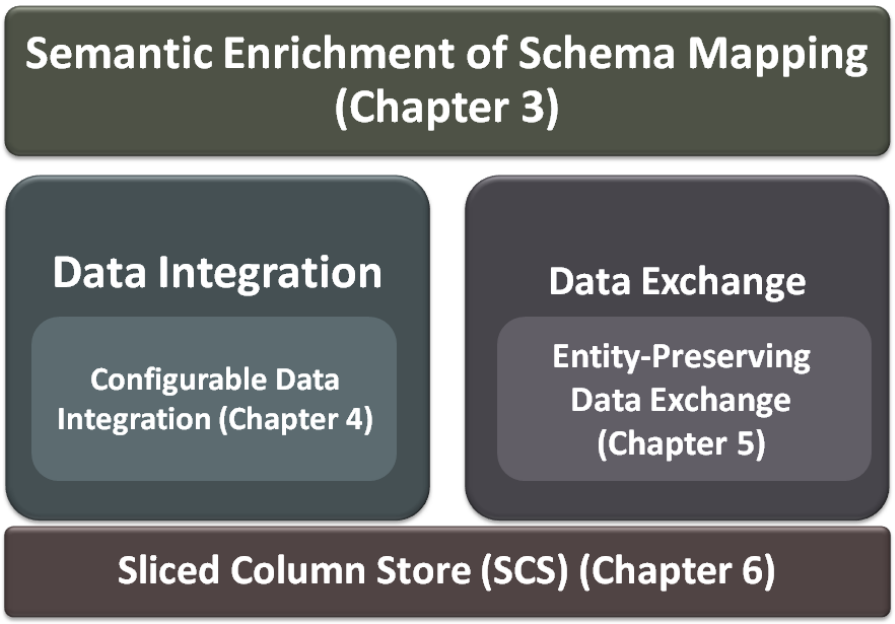


Figure 1-2: The overall architecture of the thesis

In Chapter 2, I provide a comprehensive review of the literature in information integration. In this chapter, schema mapping techniques, different types of mappings, various methods to generate schema mappings, validation of schema mappings and uncertainty in mappings are discussed. In addition, ambiguous mappings and current techniques to resolve such ambiguities in schema mapping are elaborated. This chapter follows with an overview of data integration techniques based on schema mappings. Query rewriting, query answering, query processing, and various existing query

reformulation techniques are elaborated for data integration. Then, an overview of the data exchange techniques and challenges to exchange data in distributed and peer to peer settings are discussed. This chapter continues with semantic heterogeneity reconciliation as one of the basic components of information integration. In addition, challenges in semantic heterogeneity reconciliation and fundamental issues in existing techniques are discussed. Some ontological principles regarding semantic reconciliation techniques to boost semantic information integration are provided.

In Chapter 3, I propose a set of schemas mapping generation algorithms, in which relational schemas are enriched by conceptual model semantics, to resolve mapping ambiguity. In this chapter I show how jointly considering mappings results in exploring some new plausible mappings that are not generated in many existing techniques. While existing proposals rely on the key role of user-feedback or data examples to verify mappings after mapping generation, the approach I propose in this chapter allows the direct reuse of expert knowledge. I argue that it would be much easier for a domain expert to prepare a conceptual schema (or domain ontology) than to judge the correctness of complex schema mappings and resolve ambiguous mappings after each mapping generation.

In Chapter 4, I propose principled extensions to schema mapping generator components of data exchange and data integration systems. In the technique proposed in this chapter, mappings are enhanced incrementally using auxiliary information in terms of natural relations between properties. I show how complex semantic heterogeneities can be grounded in ontological foundations that provide a theoretical basis for semantic

heterogeneity reconciliation. I argue that direct and indirect inferences can be used to bind two different data sources. To this end, I first discuss how schema mapping algorithms can employ such inferences to reconcile semantic heterogeneities. Then, I show how mappings can be enhanced by using such inferences. I discuss what it means to accurately or completely answer a target query given some basic relations between properties in different data sources.

In Chapter 5, to fill the gap between data centric and schema centric approaches for data exchange, I propose an entity preserving approach that focuses on preserving source entities in the target independent of classification. In this approach, property correspondences are directly used to find the best relations in which source entities can be located without generating schema mappings. I argue that schema mapping expressions are not expressive enough to convey the whole semantics in data exchange. This chapter is an extension of Chapter 3, in which instead of using conceptual models to semantically enhance schema mappings, a solution is proposed that relies solely on source and target data sources as well as correspondences between properties. In this chapter I employ an improved data storage model proposed in Chapter 6 to improve the efficiency of data exchange.

In Chapter 6, I address the problem of efficiency in data exchange. To speed up processing the entity preserving approach I have proposed in Chapter 5, I suggest a new data storage model based on ontological foundations. Using these theoretical foundations, I show that existing column-store data storage models can be improved by column slicing. Although the column slicing technique proposed in Chapter 6 is an independent

paper to improve read-oriented queries, the rationale for adding this chapter to the thesis is that the column slicing technique is used in Chapter 5 to speed up the data exchange process. Moreover, the slicing technique proposed in Chapter 6 is also motivated by the ontological foundations.

Finally, in Chapter 7, I summarize the techniques proposed in this thesis. In particular, I show the significance and importance of the problems and solutions discussed in this thesis. I also show how each of the research questions has been addressed in the thesis. In addition, I highlight the main contributions of the thesis and discuss opportunities that ontological foundations can contribute to various aspects of information integration.

1.6 References

Arch-Int, N., Li, Y., Roe, P., & Sophatsathit, P. (2003). Query processing the heterogeneous information sources using ontology-based approach. *Proceedings of the 18th International Conference on Computers and their Applications*, Honolulu, HI, USA. 438-441.

Beath, C., Becerra-Fernandez, I., Ross, J., & Short, J. (2012). Finding value in the information explosion. *MIT Sloan Management Review*, *53*(4), 18.

Bonifati, A., Chang, E. Q., Lakshmanan, A. V. S., Ho, T., & Pottinger, R. (2005). HePToX: marrying XML and heterogeneity in your P2P databases. *Proceedings of the 31st International Conference on Very Large Data Bases*, Trondheim, Norway. 1267-1270.

Bonifati, A., Mecca, G., Papotti, P., & Velegrakis, Y. (2011). Discovery and correctness of schema mapping transformations. In Bellahsene, Z., Bonifati, A. & Rahm, E. (Ed.), *Schema Matching and Mapping* (pp. 111-147). Berlin, Heidelberg: Springer-Verlag.

Bunge, M. (1977). *Treatise on Basic Philosophy*: *the Furniture of the World*. Boston, MA: Reidel.

Doan, A., Halevy, A. Y., & Ives, Z. (2012). *Principles of Data Integration*, Waltham, MA: Morgan Kaufmann.

Fagin, R., Kolaitis, P. G., Miller, R. J., & Popa, L. (2005). Data exchange: semantics and query answering. *Theoretical Computer Science, 336*(1), 89-124. doi: 10.1016/j.tcs.2004.10.033

Gemino, A., & Wand, Y. (2004). A framework for empirical evaluation of conceptual modeling techniques. *Requirements Engineering, 9*(4), 248-260. doi: 10.1007/s00766-004-0204-6

Haas, L. M. (2006). Beauty and the beast: the theory and practice of information integration. *Proceedings of the 11th International Conference on Database Theory,* Barcelona, Spain. 28-43. doi: 10.1007/11965893_3

Haas, L. M., Hentschel, M., Kossmann, D., & Miller, R. J. (2009). Schema AND data: a holistic approach to mapping, resolution and fusion in information integration. *Proceedings of the 28th International Conference on Conceptual Modeling,* Gramado, Brazil. 27-40. doi: 10.1007/978-3-642-04840-1_5

Halevy, A. Y. (2010). Technical perspective schema mappings: rules for mixing data. *Communications of the ACM, 53*(1), 100-101.

Hentschel, M., Kossmann, D., Florescu, D., Haas, L. M., Kraska, T., & Miller, R. J. (2009). Scalable data integration by mapping data to queries. *ETH Zurich, Computer Science, Technical Report, 633.*

Marnette, B., Mecca, G., Papotti, P., Raunich, S., & Santoro, D. (2011). ++Spicy: an open-source tool for second-generation schema mapping and data exchange. *Proceedings of the VLDB Endowment, 4*(12), 1438-1441.

Parsons, J., & Wand, Y. (2000). Emancipating instances from the tyranny of classes in information modeling. *ACM Transactions on Database Systems, 25*(2), 228-268. doi: 10.1145/357775.357778

Parsons, J., & Wand, Y. (2003). Attribute-based semantic reconciliation of multiple data sources. *Journal on Data Semantics*, *2800*(1), 21-47. doi: 10.1007/978-3-540-39733-5_2

Parsons, J. (2011). An experimental study of the effects of representing property precedence on the comprehension of conceptual schemas. *Journal of the Association for Information Systems, 12*(6), 1.

Popa, L., Velegrakis, Y., Hernández, M. A., Miller, R. J., & Fagin, R. (2002). Translating Web data. *Proceedings of the 28th International Conference on Very Large Data Bases,* Hong Kong, China. 598-609.

San, R. (2012). *Ventana research 2012 value index for data integration.* (Research). CA, USA: Ventana Research.

Shanks, G., Tansley, E., Nuredini, J., Tobin, D., & Weber, R. (2002). Representing part-whole relationships in conceptual modeling: an empirical evaluation. *Proceedings of the 23rd International Conference on Information Systems,* Barcelona, Spain. 89-100.

ten Cate, B., Kolaitis, P. G., & Tan, W. (2013). Schema mappings and data examples. *Proceedings of the 16th International Conference on Extending Database Technology*, Genoa, Italy. 777-780. doi: 10.1145/2452376.2452479

Wand, Y., Storey, V. C., & Weber, R. (1999). An ontological analysis of the relationship construct in conceptual modeling. *ACM Transactions on Database Systems*, *24*(4), 494-528. doi: 10.1145/331983.331989

Chapter 2 Literature Review

In this chapter, a comprehensive review of the literature in information integration including schema mapping, data integration, data exchange, semantic issues in information integration and ontological foundations for semantic heterogeneity reconciliation is provided.

2.1 Schema Mapping

The first step toward information integration is identifying relationships between source and target. For this purpose, the database research community has introduced the concept of schema mapping. A schema mapping is a high level expression using logical formalism that represents relations between source and target. These high level abstractions allow separating the specification of the relationships between data sources from the implementation details. The use of schema mappings helps to better understand and reason about the relationships between data sources. Such mappings can be automatically compiled to executable scripts in various databases.

Clio (Miller et al., 2000; Popa et al., 2002) is one of the leading projects in information integration providing different levels of abstraction in information integration in an incremental fashion. This system employs the semantics in simple correspondences between data items in source and target as well as semantics of constraints embedded in schemas to determine a set of mappings between source and target. This approach has been used in many recent information integration systems such as HeptoX (Bonifati et al., 2005; Bonifati et al., 2010) and ++Spicy (Marnette et al., 2011), and the general

algorithm has been subsequently extended in several ways to support nested schemas (Fuxman et al., 2006a), semantic integration (An et al., 2007), and visual languages (Bohannon et al., 2006; Raffio et al., 2008)

A concrete example of schema mapping between a source and a target schema is shown in Figure 2-1. The schema mapping is shown by a sentence in first order logic. This sentence states that whenever the source relation *patient* contains (x_1, x_2, x_3), and *observation* contains the triple (x_1, x_4, x_5), then there exist values y_1, y_2, y_3 and y_4 such that the target relation *person* contains (x_1, y_1, y_2), and *examination* contains (x_1, y_3, y_4). This schema mapping is created from a set of referential integrity constraints (r_1, r_3) in the source and target schemas, and property correspondences (c_1, c_2). In data exchange, such mappings are used to transform a source instance to a target instance, and on the data integration side, these mappings are used to rewrite the query posed to a target schema to a set of queries that are applicable to source schemas.

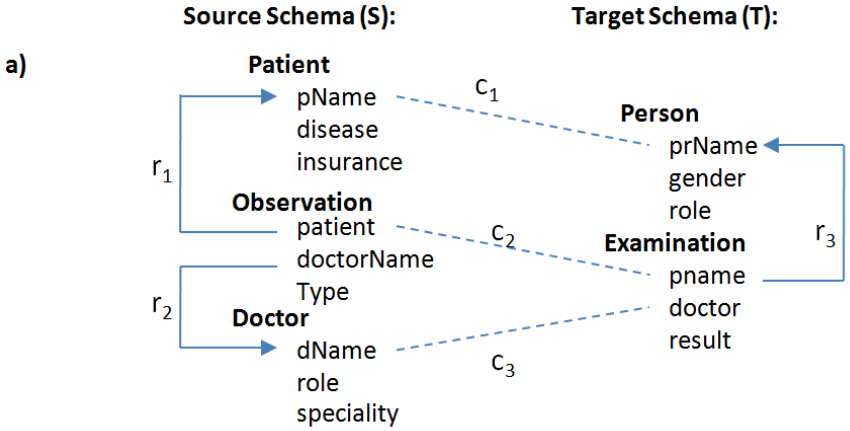


Figure 2-1: An example of schema mapping (mp_1) between source schema and target schema using property correspondences (c_1, c_2) and referential integrity constraints (r_1, r_2 r_3)

2.1.1 Mapping Types

GAV (Global-As-View) and LAV (Local-As-View) are two important types of mappings. In GAV, each element of the target schema is specified using a view over the source schema. For example, $\forall x_{1,\ldots,x_5}\ Observation(x_1,x_2,x_3),\ Doctor\ (x_2,\ x_4,\ x_5) \rightarrow \exists y_1\ Person(x_1,\ x_2,\ y_1)$ is an example of GAV mapping for the source and target schemas in Figure 2-1 in which the right hand side of the mapping expression is a single atomic formula. In this setting, to answer a query posed to the target, symbols in target query are replaced with their definition in terms of the sources. On the other hand, in the LAV approach, source items are specified in terms of queries over a (virtual) target schema. For example, $\forall x_{1,\ldots,x_3}\ Observation(x_1,x_2,x_3) \rightarrow \exists y_1 y_2,y_3\ Person(x_1,\ y_1,\ y_2),\ Examination(x_1,x_2,\ y_3)$ is an example of LAV mapping in which the left hand side of the mapping expression is a single atomic formula. In this way, precise semantics is assigned to sources. As discussed in (Calvanese et al., 2013), the LAV approach may not be practically realizable since schemas of legacy sources are not well defined enough to generate a clean description in terms of a query over the target. On the other hand, GLAV (Global-Local-As-View) are the most general form of mapping assertions, specifying relations between a view over the source and a view over the target. In other words, GLAV mappings are generalization of both GAV and LAV. This type of mapping is a sound mapping where the answers to the view on the source are a subset of the answers to the view on the target (Calvanese et al., 2013). As discussed in (ten Cate et al., 2009), in spite of the syntactic simplicity of GLAV mappings, these mappings are able to define many data interoperability tasks. GLAV mappings are also known as *source-to-target*

tuple generating dependencies (st-tgd) asserting that, if a set of entities exist in the source, then another set of entities must exist in the target. A source to target dependency is statement of the form $\forall \boldsymbol{x}\ (\phi(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y}\ \psi(\boldsymbol{x},\boldsymbol{y}))$ where $\phi(\boldsymbol{x})$ is a conjunction of atomic formulas with free variable $\boldsymbol{x}$ over S, and $\psi(\boldsymbol{x},\boldsymbol{y})$ is conjunction of atomic formulas over T with variables $\boldsymbol{x}$ and $\boldsymbol{y}$. An example of such source to target dependency is mp_1 in Figure 2-1. This thesis focuses on GLAV mappings as the most general expression representing relations between source and target schemas.

Time specific mappings have also been considered recently (Roth & Tan, 2013). Generally, in scientific databases, data is accompanied with an implicit time component representing the publication date. In addition, data may include time specific properties such as date of birth, graduation date, and transaction date. To handle such time-aware data, a new approach for schema mapping to support data integration and data exchange across time was proposed in Roth & Tan (2013). In particular, they proposed a formal framework for data integration and data exchange across time that allows generating integrated longitudinal knowledge about entities. This longitudinal knowledge of an entity indicates when one knows what one knows about the entity.

2.1.2 Schema Mapping Generation

Automatically or semi automatically generating schema mappings is an important step towards information integration that is already considered in many data integration and data exchange systems (Bohannon et al., 2006; Bonifati et al., 2005; Bonifati et al., 2010; Fuxman et al., 2006a; Fuxman et al., 2006b; Marnette et al., 2011; Miller et al., 2000; Popa et al., 2002; Raffio et al., 2008; An et al., 2007). Generally, mappings are generated

using direct correspondences between properties (representing equivalent relations) as well as integrity constraints. Property correspondences can be identified manually in a visual interface by users (drawing lines between properties of source and target), or they can be generated automatically using schema matching techniques (through string similarities data types, structure similarities).

As an alternative way to generate mappings, a technique based on data examples was proposed in (ten Cate et al., 2013). In this technique, called example-driven schema mapping, pairs of source and target examples conforming to the source and target schemas are used to generate mappings. In (Qian et al., 2012), mappings are generated in real time using users' examples without need for direct property correspondences. In this system, users are isolated from semantics of the source and target schema where there is no need for users to understand operations required to map source and target. In (Elmeleegy et al., 2011), a technique is proposed for schema mapping in which query logs of data sources being mapped are used to increase the quality of mappings generated. In this technique, an aggressive Chase algorithm is used that allows generation of different mappings including many-to-many relations. This technique uses a conflict resolution technique to resolve the attribute correspondence conflicts between logical relations of the two schemas.

2.1.3 Mapping Verification and Ambiguous Mappings

Mapping generation techniques may result in generating some ambiguous scenarios where there is more than one way to interpret a transformation rule. Examples of schema mapping scenarios, in which mapping scenario can be interpreted in several different

ways, are discussed in (Alexe et al., 2008). Inability to resolve such ambiguities can lead to unexpected results in data integration or data exchange systems using such mappings. We classify existing ambiguity resolution techniques into four categories: *verification using domain ontologies*, *verification by mapping designer*, *verification using examples*, *mapping debugging*, and *uncertainty modeling*.

A. Verification using domain ontologies*:* In (Embley et al., 2004), a domain specific ontology is used to explore relationships in terms of Merge/Split and Superset/Subset between data sources to resolve ambiguous scenarios. However, as discussed in (Wang & Pottinger, 2008), generating this type of domain ontology may not be practical in real data integration scenarios. In (Cappellari et al., 2010), a mapping verification technique, in which mappings are checked against semantic annotations in source and target schemas is proposed. This technique assumes that the reconciliation knowledge of experts is represented as annotations in source and target schemas. This technique allows reusing this knowledge without direct interference of users in mapping generation process. However, generating such annotated schemas can also be expensive.

B. Verification by mapping designers*:* Based on four properties of mappings (mapping satisfiability, mapping inference, query answerability and mapping losslessness), a mapping validation technique is proposed in (Rull et al., 2013) in which the system asks a mapping designer if the mappings satisfy these properties. These properties are used to check if there is an instance of the database for which the query can return some tuples. To resolve ambiguous mappings, (Chiticariu et al., 2008) proposed an interactive

technique in which user feedback is used to refine mappings. In particular, users are asked to select what merging technique must be used in each ambiguous case. Unlike Clio's mapping generation algorithm, this mapping algorithm resolves ambiguity by taking full advantage of users' knowledge in terms of how the integrated schema is generated. However, this assumption may limit the usage of this approach in large schemas where many ambiguous cases must be resolved by users. In addition, users must completely be aware of the semantics of the source and target schema. In a similar approach, to employ users' knowledge in schema mapping, a system developed in (Lu et al., 2011) where semantics of data are interpreted through the tags provided by users in mapping generation. In this system, the input query is also checked against the tags provided by users to interpret the semantics of the input query. In (Mandreoli & Martoglia, 2011), a technique to convert the input structures into a common format is proposed in which users assign a context to mappings regarding structure characteristics.

C. Verification using examples:

Muse (Alexe et al., 2008) is another example-driven technique using data examples in which data examples are used to indicate the best mappings among a set of candidate mappings. This system assists mapping designers to understand and refine mappings. This system is able to extract the desired semantics based on users' actions on a set of examples. In EIRENE (Alexe et al., 2011), given a set of data examples, the algorithm decides whether or not there exists a schema mapping fitting these data examples. In the case when no mapping is generated, EIRENE allows users to modify data examples. In (Bonifati et al., 2008), it is argued that the existence of several possible interpretations

results in ambiguity problems affecting the quality of mappings. They proposed a system to verify mappings (called spicy) in which a mapping verification component checks candidate mappings and selects those mappings that are consistent with target constraints.

D. Mapping debugging: TRAMP (TRAnsformation Mapping Provenance) (Glavic et al., 2010) is a mapping debugger tool which allows tracing schema mappings. In this system, data provenance is considered in terms of transformation provenance and mapping provenance. Using this technique, it is possible to understand relations between data transformed and mappings. This technique makes it possible to deal with large complex schemas in schema mapping. However, debugging integration is a very expensive process. In SPIDER (Alexe et al., 2006), candidate mappings are first generated using match driven techniques and then refined through debugging the mappings by users.

E. Uncertainty modeling: As argued in (Das Sarma et al., 2011), because the semantics of mappings between a mediated schema and sources can be approximate, schema mapping techniques must be able to handle uncertainty in schema mappings. Moreover, as discussed in (Dong et al., 2009), it is not possible to find exact mappings due to the fact that sometimes structured data are extracted from unstructured data. In the case of using keyword search, modeling uncertainty also allows extracting many possible results ranked based on the probability of matching. The notion of uncertain query answering has resulted in a new technique for query answering called probabilistic query answering. This technique returns a set of ranked top-k results to an input query. In (Dong et al., 2009), a technique to answer queries using a set of probabilistic schema mappings is

proposed. This notion has also been considered in data exchange where the purpose is generating a target instance as consistent as possible to mappings. In (Magnani et al., 2005), a technique to generate a set of alternative mediated schemas using probabilistic relations between data items in source and target is proposed. An empirical study by (Magnani & Montesi, 2007) shows that using top-k schema mappings in query answering can result in increasing recall.

2.2 Data Exchange

Data exchange is the process of residing and materialization of source data in the target such that transformed data conforms to target constraints. In other words, data exchange entails restricting source data in the target such that source data is reflected as accurately and completely as possible in the target. In addition to practical tools and systems to exchange data, there have also been theoretical studies on data exchange that have been conducted several years after practical development of these systems (Mecca, Papotti, & Raunich, 2012). These theoretical studies include formalizing the notion of data exchange (Fagin et al., 2005a; Fagin et al., 2005c), composing schema mappings (Fagin et al., 2005b), inverting mappings (Fagin, 2007), and many other studies. Formalizing the data exchange problem has been an important step towards developing data exchange tools.

Several extensions of data exchange setting are discussed in (Pablo, 2009). In (Fuxman et al., 2006b), a technique, in which target to source dependencies as well as source to target dependencies are taken into account for data exchange, is proposed. This forms a peer to peer data exchange setting in which different peers interact autonomously to exchange data.

2.2.1 Data Exchange Formalism

This thesis follows the formalized notions proposed by (Fagin et al., 2005a). In the data exchange problem, source and target databases are considered as sets of relations. A relational schema is a finite collection $R=\{R_1,..,R_k\}$ of relations in which R_i has a fixed arity (number of properties). In this setting, source $S=\{S_1,..,S_n\}$ and target $T=\{T_1,..,T_m\}$ are two disjoint schemas. An instance I of a schema S is a set of instances over relations of S where an instance of a relation $\{A_1,...,A_k\}$ is a finite set of tuples in the form of $R(A_1{:}v_1,...,A_k{:}v_k)$.

Different types of dependencies are used in the data exchange context to specify mappings between source and target. These dependencies include tuple generating dependencies (tgds) and equality generating dependencies (egds) (Bonifati et al., 2011). Tuple generating dependencies include source-to-target tuple generating dependencies (s-t tgds) and target tuple generating dependencies (target tgds). A s-t tgds is a dependency in which source relations are used in the premise and target relations are used in the conclusion. This type of dependency is used to specify how a tuple is generated in the target given a tuple in the source. On the other hand, in a target tgd, only target symbols are used that are typically employed to specify foreign-keys in the target. Target equality generating dependencies (egds) are used to show primary key constraints in the target.

In this formalized setting, data exchange is a quadruple $(\boldsymbol{S}, \boldsymbol{T}, \Sigma_{st}, \Sigma_t)$ in which S and T represent source and target schema, respectively. Σ_{st} is a set of source-to-target dependencies representing relations between source and target, and Σ_t is a set of constraints in the target. In this setting, the data exchange problem is known as finding a

target solution J over T given source instance I over S such that (I, J) satisfy Σ_{st} together, and J satisfies Σ_t. Instance J is known as a solution for this data exchange.

One of the important questions in data exchange is that given a source dataset, target schema, and a set of mappings between the source and the target, what would be the best target instance that must be materialized in the target. In (Bonifati et al., 2011), a desirable target instance is defined as a legal instance satisfying correspondences between the source and the target, and integrity constraints in the target. Such instance contains all source information while no piece of information is reported twice. From the theoretical point of view, several studies have been conducted on the concept of *universal solution*, which is a solution satisfying all mapping expressions (Fagin et al., 2005a) , and *core solution*, which is an optimal solution among universal solutions without redundancies (Fagin et al., 2005c). Core solution has been considered as a good and natural solution because of irredundancy and completeness. As shown in Chapter 5, the entity preserving approach proposed in this thesis generates a core solution for the data exchange problem.

2.2.2 Universal and Core Solutions

The notion of homomorphism among two solutions must be defined to formalize *universal solution* and *core solution*. According to the formalism in (Fagin et al., 2005a), the set of all constant values that may occur in source instances is denoted $\underline{Const}$, and an infinite set of variables (called *labeled nulls*) are denoted $\underline{Var}$ such that $\underline{Var} \cap \underline{Const} = \emptyset$. In generating target solutions, variables are used to create new values in the target that are not in the source. Each element of a tuple $t=\{a_1,a_2,...,a_n\}$ over a relation from an instance is a member of $\underline{Const} \cup \underline{Var}$. Given K_1 and K_2 denoting two instances over a relational

schema R with values in $\underline{Const} \cup \underline{Var}$, a homomorphism $h: K_1 \rightarrow K_2$ is a mapping from $Const \cup Var\ (K_1)$ to $Const \cup Var\ (K_2)$ such that:

(1) $h(c) = c$ for every $c \in Const$;

(2) for every fact $R_i(t)$ of K_1, $R_i(h(t))$ is a fact of K_2 where, if $t = (a_1,...,a_n)$, then $(t)= (h(a_1),..., h(a_n))$.

Using the concept of homomorphism between instances, a *universal solution* is defined as follows: If I is a source instance, then a *universal solution* for I is a solution J such that for every solution J' for I, there exists a homomorphism $h: J \rightarrow J'$. Several good properties of universal solutions justify its choice as a good target solution in a data exchange problem. This solution is considered as the most general possible solution because it can be homomoriphically mapped to any arbitrary solution. In addition, as discussed in (Fagin et al., 2005a), universal solutions of a given source instance in a data exchange problem are homomophically equivalent. Generally, the Chase algorithm (Popa & Tannen, 1999) is used to generate universal solutions. This algorithm starts with a source instance and an empty target instance. In each Chase step, source-to-target dependencies and target dependencies are applied on the source and target instances as long as they satisfy these dependencies.

The solution with smallest size among universal solutions is called the *core solution* (Fagin et al., 2005c). Because of the minimality and uniqueness of the core solution among universal solutions, this solution is considered as an ideal solution for data exchange. A target instance J among universal solution is called a *core solution* if there is no proper subinstance $J' \subseteq J$ such that there is a homomorphism $h: J \rightarrow J'$. A

general technique to generate the core solution in a relational data exchange problem is generating a universal solution using the Chase algorithm and then post processing the universal solution to remove redundancies. In (Fagin et al., 2005c) a greedy algorithm is proposed where given a source instance *I*, a universal solution *J* generated. Then, each tuple is checked against source-to-target dependencies and target constraints to find redundant tuples.

2.3 Data Integration

Data integration is known as virtual information integration in which a query interface works as a mediator to access data. As argued in (Haas et al., 2009), data integration must be considered as a process including understanding data, cleaning data, matching application needs, and representing and standardizing data across data sources. Data integration can be performed to fulfill different goals. While some applications such as financial applications require perfect accuracy, in some other applications fast response is more important than accuracy and completeness. On the other hand, as discussed in Chapter 4, accuracy can be compromised to achieve complete results. In the following, a review of some important issues in data integration including query rewriting, query processing, peer to peer data integration, ontological queries and incremental data integration is provided.

2.3.1 Query Rewriting

In (Arenas et al., 2004), query rewriting is defined as the following problem: For a query q, find a query q´ such that given a source instance, the answers of q over the source

returns the same result as evaluating q´ over the target. They proposed a technique to indicate the possibility of rewriting an input query. Query rewriting has also been considered for keyword query. In (Fagin et al., 2011), an example of keyword query is proposed in which auxiliary data extracted from rewriting rules are used to reformulate an input query to generate a set of alternative queries. They proposed a formalized notion of rewriting based on rewriting rules in which an input keyword query augmented by additional queries are executed against the underlying search index. In (Marnette et al., 2010), a rewriting algorithm is proposed in which a set of source to target dependencies and equality generating dependencies are rewritten to some dependencies without equality generating dependencies. Then, SQL scripts are used to implement such dependencies. In this technique the overlap between dependencies is employed for rewriting.

2.3.2 Query Processing

In query processing using schema mappings, a query posed to the target schema is answered using source databases and also mappings between sources and target. Query processing can be performed using two different approaches. First, finding certain answers, which are answers in all target databases satisfying the schema mapping with respect to the source. The concept of certain answers originated in the study of incomplete databases. The concept of certain answers ensures that if a tuple *t* belongs to certain answers of a query posed to the source, then *t* belongs to the result of the query posed to the target on every solution *J* for *I*. On the other hand, in the query rewriting approach, the target query is reformulated using mappings. Then the rewritten queries are

evaluated over the source schema. In the case of different rewritings, the best rewriting is selected which is the rewriting guaranteeing the maximal set of answers. This approach ensures returning certain answers. Differences and relations between query rewriting and query answering are elaborated in (Calvanese et al., 2013). Query rewriting can be considered as a means to perform query answering while query answering can be achieved using query rewriting.

Different query processing approaches are proposed depending on various types of schema mapping expressions. As discussed in (Calvanese et al., 2013), in query processing using GAV mappings, each target atom is mapped to a query over the source, and in LAV mappings, each source atom is mapped to a query over the target. On the other hand, in the general case, queries over the source are mapped to queries over the target, which is called Query processing under GLAV mappings.

In data integration through query processing, the data integration setting is defined as: what does answering query q over T mean given a source instance I over the source schema S, and a set of source-to-target dependencies between source S and target T. As discussed in (Fagin et al., 2005a) because there may exist more than one target solution J satisfying Σ_{st}, $q(J)$ may return different results. To address this problem Fagin et al., (2005a) proposed the concept of certain answers as the answers satisfying intersection of all $q(J)$'s where J is any possible target instance satisfying Σ_{st}.

2.3.3 Ontological Queries

Another approach in query answering is using domain ontologies in which an input query is evaluated over ontological constraints as well as a database. Generally, in this approach

an input query is rewritten such that the semantics of ontologies are satisfied in the rewritten query. In (Orsi & Pieris, 2011), an algorithm to compute the perfect rewriting of a conjunctive query with respect to a linear Datalog± ontology is proposed. An ontological query answering technique is proposed in (Calì et al., 2012) in which an extended version of entity relationship diagram (including is-a relations) is used for query answering. In this technique, chasing a database against a set of inclusion dependencies is an important step to find answers in an ontological query answering.

2.3.4 Peer to Peer (P2P) Data Integration

Peer to peer data integration is an extension of data integration in which rather than creating and relying on a mediated schema, peers collaborate directly to integrate data (Doan & Halevy, 2005). This can be an ideal architecture for organizations to share data where no peer wants to be in charge of generating and maintaining mediated schema and mappings between schemas and the mediated schema.

In the P2P architecture, traversing semantic paths of mappings allows finding answers to a query from other peers that are reachable from the peer to which a query is posed. Generally, query rewriting techniques are used to reformulate queries. In (Tatarinov & Halevy, 2004) a technique is proposed to prune paths in the rewriting process for reducing the number of queries generated. In P2P query answering, first immediate peers are investigated to find answers. Then, neighbors of peers are taken into account and so on. The main idea behind this approach is that peers may not have the complete answers, and thus, the final result will be the union of the results returned from all peers.

The most important advantage of P2P architecture is that there is no need to generate a mediated schema before information integration. In addition, in this architecture, peers are free to select from which peer they want to obtain data without affecting other peers. Moreover, local mapping between peers allows a peer to build a query using its own vocabulary, where there is no need to know about other peers' schemas. In the P2P architecture, data integration is conducted locally in each peer where semantic mappings between data sources of peers must be established. As discussed in (Halevy et al., 2006; Tatarinov & Halevy, 2004), this is a complex data integration scenario that requires handling semantics between peers in general while mappings are distributed between peers. There are also some challenges in peer to peer query answering. First, finding semantic paths and pruning paths is not a trivial task as some peers who have the answer may be pruned at an early stage to improve efficiency. On the other hand, this architecture can result in generating redundant answers.

2.3.5 Incremental Data Integration

The need for incremental data integration has resulted in a new approach in data integration called pay-as-you-go data integration (Das Sarma et al., 2008). The main advantage of this approach is that there is no need for full integration of data to provide useful services for users. In this approach, the integration system starts with few semantic mappings (can be unsound), and over time, these mappings are refined and improved. Then, a set of on-the-fly mappings are created that are revised over time. At any point during incremental data integration, the system must be able to answer queries as best as possible based on available semantic relationships. This is different from traditional data

integration that requires a complete knowledge of semantic relationships before query answering.

PAYGO (Madhavan et al., 2007) is an example of pay-as-you-go data integration systems without need for generating a single mediated schema. In this system, approximate semantic mappings are established between data sources. Consequently, uncertainty is modeled in mappings as well as underlying data. An important feature of this system is that PAYGO does not return a single true answer to a given query. Instead, the system returns a ranked list of potential answers.

2.4 Semantic Interoperability in Information Integration

According to (Hakimpour & Geppert, 2001), semantic heterogeneity problems occur in information integration when different data sources use different terms to refer to the same concept, or they use the same term to refer to different concepts in the real-world. Such semantic heterogeneities are the consequence of different interpretations of concepts that usually happens during conceptualization of the real world in a data source (George & Preston, 2005). This is due to the fact that different people with different background and vocabularies may design data sources. Such semantic heterogeneity can also be the consequence of presenting the same concepts at different levels of abstraction.

2.4.1 Semantic Heterogeneity Reconciliation

Semantic reconciliation is a fundamental process required to make intelligent systems and interaction between them. Semantic reconciliation is required because different people collect features of the same object in the real-world in a different way. This process

should be considered at earlier steps of information integration because secondary analysis and revising data can be much harder and error prone.

In spite of considerable improvements in information integration to simplify the schema mapping process, semantic issues are not considered as much as syntax issues in these systems. However, as discussed earlier, neglecting the semantics of data in data integration can hinder proper information interoperability. Properly handling semantics provides common agreement on accurate and complete transformation semantics that can empower data integration and data exchange processes. In addition, considering semantics can improve a large and growing body of research on mappings' management activities such as mapping inversion, mapping composition, mapping evolution and mapping maintenance.

Manually extracting semantics of data from documentations and designers knowledge cannot be a practical solution (Doan & Halevy, 2005). In addition, relying on clues such as data type, data structure, referential integrity constraints and string similarities cannot handle full semantic heterogeneities as these clues address only syntaxes and representations of concepts. Fully reconciling semantics heterogeneities requires knowledge outside the syntax of data sources. As argued in (Hassanzadeh et al., 2009), because a real-world concept or characteristic can be represented through different syntactic representations in different data sources, higher level information such as domain knowledge is necessary for schema mapping. A general approach for semantic heterogeneity reconciliation is through relying on a domain ontology as common sense including explicit definition of terms used in different schemas. In (Embley et al., 2004),

a domain-specific ontology is used to explore relationships in terms of Merge/Split and Superset/Subset between data sources. However, as discussed in (Wang & Pottinger, 2008), generating such domain ontologies may not be practical in real data integration scenarios. In (Pankowski, 2013) data exchange operations are represented in a formalized and knowledge based system using shared domain ontologies.

2.4.2 Challenges in Semantic Reconciliation

Despite the large amount of research on information integration techniques, semantic heterogeneity reconciliation is not well studied for this purpose.

From the schema mapping point of view, identifying property correspondences representing equivalent relations between properties of the source and the target is the key to generate mappings. Although generating such property correspondences can be automated using schema matching techniques, and understanding and maintaining them can be easily performed by users using visual interfaces, such property correspondences are not expressive enough to convey the full semantics of a data exchange or data integration. In (Bonifati et al., 2011), this issue is attributed to the problem of inherent ambiguity in direct correspondences between properties. More specifically, while mappings may conform to a set of property correspondences, they may not express the same semantics. As discussed in (Wang & Pottinger, 2008), deepening the mapping semantics results in generating richer information integration application. In SeMap (Wang & Pottinger, 2008), a semantically richer mapping including generalization-specialization, and ‘Has-a’ relations are used for this purpose.

Considering a set of different types of relations between source and target in schema mapping is not a trivial task because of need to reconcile conflicts between different matches. This requires handling a larger number of correspondences compared to equivalent correspondences. Enhanced matching techniques are required for this purpose, which is a different area of research under schema matching.

Unlike many existing mapping techniques, in which semantic heterogeneity reconciliation is performed through matching attributes that have similar or the same meaning in different data sources, an attribute-based semantic heterogeneity reconciliation proposed in (Parsons & Wand, 2003) uses relations between properties instead of their actual meaning. They showed how structurally different attributes can manifest the same higher level property, that consequently can be used for semantic heterogeneity reconciliation. This thesis shows that relations between properties of different instances can also be used to describe semantic similarities. More specifically, I show how complex semantic heterogeneities can be grounded in ontological foundations that provide a theoretical basis for semantic heterogeneity reconciliation techniques.

In (Pottinger & Bernstein, 2008), it is argued that simple correspondences between properties of source and target cannot handle the full semantics in schema mapping. As a result, formal semantics regarding overlapping elements is required to generate mappings. They employed conjunctive queries that are expressive enough to address this problem. In this technique, that performs mapping on set of relational schemas, each relation schema R includes a set of attributes denoted *attr(R)*. A conjunctive query *Q* is shown in the form of $q(X):- e_1(X_1), \ldots, e_n(X_n)$, in which *q* and *e(X)*

are relation names. In this formula, the tuples $X_1, \ldots, X_n$ have the same arity similar to their relations. Every variable in X also exists in $e_1(X_1), \ldots, e_n(X_n)$. To answer such a query, constants are assigned to the variable X such that for some assignment of constants to the variables of their query, $e_1(X_1), \ldots, e_n(X_n)$ is true. Consequently, the final answer is the union of the answers of the queries in the set.

The gap between data centric and schema centric integration activities has been another drawback in existing information techniques. In Chapter 4, I discusse how schema mapping expressions can be enhanced considering data centric and metadata centric information simultaneously. Preserving data semantics in data exchange is a fundamental issue in data exchange where neglecting this issue can result in improper data exchange.

2.5 Theoretical Foundations for Semantic Information Integration

Although a large amount of research has been conducted in information integration, there has been a limited progress in practical use of these techniques (San, 2012). As discussed in (Haas, 2006), there is a lack of a deep understanding of what fundamental operations are required to integrate information. In (Parsons & Wand, 2003) it is argued that the lack of theoretical foundation behind semantic reconciliation techniques has been the main reason for limited progress in this area. This thesis searches for a solution for this problem using upper-level ontology, which considers concepts (in particular things and properties) to describe the real world. Ontology is a branch of philosophy that considers “what is out there” in the real world (Wand et al., 1999). Unlike domain ontologies used in Artificial Networks representing a specific domain, ontology can be considered as an “upper-level

ontology", "meta ontology" or "top-level ontology". Carnap's ontology (Carnap & George, 1969) and Bunge's ontology (Bunge, 1977) are instances of such upper-level ontologies. In (Milton, 2004), it is argued that upper-level ontologies can provide theoretical foundations for representation and modelling in information systems to design more reliable applications and better quality database design.

2.5.1 Rationale to Use Ontological Foundations

As discussed in (Parsons & Wand, 2003), the first step towards semantic heterogeneity reconciliation in information integration is understanding the meaning and semantics of what data represent. Since databases represent the characteristics and properties of things and concepts in the real world, ontological principles can be used to deal with concepts, things and properties in the real world. In (Fonseca, 2007), the role of ontological principles for development of information systems is discussed. In (Ceusters et al., 2004), it is argued that a top-level ontology can be used to support data integration and information interoperability.

As contended in (Mutis & Issa, 2012), the lack of meaning definition in conceptual models and data models are the consequence of using representations to map the world into models. Understanding of a modeller requires answering philosophical questions such as "What does exist in the real world?", "What representations should be used to correctly perform interpretations?" Data modellers use data modeling items as logical structures of information to represent data that must be understandable by computers and human. These data representations can be structured when they reflect

information based on logic or reflect a description of the logical relations between data elements in such a way that computer languages or systems can process them.

Organization of data representations in a logical model forms a schema indicating data content and relationships between them. In (Wand, 1996), it is contended that a real-world system can be represented using an information system. To analyze the deep structure of the information system and investigate the role of information systems in representing the real world, a generalized model of reality is required. To address this issue, Bunge's ontology is used in (Wand, 1996) as a basis for information system modelling. This ontological approach uses fundamental ontological principles as theoretical foundations for information system modelling.

2.5.2 Using Bunge's Ontology as a Theoretical Foundation

In this thesis, I use Bunge's ontology as theoretical foundation for semantic heterogeneity reconciliation in information integration. To this end, I use some notions from Mario Bunge's ontology (Bunge, 1977). The rationale for choosing Bunge's ontology is that this ontology deals with systems in general; it is a comprehensive ontology covering many other ontologies; it is well-formalized in terms of set-theory; it has been widely considered as a theoretical foundation for conceptual modeling both in theoretical analyses (Parsons & Wand, 2000; Wand et al., 1999) and in empirical studies (Gemino & Wand, 2004; Parsons, 2011; Shanks et al., 2002). A UML model and an OWL model of Bunge's ontology is proposed in (Evermann, 2009) to make Bunge's ontology more accessible to researchers in both the Conceptual Modelling and Semantic Web community. Note that ontological foundations are only one of the possible foundations

for modelling knowledge about the world. As discussed in (Wand, 1996), linguistics and cognitive sciences can also be studied for this purpose.

According to Bunge's ontology (Bunge, 1977), the world is made of things that possess properties. Regardless of data models, databases contain information about the properties of things. More precisely, databases represent human (data modelers) perceptions of existing things. Consequently, data items in a database may exist or be perceived to exist. As theoretical foundations for information integration, I used two important notions of Bunge's ontology (Bunge, 1977) including *property precedence* and *composites*. These notions are explored and used in Chapter 3 and Chapter 4. In addition, we used the concept of *assumption of inherent classification* advanced by (Parsons & Wand, 2000) to analysis of the schema mapping techniques.

A. ***Property precedence*:** According to Bunge's ontology (Bunge, 1977), the semantics of a property can be defined based on its relationships with other properties. In this thesis, property precedence is used as one of the basic and fundamental types of relations between properties that can play a significant role to identify the semantics of properties. In particular, property P_1 precedes property P_2, if and only if, the set of things possessing P_2 is a subset of things possessing P_1. For example, the property of 'having color' precedes the property of 'having red color' since the set of instances that are red is a subset of the set of instances that have color. In this definition, the preceding property is more common property than preceded property.

   This notion of Bunge's ontology has an important role in the context of classification, and identifying properties and relations between properties. In

particular, classes are specified using generic properties, while an instance of a class possesses specific properties of these generic properties. Possessing a specific property implies possessing the general property, or in other words, possessing a general property precedes possessing its specific properties. For example, *gender* is a general property of a person class while *female* is a specific property of gender that might be possessed by any instance of person class. Each instance having a specific gender can be a member of a class that includes in its definition "has gender." (Parsons & Wand, 2000).

The concept of precedence also plays a crucial role in identifying the semantics of properties that can provide basis to support semantic information interoperability. Integrating data from multiple heterogeneous data sources requires identifying data fields in these data sources that represent the same meaning. Since data is expressed in terms of properties of things, we need to identify properties that are similar in some sense. In (Parsons & Wand, 2003), a new definition for similarity using property precedence notion is proposed. According to this definition, two properties P_1 and P_2 can be considered as similar if there exists a general property P_G such that P_G precedes P_1 and P_2. According to this definition, P_1 and P_2 are specific properties manifesting P_G. This notion can provide a new approach for information interoperability showing that properties that appear different may be manifestations of a higher-level property that has the same meaning across sources. In other words, the same higher level concept can be represented through different manifestations that have different structures. For example, the query to find patients who are in critical condition can be

accomplished by a query that searches for patients who have high diastolic blood pressure in one data source, or by a query that searches for patients with high body temperature in another data source. In other words, although *diastolic blood pressure* and *body temperature* are not similar properties, and also they are not generalizations or specializations of the *critical condition*, they are manifestations of the same higher level property, which is 'being in a critical condition'. This is an example of a new approach in which similarities are not necessarily found in similar structures. Instead, similarities are inferred from different structures when they are manifestation of the same higher level concept. This example shows that there may exist different point of view to interpret similarity. In this example, although *diastolic blood pressure* and *body temperature* are two completely different symptoms, from patients' point of view, they are considered as a critical condition.

The property precedence notion has also been used as a theoretical rationale for explicitly modelling dependence in conceptual schemas (Parsons & Cole, 2004). They designed and performed an experimental framework to assess the impact of explicitly representing precedence relations to show semantics of a domain in conceptual modeling. Their experiments showed the positive impact of including precedence semantics in class diagrams where properties can be expressed in a hierarchy from more general to more specific.

In chapters 2 and 3, the concept of property precedence is used for semantic heterogeneity relations.

B. ***Inherent Classification*:** According to Bunge's ontology (Bunge, 1977), things exist independent of any classes. In other words, existence of things are first recognized without assigning them to specific class, and then, classes are formed to organize the knowledge about the characteristics of individual things. As discussed in (Parsons & Wand, 2000), classification theory also adheres to independence of existence of things from classification. On the contrary to this fact, as argued in (Parsons & Wand, 2000), an underlying assumption in information modeling is that everything that is modeled in a domain of interest in an information system is treated as an instance of a class (or entity) in an Object-Oriented model (or Entity Relationship model). In (Parsons & Wand, 2000), this is termed as *the assumption of inherent classification*. Even though this is not stated explicitly, the assumption of inherent classification dominates information modeling where identifying classes of things is considered the first step in information system modeling. As discussed in (Parsons & Wand, 2000), the assumption of inherent classification contradicts some theoretical notions in Bunge's ontology and also cognitive science. They claim that many difficulties and complexities in information integration, schema evolution, and information interoperability in a large extent are the consequence of the assumption of inherent classification. They also contend that information system modeling based on this assumption can result in database operation problems including: handling exceptional instances, reclassifying instances, manipulating instances, and modifying classes.

Another important consequence of the assumption of inherent classification in information system modeling is that the semantics of properties of instances must be

identified within semantics of being an instance of class. This prevents defining and assigning an independent semantic to an individual property. However, according to Bunge's ontology (Bunge, 1977), properties are basic constructs in the real world providing the basis to determine similarity of things and to establish classes. Chapter 4 shows how an entity preserving approach avoids the assumption of inherent classification in data exchange. In Chapter 5, the row store technique is also analysed based on the assumption of inherent classification. The new storage model proposed in this thesis (SCS) uses these theoretical foundations to improve query performance in read-oriented queries.

C. ***Composites*:** According to Bunge's ontology (Bunge, 1977), things in the real world are either simple or composite which are involved in composition relationships. Ontological principles postulate that things can be combined to form a "composite". Composites possess at least one "emergent property" not possessed by any component. An emergent property is a property that is derived from properties of its parts, but it is not reducible to them. For example, a computer can be considered as a composite thing made up of many parts including processor, memory and other parts. Properties such as the capacity of memory or CPU speed are hereditary properties of this computer while a property such as computer power is an emergent property of this computer as a composite. For another example, considering "a student who is enrolled in a course" as a composite, the *grade* of a student is an emergent property of this composite while student's age and course name are "hereditary" properties.

According to Bunge's ontology (Bunge, 1977), every composite possesses at least one emergent property that is not inherited from any of the parts. As discussed in (Shanks et al., 2004), a composite thing can be represented as a separate entity class, or a relationship class (associations) in a conceptual model.

According to (Evermann & Wand, 2005), in a composition relationship, the parts of the composite are existentially dependent on the composite which cannot be part of any other composite at the same time. He contends that the Object-Oriented aggregation semantics are equivalent to the ontological notion of composition. In Chapter 2, the composite notion of Bunge's ontology is used as a theoretical foundation to describe the concept of sequence of relations in schema mappings.

2.6 References

Alexe, B., Chiticariu, L., & Tan, W. (2006). Spider: A schema mapping debugger. *Proceedings of the 32nd International Conference on Very Large Data Bases,* Seoul, Korea. 1179-1182.

Alexe, B., Chiticariu, L., Miller, R. J., & Wang-Chiew Tan. (2008). Muse: mapping understanding and design by example. *Proceedings of the IEEE 24rd International Conference on Data Engineering,* Cancún, México. 10-19. doi: 10.1109/ICDE.2008.4497409

Alexe, B., ten Cate, B., Kolaitis, P. G., & Tan, W. (2011). EIRENE: Interactive design and refinement of schema mappings via data examples. *Proceedings of the VLDB Endowment, 4*(12), 1414-1417.

An, Y., Borgida, A., Miller, R. J., & Mylopoulos, J. (2007). A semantic approach to discovering schema mapping expressions. *Proceedings of the IEEE 23rd International Conference on Data Engineering,* Istanbul, Turkey. 206-215. doi: 10.1109/ICDE.2007.367866

Arenas, M., Barceló Pablo, Fagin, R., & Libkin, L. (2004). Locally consistent transformations and query answering in data exchange. *Proceedings of the 23rd ACM*

SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, Paris, France. 229-240. doi: 10.1145/1055558.1055592

Bohannon, P., Elnahrawy, E., Fan, W., & Flaster, M. (2006). Putting context into schema matching. *Proceedings of the 32nd International Conference on Very Large Data Bases*, Seoul, Korea. 307-318.

Bonifati, A., Chang, E. Q., Lakshmanan, A. V. S., Ho, T., & Pottinger, R. (2005). HePToX: marrying XML and heterogeneity in your P2P databases. *Proceedings of the 31st International Conference on Very Large Data Bases*, Trondheim, Norway. 1267-1270.

Bonifati, A., Mecca, G., Pappalardo, A., Raunich, S., & Summa, G. (2008). Schema mapping verification: the spicy way. *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, Nantes, France. 85-96. doi: 10.1145/1353343.1353358

Bonifati, A., Chang, E., Ho, T., Lakshmanan, L. V., Pottinger, R., & Chung, Y. (2010). Schema mapping and query translation in heterogeneous P2P XML databases. *The VLDB Journal, 19*(2), 231-256. doi: 10.1007/s00778-009-0159-9

Bonifati, A., Mecca, G., Papotti, P., & Velegrakis, Y. (2011). Discovery and correctness of schema mapping transformations. In Bellahsene, Z., Bonifati, A. & Rahm, E. (Ed.), *Schema Matching and Mapping* (pp. 111-147). Berlin, Heidelberg: Springer-Verlag.

Bunge, M. (1977). *Treatise on Basic Philosophy*: *the Furniture of the World*. Boston, MA: Reidel.

Calì, A., Gottlob, G., & Pieris, A. (2012). Ontological query answering under expressive Entity–Relationship schemata. *Information Systems, 37*(4), 320-335.

Calvanese, D., De Giacomo, G., Lenzerini, M., & Vardi, M. Y. (2013). Query processing under GLAV mappings for relational and graph databases. *Proceedings of the 39th International Conference on Very Large Data Bases*, Trento, Italy. 61-72.

Cappellari, P., Barbosa, D., & Atzeni, P. (2010). A framework for automatic schema mapping verification through reasoning. *Proceedings of the IEEE Data Engineering Workshops*, Long Beach, CA, USA, 245-250. doi: 10.1109/ICDEW.2010.5452703

Carnap, R., & George, R. A. (1969). *The logical Structure of the World: and, Pseudoproblems in Philosophy*. Berkley, CA: Open Court Publishing.

Ceusters, W., Smith, B., & Fielding, J. M. (2004). LinkSuiteTM: formally robust ontology-based data and information integration. *Data Integration in the Life Sciences, 2994*(1), 124-139.

Chiticariu, L., Kolaitis, P. G., & Popa, L. (2008). Interactive generation of integrated schemas. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Vancouver, Canada. 833-846. doi: 10.1145/1376616.1376700

Das Sarma, A., Dong, X. L., & Halevy, A. Y. (2008). Bootstrapping pay-as-you-go data integration systems. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Vancouver, Canada. 861-874. doi: 10.1145/1376616.1376702

Das Sarma, A., Dong, X. L., & Halevy, A. Y. (2011). Uncertainty in data integration and dataspace support platforms. In Bellahsene, Z., Bonifati, A., & Rahm, E. (Ed.), *Schema Matching and Mapping* (pp. 75-108). Berlin, Heidelberg: Springer-Verlag.

Doan, A., & Halevy, A. Y. (2005). Semantic integration research in the database community: a brief survey. *AI Magazine, 26*(1), 83-94.

Dong, X. L., Halevy, A. Y., & Yu, C. (2009). Data integration with uncertainty. *The VLDB Journal, 18*(2), 469-500. doi: 10.1007/s00778-008-0119-9

Elmeleegy, H., Elmagarmid, A., & Lee, J. (2011). Leveraging query logs for schema mapping generation in U-MAP. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Athens, Greece. 121-132. doi: 10.1145/1989323.1989337

Embley, D. W., Xu, L., & Ding, Y. (2004). Automatic direct and indirect schema mapping: experiences and lessons learned. *SIGMOD Record, 33*(4), 14-19. doi: 10.1145/1041410.1041413

Evermann, J., & Wand, Y. (2005). Ontology based object-oriented domain modelling: fundamental concepts. *Requirements Engineering, 10*(2), 146-160. doi: 10.1007/s00766-004-0208-2

Evermann, J. (2009). A UML and OWL description of Bunge's upper-level ontology model. *Software & Systems Modeling, 8*(2), 235-249. doi: 10.1007/s10270-008-0082-3

Fagin, R., Kolaitis, P. G., Miller, R. J., & Popa, L. (2005a). Data exchange: semantics and query answering. *Theoretical Computer Science, 336*(1), 89-124. doi: 10.1016/j.tcs.2004.10.033

Fagin, R., Kolaitis, P. G., Popa, L., & Tan, W. (2005b). Composing schema mappings: second-order dependencies to the rescue. *ACM Transactions on Database Systems, 30*(4), 994-1055. doi: 10.1145/1114244.1114249

Fagin, R., Kolaitis, P. G., & Popa, L. (2005c). Data exchange: getting to the core. *ACM Transactions on Database Systems, 30*(1), 174-210. doi:10.1145/1061318.1061323

Fagin, R. (2007). Inverting schema mappings. *ACM Transactions on Database Systems, 32*(4), 25-53. doi: 10.1145/1292609.1292615

Fagin, R., Kimelfeld, B., Li, Y., Raghavan, S., & Vaithyanathan, S. (2011). Rewrite rules for search database systems. *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems,* Athens, Greece. 271-282. doi: 10.1145/1989284.1989322

Fonseca, F. (2007). The double role of ontologies in information science research: research articles. *Journal of the American Society for Information Science and Technology, 58*(6), 786-793. doi: 10.1002/asi.v58:6

Fuxman, A., Hernandez, M. A., Ho, H., Miller, R. J., Papotti, P., & Popa, L. (2006a). Nested mappings: schema mapping reloaded. *Proceedings of the 32nd International Conference on Very Large Data Bases,* Seoul, Korea. 67-78.

Fuxman, A., Kolaitis, P. G., Miller, R. J., & Tan, W. (2006b). Peer data exchange. *ACM Transactions on Database Systems, 31*(4), 1454-1498. doi: 10.1145/1189769.1189778

Gemino, A., & Wand, Y. (2004). A framework for empirical evaluation of conceptual modeling techniques. *Requirements Engineering, 9*(4), 248-260. doi: 10.1007/s00766-004-0204-6

George, D., & Preston, U. (2005). Understanding structural and semantic heterogeneity in the context of database schema integration. *Journal of the Department of Computing, UCLAN 4*(1), 29-44.

Glavic, B., Alonso, G., Miller, R. J., & Haas, L. M. (2010). TRAMP: understanding the behavior of schema mappings through provenance. *Proceedings of the VLDB Endowment, 3*(1-2), 1314-1325.

Haas, L. M. (2006). Beauty and the beast: the theory and practice of information integration. *Proceedings of the 11th International Conference on Database Theory,* Barcelona, Spain. 28-43. doi: 10.1007/11965893_3

Haas, L. M., Hentschel, M., Kossmann, D., & Miller, R. J. (2009). Schema AND data: a holistic approach to mapping, resolution and fusion in information integration. *Proceedings of the 28th International Conference on Conceptual Modeling*, Gramado, Brazil. 27-40. doi: 10.1007/978-3-642-04840-1_5

Hakimpour, F., & Geppert, A. (2001). Resolving semantic heterogeneity in schema integration. *Proceedings of the International Conference on Formal Ontology in Information Systems*, Ogunquit, ME, USA. 297-308. doi: 10.1145/505168.505196

Halevy, A. Y., Rajaraman, A., & Ordille, J. (2006). Data integration: the teenage years. *Proceedings of the 32nd International Conference on Very Large Data Bases*, Seoul, Korea. 9-16.

Hassanzadeh, O., Kementsietsidis, A., Lim, L., Miller, R. J., & Wang, M. (2009). A framework for semantic link discovery over relational data. *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, Hong Kong, China. 1027-1036. doi: 10.1145/1645953.1646084

Lu, M., Agrawal, D., Dai, B. T., & Tung, A. K. (2011). Schema-as-you-go: on probabilistic tagging and querying of wide tables. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Athens, Greece. 181-192.

Madhavan, J., Jeffery, S., Cohen, S., Dong, X. L., Ko, D., Yu, C., & Halevy, A. Y. (2007). Web-scale data integration: you can only afford to pay as you go. *Proceedings of the Biennial Conference on Innovative Data Systems Research*, Asilomar, CA, USA. 342-350.

Magnani, M., Rizopoulos, N., McBrien, P., & Montesi, D. (2005). Schema integration based on uncertain semantic mappings. *Proceedings of the 24th International Conference on Conceptual Modeling*, Klagenfurt, Austria. 31-46. doi: 10.1007/11568322_3

Magnani, M., & Montesi, D. (2007). Uncertainty in data integration: current approaches and open problems. *Proceedings of the VLDB Workshop on Management of Uncertain Data*, Vienna, Austria. 18-32.

Mandreoli, F., & Martoglia, R. (2011). Knowledge-based sense disambiguation (almost) for all structures. *Information Systems, 36*(2), 406-430. doi: 10.1016/j.is.2010.08.004

Marnette, B., Mecca, G., & Papotti, P. (2010). Scalable data exchange with functional dependencies. *Proceedings of the VLDB Endowment, 3*(1-2), 105-116.

Marnette, B., Mecca, G., Papotti, P., Raunich, S., & Santoro, D. (2011). ++Spicy: an open-source tool for second-generation schema mapping and data exchange. *Proceedings of the VLDB Endowment, 4*(12), 1438-1441.

Mecca, G., Papotti, P., & Raunich, S. (2012). Core schema mappings: scalable core computations in data exchange. *Information Systems, 37*(7), 677-711. doi: 10.1016/j.is.2012.03.004

Miller, R. J., Haas, L. M., & Hernández, M. A. (2000). Schema mapping as query discovery. *Proceedings of the 26th International Conference on Very Large Data Bases,* Cairo, Egypt. 77-88.

Milton, S. K. (2004). Top-level ontology: the problem with naturalism. *Proceedings of the International Conference on Formal Ontology in Information Systems,* Torino, Italy. 85-94.

Mutis, I., & Issa, R. R. (2012). Framework for semantic reconciliation of construction project information, *Journal of Information Technology in Construction*, *17*(1), 1-24

Orsi, G., & Pieris, A. (2011). Optimizing query answering under ontological constraints. *Proceedings of the VLDB Endowment, 4*(11), 1004-1015.

Pablo, B. (2009). Logical foundations of relational data exchange. *SIGMOD Record., 38*(1), 49-58. doi: 10.1145/1558334.1558341

Pankowski, T. (2013). Semantics preservation in schema mappings within data exchange systems. *Proceedings of the 16th International Conference on Knowledge Engineering, Machine Learning and Lattice Computing with Applications,* San Sebastian, Spain. 88-97. doi: 10.1007/978-3-642-37343-5_10

Parsons, J., & Wand, Y. (2000). Emancipating instances from the tyranny of classes in information modeling. *ACM Transactions on Database Systems, 25*(2), 228-268. doi: 10.1145/357775.357778

Parsons, J., & Wand, Y. (2003). Attribute-based semantic reconciliation of multiple data sources. *Journal on Data Semantics*, *2800*(1), 21-47. doi: 10.1007/978-3-540-39733-5_2

Parsons, J., & Cole, L. (2004). An experimental examination of property precedence in conceptual modelling. *Proceedings of the 1st Asian-Pacific Conference on Conceptual Modelling,* Dunedin, New Zealand. 101-110.

Parsons, J. (2011). An experimental study of the effects of representing property precedence on the comprehension of conceptual schemas. *Journal of the Association for Information Systems, 12*(6), 1.

Popa, L., & Tannen, V. (1999). An equational chase for path-conjunctive queries, constraints, and views. *Proceedings of the 7th International Conference on Database Theory,* Jerusalem, Israel. 39-57.

Popa, L., Velegrakis, Y., Hernández, M. A., Miller, R. J., & Fagin, R. (2002). Translating Web data. *Proceedings of the 28th International Conference on Very Large Data Bases,* Hong Kong, China. 598-609.

Pottinger, R., & Bernstein, P. A. (2008). Schema merging and mapping creation for relational sources. *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology,* Nantes, France. 73-84. doi: 10.1145/1353343.1353357

Qian, L., Cafarella, M. J., & Jagadish, H. V. (2012). Sample-driven schema mapping. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Scottsdale, AZ, USA. 73-84. doi: 10.1145/2213836.2213846

Raffio, A., Braga, D., Ceri, S., Papotti, P., & Hernandez, M. A. (2008). Clip: a visual language for explicit schema mappings. *Proceedings of the IEEE 24th International Conference on Data Engineering,* Cancún, México. 30-39. doi: 10.1109/ICDE.2008.4497411

Roth, M., & Tan, W. (2013). Data integration and data exchange: it's really about time. *Proceedings of the Biennial Conference on Innovative Data Systems Research,* Asilomar, CA, USA. 342-350.

Rull, G., Farré C., Teniente, E., & Urpí Toni. (2009). MVT: a schema mapping validation tool. *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology,* Saint Petersburg, Russia. 1120-1123. doi: 10.1145/1516360.1516492

Rull, G., Farré, C., Teniente, E., & Urpí, T. (2013). Validation of schema mappings with nested queries. *Computer Science and Information Systems, 10*(1), 79-104.

San, R. (2012). *Ventana research 2012 value index for data integration.* (Research). CA, USA: Ventana Research.

Shanks, G., Tansley, E., Nuredini, J., Tobin, D., & Weber, R. (2002). Representing part-whole relationships in conceptual modeling: an empirical evaluation. *Proceedings of the 23rd International Conference on Information Systems,* Barcelona, Spain. 89-100.

Shanks, G., Tansley, E., & Weber, R. (2004). Representing composites in conceptual modeling. *Communications of the ACM, 47*(7), 77-80. doi: 10.1145/1005817.1005826

Tatarinov, I., & Halevy, A. Y. (2004). Efficient query reformulation in peer data management systems. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Paris, France. 539-550. doi: 10.1145/1007568.1007629

ten Cate, B., Chiticariu, L., Kolaitis, P., & Tan, W. (2009). Laconic schema mappings: computing the core with SQL queries. *Proceedings of the VLDB Endowment, 2*(1), 1006-1017.

ten Cate, B., Kolaitis, P. G., & Tan, W. (2013). Schema mappings and data examples. *Proceedings of the 16th International Conference on Extending Database Technology*, Genoa, Italy. 777-780. doi: 10.1145/2452376.2452479

Wand, Y. (1996). Ontology as a foundation for meta-modelling and method engineering. *Information and Software Technology, 38*(4), 281-287. doi: 10.1016/0950-5849(95)01052-1

Wand, Y., Storey, V. C., & Weber, R. (1999). An ontological analysis of the relationship construct in conceptual modeling. *ACM Transactions on Database Systems, 24*(4), 494-528. doi: 10.1145/331983.331989

Wang, T., & Pottinger, R. (2008). SeMap: a generic mapping construction system. *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, Nantes, France. 97-108. doi: 10.1145/1353343.1353359

Chapter 3 SESM: Semantic Enrichment of Schema Mappings[1]

Abstract

Schema mapping is becoming pervasive in information integration through data exchange and data integration. In this thesis, we show that current schema mapping generation and verification techniques are not capable of reconciling some semantic heterogeneity because of ambiguities in interpreting different types of relations. To address this problem, we enrich the mapping generation process using conceptual models to recover the semantics of relations. The technique we propose in this chapter not only avoids generating ambiguous mappings, but also generates some new semantic mappings that are neglected in many schema mapping techniques because existing techniques ignore implicit associations between single and composite relations.

3.1 Introduction

Schema mappings are logical expressions representing relations between a source and a target schema used for either data integration or data exchange. Several different schema mapping systems have been developed to facilitate the process of data integration. Clio (Fagin et al., 2009; Miller et al., 2000; Popa et al., 2002) and HePToX (Bonifati et al., 2005) are two prominent examples of such systems. The prevailing approach in schema mapping generation has been based on attribute correspondences, where each correspondence represents a semantic relationship between an attribute of the target and an attribute in the source. Such correspondences can be created manually by users in a visual interface, or automatically by schema matchers (Bellahsene, 2011). Previous

[1] Sekhavat, Y. A., & Parsons, J. (2013). SESM: Semantic enrichment of schema mappings. *Proceedings of 4th ICDE International Workshop on Data Engineering Meets Semantic Web,* Brisbane, Australia. (to appear).

research in schema mapping shows that relying solely on simple attribute correspondence is not enough because schema mapping algorithms using such correspondences are incapable of reconciling some semantic heterogeneities. As discussed in (Alexe et al., 2011), no schema matching tool is capable of generating perfect matching and consequently, schema mapping algorithms based on them may produce inconsistent and erroneous mappings. In addition, they may generate ambiguous mappings where several alternative ways of mapping the source into the target are generated. Moreover, some semantic mappings are neglected by these algorithms.

We argue that such deficiencies in schema mapping result from the lack of high level information required to specify the semantics of schemas. In practice, usually human intervention is necessary to analyse and validate mappings. As a result, it is the user or subject matter expert's responsibility to inspect mappings and select preferred ones among a set of mappings. As an alternative option, some schema mapping techniques (e.g., (Alexe et al., 2011)) rely on sample source and target data examples to refine mappings and resolve ambiguities. However, the quality of refinement is strongly dependent on the quality of available data examples where they may not be perfectly representative of data sources. Consequently, human intervention is still required to judge the quality of mappings generated. In addition, data examples may not be available due to privacy or commercial limitations. Alternatively, a mapping verification technique is proposed in (Cappellari et al., 2010) by automatically checking mappings based on semantic annotations and tuning the mappings using a domain ontology. In this approach, the knowledge of experts (represented formally in a domain ontology) is used to refine

mapping expressions. Each mapping expression is evaluated separately where the verifier algorithms decide whether a given mapping is valid or not. However, as we discuss in the next section, considering mappings jointly may result in generating some new mappings that are not originally generated by common schema mapping algorithms such as Clio (Fagin et al., 2009; Miller et al., 2000; Popa et al., 2002). In other words, mapping verification does not necessarily entail generating complete mappings. We argue that domain knowledge must be considered at the outset of mapping generation rather than in post-processing of mappings. We show this approach not only prevents generating ambiguous mappings, but also generates some new semantic mappings that are not achievable in traditional approaches.

In this chapter, we use conceptual models as domain knowledge to capture semantics of schema elements. Generally, conceptual models (e.g., Entity Relationship or Class diagrams) represent domain knowledge in an implementation-independent fashion, and are commonly used to guide database design. Since a relational model representation (using tables containing attributes and relations between tables) is not expressive enough to show the semantics originally contained in a conceptual model, not all semantics in a conceptual model are preserved in a relational model representation. We argue that without high level knowledge to reconcile semantic heterogeneities, a schema mapping algorithm is not capable of identifying the semantics behind different types of relations represented using the same technique. This can result in ambiguity in interpreting the relations that consequently may lead to incorrect mappings.

In particular, our contributions in this chapter are as follows: To avoid generating ambiguous mappings, we propose a set of schema mapping generation algorithms based on Clio (Fagin et al., 2009; Miller et al., 2000; Popa et al., 2002), but in which relational schemas are enriched by semantics of conceptual models before schema mapping. This technique also explores some new plausible mappings that are not generated using Clio (Fagin et al., 2009; Miller et al., 2000; Popa et al., 2002). While existing proposals rely on the key role of user-feedback or data examples to verify mappings after mapping generation, our approach allows the direct reuse of expert knowledge. We argue that it would be much easier for a domain expert to prepare a conceptual schema (or domain ontology) than to judge the correctness of complex schema mappings and resolve ambiguous mappings after each mapping generation. We built a prototype system (SESM), and evaluate the technique by applying it on a case study including three schemas in the healthcare domain.

3.2 Need for Semantic Heterogeneity Reconciliation

The need for semantic reconciliation emerges because different data structures and elements may be used to realize the same concept. In particular, two relational data sources might differ in the choice of tables and relations between them. This heterogeneity results in ambiguity in interpreting relations that consequently leads to generating some ambiguous mappings.

Example 1: As discussed in (An et al., 2010), a generalization relation can be realized through different techniques in a relational model, as shown in Figure 3-1. Suppose the generalization relation shown in Figure 3-1(a) is implemented using

technique G3 in the source and technique G1 in the target. Without considering mapping refinement, Clio generates mp_1: $\forall x_1,..,x_5$ $Person(x_1, x_2, x_3, x_4, x_5) \rightarrow Employee(x_2, x_3)$, $Doctor(x_2, x_4)$ and mp_2: $\forall x_1,..,x_5$ $Person(x_1, x_2, x_3, x_4, x_5) \rightarrow Employee(x_2, x_3)$, $Nurse(x_2, x_5)$ as two independent mappings. However, the generalization relation in Figure 3-1(a) is a disjoint generalization indicating that only one of these mappings is applicable given a tuple in the source according to the value of *role* in the source. In addition, in the case of using technique G4 in the source, the first mapping is applicable when *specialty* is not specified in the source, and the second mapping is applicable when *degree* is not specified.

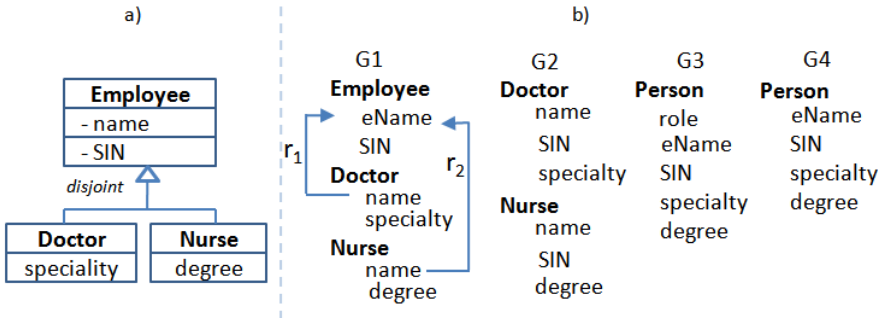


Figure 3-1: A generalization relation (a) and its different implementations (b)

Example 2: Consider two relational schemas in Figure 3-2 that represent patients examined by doctors. For this example, Clio generates:

mp_3: $\forall x_1,x_2\ Patient(x_1),ExaminedBy(x_1,x_2),Doctor(x_2) \rightarrow \exists x_3\ ExaminedIn(x_1,x_3)$

mp_4: $\forall x_1,x_2\ Doctor(x_1), WorksFor(x_1,x_2)Hospital(x_2) \rightarrow \exists x_3 ExaminedIn(x_3,x_2)$.

Although these are correct mapping expressions, given the sequence of relations in Figure 3-2, another mapping exists between the source and target considering the correlation between mp_3 and mp_4 which is mp_5: $\forall x_1,..,x_3$ $Patient(x_1), ExaminedBy(x_1, x_2)$,

Doctor(x_2) *WorksFor*(x_2, x_3), *Hospital*(x_3) → *ExaminedIn*(x_1,x_3). This problem is discussed in (An et al., 2007) as a *sequence of relations*.

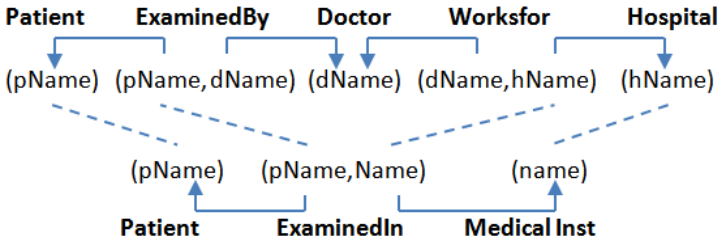


Figure 3-2: An example of sequence of relations in schema mapping

Abstracting from these examples because the same technique (i.e., using key/foreign keys to implement referential integrity constraints) is employed to implement functional and generalization relations, there are ambiguities in interpreting these relations. In addition, mapping expressions based on only class-level properties (e.g., name, role) are not expressive enough to convey the whole semantics of the mappings, and some specific properties (e.g., specialty = "surgeon") are required for this purpose. In addition, the inability to correlate mappings results in some typical problems including data duplication (i.e., multiple tuples are generated in the target for a specific tuple in the source), and loss of associations (i.e., tuples generated in the target are not associated correctly).

3.3 Related Work

There have been different approaches to add data semantics during schema mapping. A method is proposed in (Chiticariu et al., 2008) that generates mapping expression through interaction with users and gain the knowledge of users to refine intermediate schemas created during mapping generation. A validation tool that allows mapping designers to

decide if a given mapping expression has certain desirable properties is proposed in (Rull et al., 2009). This feedback is used to indicate whether the mappings adequately meet the intended requirements of users. Spider (Alexe et al., 2006) is a mapping debugging tool in which users select a set of target tuples, and the system shows how these tuples are transformed to target tuples through given mappings. In the system proposed in (Belhajjame et al., 2010) users annotate, select and refine schema mappings by commenting on results of queries evaluated using the mappings. Alternatively, mapping refinement is performed using data examples in (Alexe et al., 2011) where each data example is used as a partial specification of semantics to refine mappings. Spicy (Bonifati et al., 2008) refines schema mappings by comparing generated target instances with a sample target instance. In this system, mappings are ranked and finally selected by a mapping designer. In (Alexe et al., 2008), data examples are employed to help mapping designers in understanding and refining mapping expressions. This system interacts with users to infer desired grouping semantics and selecting best mappings among alternative interpretations of an ambiguous mapping. However, relying on a small set of data examples may not reveal all the potential pitfalls during mapping refinement. Recently, global domain knowledge has been used to refine and verify mappings. In (Cappellari et al., 2010) schema mappings are verified by checking whether each mapping expression is consistent with semantic annotations in source and target schemas. The technique we propose in this thesis is based on enhancing schema mappings by enriching schemas using domain knowledge. The main difference between our approach and mapping verification and refinement techniques based on domain ontologies is that we use domain

knowledge in the first phase of mapping generation to avoid ambiguities, rather than mapping refinement based on domain knowledge.

3.4 Schema mapping: from Clio to SESM

The technique we propose generates semantic mappings between relational schemas, which is defined as a finite collection of relations $R=\{R_1,..,R_k\}$ with fixed number of attributes in each relation. In a relational schema, integrity constraints are provided through key/foreign key constraints where each relation R is accompanied with some constraints Σ_R in terms of foreign keys denoted (R, Σ_R).

To generate schema mappings, Clio (Fagin et al., 2009; Miller et al., 2000; Popa et al., 2002) first identifies the logical relations in both the source and target schemas. Such logical relations (called paths) are extracted by enumerating all paths from the root to any intermediate class in a schema. For example, in Figure 3-1(b), PT_1: $\forall x,y$ *Employee(x, y)*, PT_2: $\forall x,y$ *Doctor(x, y)* and PT_3: $\forall x,y,t$ *Employee(x, y), Doctor(x, t)* are examples of paths for G1. We use the concept of path to define integrity constraints for mapping expressions which is formally defined as ($\forall PT_i \exists PT_j, E$) where PT_i and PT_j are paths, and E is a conjunction of equalities of the form $e_1=e_2$, in which e_1 is an expression depending on one of the variables of the PT_1 and e_2 is an expression depending on one of the variables of the PT_2. For example, the referential integrity constraint r_1 regarding the source schema in Figure 3-1(b) is: $\forall x, y$ *Employee(x, y),* $\exists x', y'$ *Doctor(x', y'), (x=x')*. Clio (Fagin et al., 2009; Miller et al., 2000; Popa et al., 2002) employs logical associations as well as a set of attribute correspondences to generate mappings. Formally, a correspondence between a property p_1 in path PT_1 and property p_2 in path PT_2 is a triple

$\langle PT_1, PT_2, p_1=p_2 \rangle$ which is denoted: $p_1 \leftrightarrow p_2$. Logical associations allow compiling attribute correspondences in a meaningful way. The mappings are generated based the intuition that when there exists a set of correspondences between source and target schemas, if all elements of the source all occur in an association of the source schema, and all elements of the target all occur in the same association of the target schema, these correspondences can be processed together.

3.4.1 SESM: Semantic Enrichment of Schema Mapping

We use a similar approach used in Clio (Fagin et al., 2009; Miller et al., 2000; Popa et al., 2002) to capture logical associations, with the difference that schemas are annotated with the semantics of conceptual models before mapping generation. For this purpose, using conceptual models corresponding to relational models in source and target, foreign keys are annotated with generalization (g) or functional (f) relations. Mapping between a conceptual model and a relational schema is a different research problem which is beyond the scope of this thesis. Without loss of generality and in order to simply explain the algorithm, we assume the same name is used for concepts (entities) in conceptual models and relations (tables) in a relational schema. In the case of a generalization relation, this algorithm finds a set of entity pairs (a_1, a_2) where a_1 is a subclass and a_2 is a superclass. Then, for each pair, the algorithm finds corresponding tables and marks foreign keys implementing these relations as generalization (g). Accordingly, foreign keys implementing functional relations are labeled as (f). The details of this process are shown in Algorithm I.

Algorithm I (Schema Annotator): Preprocessing a schema based on a conceptual model

---

Input: Relational schema RS=(R, null), Conceptual model CM
Output: Annotated relational schema AR= (R, Σ_R=null)
1: S ← extract the set of relations in CM
2: **foreach** relation r **in** S
3: Extract entity pair (a_1, a_2) where r is the relation between a_1 and a_2
4: such that a_1 references a_2
5: T← Type of the r in CM //that can be functional (f) or generalization (g)
6: (x_1, x_2) ← find corresponding entities in RS such that there exists a
7: foreign key between x_1, and x_2 having x_1=a_1 and x_2=a_2
8: Mark the foreign key related to this relation as T
9: Add the path related to this foreign key (i.e., ∀ PTi ∃PTj, (E, T)) to Σ_R
10: **Return** AR

We use different techniques to generate logical associations based on generalization and functional relations. In the case of functional relations, the Chase algorithm is applied on a path *PT* based on a referential integrity constraint *(*$\forall PT_1 \exists PT_2$, *E*) where in each step, when PT_1 is subsumed in *PT* but PT_2 is not subsumed in *PT*, a new path with PT_2 added to existing associations (Fagin et al., 2009). In the case of generalization, we use a modified version of the Chase that finds maximal generalization associations. A *maximal generalization association* is defined as a path from the highest superclass to a lower subclass. For a given generalization relation g_1, Algorithm II checks if there exists another generalization g_2 such that the superclass of g_2 is the subclass of g_1. Once the algorithm finds a g_2 with this condition, (g_1, g_2) is added as a logical generalization association. In the next round, the last item in each sequence is checked with other generalization relations to see if it can be expanded. The algorithm continues

until no generalization relation can be added to the existing set of associations. The final set is returned as a maximal set of generalization associations.

Algorithm II (Association Generator): Generating logical generalization associations

---

Input: Annotated relational schema AR
Output: A set of logical generalization associations GA
1: S ← set of relations in AR marked with g (indicating generalization)
2: GAcollection ← null
3: **foreach** relation r **in** S
4: if r= (e_1, e_2) and e_1 is a root class, then add (e_1, e_2) to GAcollection
5: **do** {GA ← GAcollection
6: GAcollection ← null;
7: **foreach** (e_1, e_2) in GA
8: **If** there exist e_3 in AR such that e_3 is a subclass of e_2, **then**
9: Add (e_1, e_2, e_3) to GAcollection
10: **Else** Add (e_1,e_2) to GAcollection
11: }**while** (GA != GAcollection)
12: **return** GA

Clio (Fagin et al., 2009; Miller et al., 2000; Popa et al., 2002), uses the triple $\langle A^S, A^T, E\rangle$ to represent a mapping expression where A^S and A^T are logical associations in the source and target schemas, respectively, and E is a conjunction of equalities between class level properties of A^S and A^T. However, as discussed in Section 3.2, class-level properties are not capable of conveying the full semantics of mapping as some mappings exist based on possessing specific properties. To address this problem, we use the concept of *manifestation* which is a specific value $v \in D$ assigned to property p where D is the domain of values of p. Using this concept, we employ a new structure for mappings supporting specific properties called *manifestation-based mapping*.

Definition 1: A *manifestation-based mapping (mp)* is a statement in form of: *mp*: A^S ***in*** *S*, A^T ***in*** *T [**with** M]*

In this formula, A^S and A^T are logical associations in source (S) and target (T), respectively. A mapping statement can have a set of manifestations *M* (i.e., constraints in terms of possessing specific properties) that represents conditions under which a mapping statement exists. The semantics of a mapping statement is that for a logical association A^S in source, there exists a logical association A^T in target regarding manifestations in *M*.

3.4.1.1 Ontological Foundations

Upper-level ontology has been widely used as a theoretical foundation for conceptual modeling, both in theoretical analyses (Parsons & Wand, 2000) and in empirical studies (Gemino & Wand, 2004). Using ontology, we aim to address some problems in schema mapping that have origins in specifications used to represent data in heterogeneous data sources. Resolving ambiguities in schema mapping requires carefully studying structures and relations between them in data models. In (Sekhavat, 2012; Sekhavat & Parsons, 2012), we showed how ontological foundations can be used for semantic heterogeneity reconciliation in data integration. In this thesis, we use these foundations to enhance schema mapping. In particular, we employ some aspects of Bunge's ontology (Bunge, 1977) to resolve semantic heterogeneities and reconcile ambiguities. According to this ontology, a domain of interest is a set of things, where each thing possesses at least one property (the term *thing* refers to any phenomenon exists in the domain in a specific period of time). *Properties* are the basic constructs in Bunge's ontology that represent the characteristics of existing things in the real world. In particular, we use *property precedence* and *composite* notions in Bunge's ontology to provide foundations for enhanced schema mapping.

Property precedence in Bunge's ontology provides a basis to study relations between properties in information and database systems (Wand & Weber, 1990). Property P_1 precedes property P_2 (denoted: $P_2 \rightarrow P_1$) if and only if the set of things possessing P_2 is a subset of things possessing P_1 (e.g., the property 'having gender' precedes the property 'being female' since the set of instances that are female is a subset of instances that have gender). We use this concept for mapping generation to infer general properties of a class from its specific properties for semantic heterogeneity reconciliation purposes. In (Sekhavat & Parsons, 2012), we showed how property precedence relations in a schema can be used to infer implicit properties and enhance mapping expression.

3.4.1.2 Mapping Generation for Generalizations Relations

Abstracting from the definition of property precedence, a generalization relation is a special case of property precedence. According to this definition, when property p_1 precedes property p_2, everything possessing p_2 also possess p_1. This implies that possessing p_1 by an instance can be inferred from the fact it possesses p_2. In the same way, in a generalization relation, possessing a specific property of a subclass implies possessing general properties of its superclass. In other words, properties of a superclass precede specific properties of the subclasses. For example, in Figure 3-1(a), having *SIN* precedes *specialty* of a doctor, or in other words, possessing *specialty* implies possessing *SIN*. To recover the concept of generalization, we use correspondences between properties in source and target schemas as well as property precedence relations in the target. More specifically, it is possible to infer a property precedence relation between two properties in the source if there exists a property precedence relation between their

corresponding properties in the target. The details of the process to generate mapping expressions based on generalization are shown in Algorithm III.

Algorithm III (Mapping Generator): Generating mapping expressions based on generalization

---

Input: Set of logical generalization associations GA
Relational schema in the source R_S and target R_T
Set of property correspondences CR
Output: Set of mapping MP

// Step1 (using lemma 1 to recover property precedence relations)
foreach property precedence pp = $p'_2 \rightarrow p'_1$ in R_T
  foreach relation rs in the source
    If there exist $p_1, p_2 \in$ rs such that $p_1 \leftrightarrow p'_1$ and $p_2 \leftrightarrow p'_2$ **then**
      Add $p_2 \rightarrow p_1$ to Σ_{RS}
//Step2 (creating mappings based on logical generalization associations)
foreach generalization association g in R_T
  sb ← the final subclass of g
  P←set of properties in sb that are not in their superclass
  foreach relation rs in the R_S
    If there is property precedence $p_2 \rightarrow p_1$ in Σ_{RS} such that p_1 has
      a correspondence c_1 in CR between properties p_1 and p'_1
      in superclass of sb, and a correspondence c_2 between p_2
    and p'_2 in sb **then**
      If there is a property in rs indicating the type of subclass *// G3*
        Create manifestation M regarding the value representing
          the subclass
      else *// G4*
        Create manifestation M such that for each property p in rs
          representing other subclasses p=null is added to M
        add mapping [mp: rs in S, g in T with M] to MP
return MP

Lemma 1: let p_1 and p_2 denote two properties in the source, and p'_1 and p'_2 two properties in the target. The relation $p_2 \rightarrow p_1$can be inferred from $p'_2 \rightarrow p'_1$ if there exist correspondences $p_1 \leftrightarrow p'_1$ and $p_2 \leftrightarrow p'_2$.

Proof: A correspondence $p_1 \leftrightarrow p'_1$ implies that there exists a path PT_1 with property p_1 and PT'_1 with property p'_1 such that $p_1=p'_1$. Accordingly, $p_2 \leftrightarrow p'_2$ implies that there

exists a path PT_2 and PT'_2 such that $p_2=p'_2$. As a result, by replacing the equivalent properties in $p'_2 \rightarrow p'_1$ (p_1 with p'_1, and p_2 with p'_2), we can infer $p_2 \rightarrow p_1$. Lemma 1 is depicted in Figure 3-3.

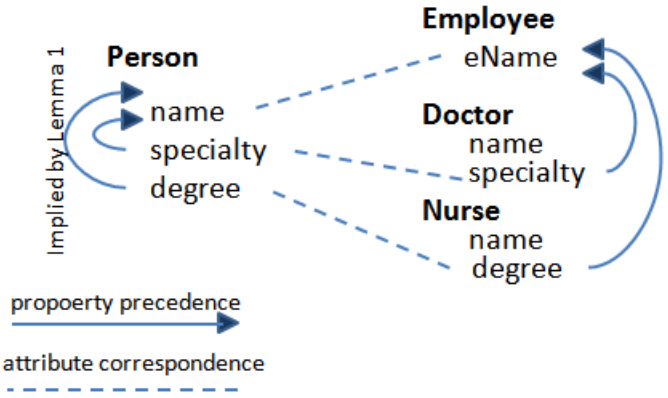


Figure 3-3: Graphical representation of Lemma 1

3.4.1.3 Mapping Generation for Sequence of Relations

According to Bunge's ontology (Bunge, 1977), things can be combined to form a "composite". Composites possess at least one "emergent property" not possessed by any component. An emergent property is a property that is derived from properties of its parts, but it is not reducible to them. For example, considering "a student who is enrolled in a course" as a composite, the *grade* of a student is an emergent property of this composite while student's age and course name are "hereditary" properties. According to Bunge's ontology, every composite possesses at least one emergent property that is not inherited from any of the parts (Bunge, 1977). As discussed in (Shanks et al., 2004), a composite thing can be represented as a separate entity class, or a relationship class (associations) in a conceptual model.

When two entities and an association between them represent a composite, the properties of this association can be considered as emergent properties of the composite (i.e., not properties of its parts), and the properties of parts can be considered as hereditary properties of this composite. For example, as shown in Figure 3-4, considering a "doctor who works for a hospital" as a composite, the hereditary properties of this composite will be the properties of a doctor and a hospital, and the mutual properties between a doctor and a hospital (the properties of *worksFor*) will be the emergent properties of this composite. Considering a "doctor who work for a hospital as a composite", and the *examined* relation between patient and this composite, there is an indirect relation between a patient and a hospital (as a part of this composite).

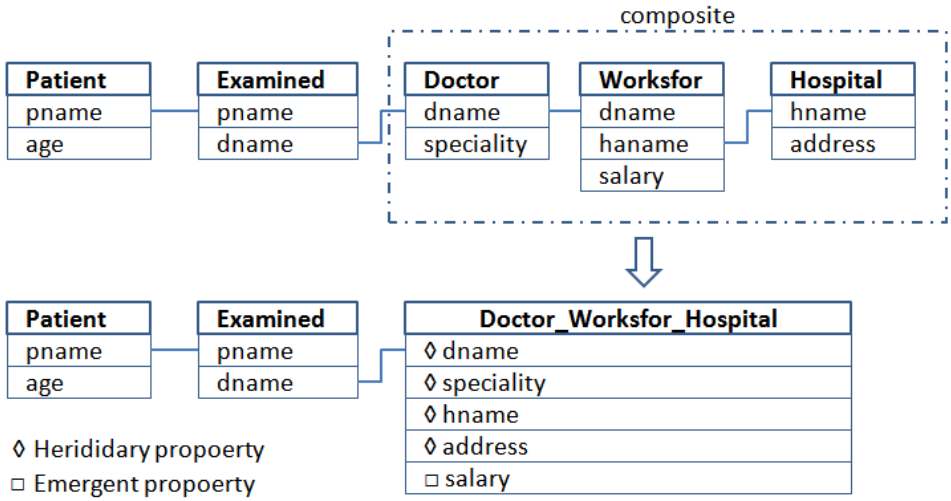


Figure 3-4: forming a composite from two components

To generate new mappings based on the concept of sequence of relations, the algorithm proposed in this chapter generates composite in the first step as shown in Algorithm IV. Starting from simple entities and functional relations between them, this algorithm generates composites by adding the properties of parts as hereditary properties, and the

properties of a relation between parts as emergent properties. Formally, we define a *composite* as a combination of two entities E_1 (with properties P_1), and E_2 (with properties P_2) and relations between them r (with properties P_r) and denoted: (P_h, P_e) where P_h = $P_1 \cup P_2$ is the set of hereditary properties, and $P_e = P_r$ ($P_e \neq \emptyset$) is the set of emergent properties.

Algorithm IV (Mappings for Composites): Generating mappings based on a sequence of relations

---

Input: Relational schema in the source R_S and target R_T
Set of property correspondences CR
Output: Set of mappings MP
*1: // **Step1** (generating composites)*
2: S ←a subset of relations in R_S having at least one extra property except
3: the IDs of entities connected through this relation
4: **foreach** relation r **in** S
5: (e_1, e_2) ← a pair of entities that are connected through r
6: create comp = (properties of $e_1 \cup$properties of e_2, properties of r)
7: Add comp to C_S *// set of composites in the source*
8: Repeat Step 1 for R_T and generate T and C_T
9: *// **Step2** (generating mappings from a set of entities and composites)*
10: AS (and AT) ← logical associations generated from entities and C_S
11: (and C_T) in the source (target)
12: A (and B)← a subset of associations in AS (and AT) such that covers at
13: least one source (target) path in correspondences of CR
14: **foreach** pair <a, b> **in** A×B
15: C ← {c| c∈ CR and c is covered by <a, b>}
16: If C = null then continue;
17: Let C= {c_1,...,c_m}
18: **foreach** ci in C
19: let e the equality in ci
20: update variables of a and b according to e
21: m = a in S b in T ; MP←MP ∪ {m}
22: **Return** MP

3.5 Experience

We built a prototype system (SESM) and conducted a case study to indicate the effectiveness of our algorithms. The current version of SESM does not have a visual

interface, and relational schemas including entities, properties and constraints as well as conceptual models, property precedences, and attribute correspondences, are provided as text files for the system. SESM is compared with ++Spicy (Marnette et al., 2011) which is an open source implementation of Clio's schema mapping algorithms, in which schema matching is built in schema mapping process. Through this study, we aim to indicate to what extent mappings generated by SESM are capable of resolving ambiguities, and if there is a sufficient number of such mappings to make a considerable improvement in data integration process. We employed three real-world physical relational schemas in the healthcare domain including HSRC, PHINand NEDSS that are different physical implementations of HL7 (Health Level 7) reference model including 48, 36 and 42 tables, respectively. Considering each of these schemas as a source or target, we formed a setting including six states in the form of $schema_1$-$schema_2$ in which $schema_1$ and $schema_2$ are source and target schemas, respectively. We identified two different scenarios C_1 (a generalization relation is realized through technique G3 in $schema_1$, and technique G1 in $schema_2$), C_2 (a generalization relation is realized through technique G4 in $schema_1$, and technique G1 in $schema_2$). According to examples discussed in Section 3.2, these scenarios are error prone scenarios that result in generating ambiguous mappings. HSRC, PHIN and NEDSS include 15, 24 and 20 generalization relations, respectively. As shown in Table 3-1, for the scenarios C_1 and C_2, a significant number of ambiguous mappings is generated by ++Spicy when the semantics of generalization relations are ignored. For example, in HSRC, relation *Individual* is realized through technique G4 (see Figure 3-1) in which table *Entity* is used to store Person and nonPerson organism. On the other hand,

two separate tables (i.e., *Person*, *nonPerson*) are used in PHIN where common properties are stored in *Entity* table. In this case, ++Spicy generates two ambiguous mappings (one for mapping *Person* to *Entity* and one for mapping *nonPerson* to *Entity*) that must be resolved manually by a mapping designer. However, SESM generates two different mappings including specific properties indicating which mapping must be applied in data integration regarding the value those properties. Note the focus of this evaluation is on the ability to handle ambiguity in the presence of generalization relations. In the remaining scenarios, similar mappings are generated for both systems.

To understand difference between SESM and ++Spicy in schema mapping, we review an example. Suppose the source schema includes relation Instructor (Name, STNo, EMPNo), and the target includes relations Grads (name, STNo) and Profs (name, EmpNo). Given these schemas and property correspondences between *name* in source (Instructor) and *name* in target (Grad and Prof) as well as correspondences between STNo and EMPNo in the source and target, the following schema mapping expressions m_1, m_2 are generated by ++Spicy.

***m_1**: for each x_1, x_2, x_3: Instructor (Name: x_1, STNo: x_2, EMPNo: x_3) → Grad (name: x_1, STNO: x_2).*

***m_2**: for each x_1, x_2, x_3: Instructor (Name: x_1, STNo: x_2, EMPNo: x_3) → Prof (name: x_1, EMPNO: x_3).*

Given a source instance [*Instructor(I_1, st_1, null), Instructor(I_2, null, emp_1)*], ++Spicy uses these mappings to generate the target instance:

[*Grad(I_1, st_1), Grad(I_2, null), Prof(I_1, null), Prof(I_2, emp_1)*].

The problem is that for a given tuple in *Instructor*, ++Spicy generates two different mappings while only one of them is acceptable according to *STNo* and *EMPNo*. This ambiguity between m_1 and m_2 results in generating redundant information in the target while *Grad*(I_2, *null*) and *Prof*(I_1, *null*) are unacceptable.

On the other hand, SESM generates the mappings n_1 and n_2 where:

$\boldsymbol{n_1}$*: for each* x_1, x_2, x_3*: Instructor (Name:* x_1*, STNo:* x_2*, EMPNo:* x_3*)* $\rightarrow$ *Grad (name:* x_1*, STNO:* x_2*) with EMPNO = null*

$\boldsymbol{n_2}$*: for each* x_1, x_2, x_3*: Instructor (Name:* x_1*, STNo:* x_2*, EMPNo:* x_3*)* $\rightarrow$ *Prof (name:* x_1*, EMPNO:* x_3*) with STNo = null*

In these mappings, *EMPNO = null* and *STNo = null* provide conditions under which a mapping is applied. Such mappings avoid generating redundant tuples *Grad*(I_2, *null*) and *Prof*(I_1, *null*) for the source instance [*Instructor*(I_1, st_1, *null*), *Instructor*(I_2, *null*, emp_1)] in comparison to ++Spicy.

Table 3-1: Ambiguous mappings generated in scenarios C1 and C2

Source-target	**++Spicy**		**SESM**	
	C_1	C_2	C_1	C_2
HSRC-PHIN	9	2	-	-
HSRC-NEDSS	8	6	-	-
PHIN-HSRC	-	8	-	-
PHIN-NEDSS	15	8	-	-
NEDSS-HSRC	14	4	-	-
NEDSS-PHIN	7	13	-	-

The number of functional relations between a single entity and a composite within a schema indicates to what extent that schema has potential to generate mappings based on *sequence of relations*. As shown in Table 3-2, a considerable number of mappings are generated based on this concept in all six settings that shows the importance of identifying composites and relations between them for capturing semantic relations among entities. The importance of generating these new mappings is identifying more

complete associations between source and target schemas. Such complete set of mappings prevents generating redundant tuples when these mappings are used to generate the target instance. Repetitive composites are pruned by assigning a unique identifier for each entity and assigning a concatenation of ids of parts to each composite.

Table 3-2: New mappings generated based on sequence of relations

Source-Target	HSRC-PHIN	HSRC-NEDSS	PHIN-HSRC	PHIN-NEDSS	NEDSS-HSRC	NEDSS-PHIN
++Spicy	-	-	-	-	-	-
SESM	36	31	36	45	31	45

3.6 Conclusion and Future Work

In this chapter, we discussed how ignoring the semantics of generalization relations can results in generating ambiguous mappings. Unlike many existing works that try to resolve ambiguities and verify mappings after mapping generation, we proposed a set of algorithms that address this problem by preprocessing relational schemas and recovering the semantics of generalization relations based on conceptual models and property precedences. We used the property precedence notion in Bunge's ontology (Bunge, 1977) to differentiate between class-level and specific properties when properties are represented at the same level in the same relation. We also used the concepts of *composites* and *emergent properties* as an ontological foundation for the problem of *sequence of relations*. Using these concepts, we showed how indirect relations between entities in a schema can be employed to create some new plausible semantic mappings that cannot be identified in traditional techniques (Marnette et al., 2011). We built a text-based version of our algorithm (SESM), and used a case study in the healthcare domain to

show the effectiveness of our approach in heterogeneity avoidance without need for human intervention after generating mappings.

One direction for future work involves exploring other implicit semantics in conceptual models to generate new plausible mappings. Using this technique, we can expect to generate some new complex mapping expressions that cannot be generated by current mapping techniques.

3.7 References

Alexe, B., Chiticariu, L., & Tan, W. (2006). Spider: A schema mapping debugger. *Proceedings of the 32nd International Conference on Very Large Data Bases,* Seoul, Korea. 1179-1182.

Alexe, B., Chiticariu, L., Miller, R. J., & Wang-Chiew Tan. (2008). Muse: mapping understanding and design by example. *Proceedings of the IEEE 24rd International Conference on Data Engineering,* Cancún, México. 10-19. doi: 10.1109/ICDE.2008.4497409

Alexe, B., ten Cate, B., Kolaitis, P. G., & Tan, W. (2011). Designing and refining schema mappings via data examples. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Athens, Greece. 133-144. doi: 10.1145/1989323.1989338

An, Y., Borgida, A., Miller, R. J., & Mylopoulos, J. (2007). A semantic approach to discovering schema mapping expressions. *Proceedings of the IEEE 23rd International Conference on Data Engineering,* Istanbul, Turkey. 206-215. doi: 10.1109/ICDE.2007.367866

An, Y., Hu, X., & Song, I. (2010). Maintaining mappings between conceptual models and relational schemas. *Journal of Database Management, 21*(3), 36-68.

Belhajjame, K., Paton, N. W., Embury, S. M., Fernandes, A. A. A., & Hedeler, C. (2010). Feedback-based annotation, selection and refinement of schema mappings for dataspaces. *Proceedings of the 13th International Conference on Extending Database Technology,* Lausanne, Switzerland. 573-584. doi: 10.1145/1739041.1739110

Bellahsene, Z., Bonifati, A. & Rahm, E. (2011). *Schema Matching and Mapping*. Berlin, Heidelberg: Springer-Verlag.

Bonifati, A., Chang, E. Q., Lakshmanan, A. V. S., Ho, T., & Pottinger, R. (2005). HePToX: marrying XML and heterogeneity in your P2P databases. *Proceedings of the 31st International Conference on Very Large Data Bases*, Trondheim, Norway. 1267-1270.

Bonifati, A., Mecca, G., Pappalardo, A., Raunich, S., & Summa, G. (2008). The spicy system: Towards a notion of mapping quality. *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data,* Vancouver, Canada. 1289-1294. doi: 10.1145/1376616.1376757

Bunge, M. (1977). *Treatise on Basic Philosophy*: *the Furniture of the World*. Boston, MA: Reidel.

Cappellari, P., Barbosa, D., & Atzeni, P. (2010). A framework for automatic schema mapping verification through reasoning. *Proceedings of the IEEE Data Engineering Workshops,* Long Beach, CA, USA, 245-250. doi: 10.1109/ICDEW.2010.5452703

Chiticariu, L., Kolaitis, P. G., & Popa, L. (2008). Interactive generation of integrated schemas. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Vancouver, Canada. 833-846. doi: 10.1145/1376616.1376700

Fagin, R., Haas, L. M., Hernández, M., Miller, R. J., Popa, L., & Velegrakis, Y. (2009). Conceptual modeling: foundations and applications. In A. Borgida, er T., V. K. Chaudhri, P. Giorgini & E. S. Yu (Eds.), *Essays in Honor of John Mylopoulos* (pp. 198-236). Berlin, Heidelberg: Springer-Verlag. doi: 10.1007/978-3-642-02463-4_12

Gemino, A., & Wand, Y. (2004). A framework for empirical evaluation of conceptual modeling techniques. *Requirements Engineering, 9*(4), 248-260. doi: 10.1007/s00766-004-0204-6

Marnette, B., Mecca, G., Papotti, P., Raunich, S., & Santoro, D. (2011). ++Spicy: an open-source tool for second-generation schema mapping and data exchange. *Proceedings of the VLDB Endowment, 4*(12), 1438-1441.

Miller, R. J., Haas, L. M., & Hernández, M. A. (2000). Schema mapping as query discovery. *Proceedings of the 26th International Conference on Very Large Data Bases,* Cairo, Egypt. 77-88.

Parsons, J., & Wand, Y. (2000). Emancipating instances from the tyranny of classes in information modeling. *ACM Transactions on Database Systems, 25*(2), 228-268. doi: 10.1145/357775.357778

Popa, L., Velegrakis, Y., Hernández, M. A., Miller, R. J., & Fagin, R. (2002). Translating Web data. *Proceedings of the 28th International Conference on Very Large Data Bases,* Hong Kong, China. 598-609.

Rull, G., Farré C., Teniente, E., & Urpí Toni. (2009). MVT: a schema mapping validation tool. *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology,* Saint Petersburg, Russia. 1120-1123. doi: 10.1145/1516360.1516492

Sekhavat, Y. A. (2012). Semantic heterogeneity reconciliation in data integration. *Proceedings of the PhD Workshop of 38th International Conference on Very Large Data Bases,* Istanbul, Turkey. 19-24

Sekhavat, Y. A., & Parsons, J. (2012). Semantic schema mapping using property precedence relations. *Proceedings of the IEEE 6th International Conference on Semantic Computing,* Palermo, Italy. 210-217. doi: 10.1109/ICSC.2012.24

Shanks, G., Tansley, E., & Weber, R. (2004). Representing composites in conceptual modeling. *Communications of the ACM, 47*(7), 77-80. doi: 10.1145/1005817.1005826

Wand, Y., & Weber, R. (1990). Mario Bunge's ontology as a formal foundation for information systems concepts. In Weingartner, P., Dorn, & G. J. W. (Ed.), *Studies on Mario Bunge's Treatise* (pp. 123-149). Atlanta: Rodopi.

Chapter 4 Semantic Schema Mapping and Configurable Data Integration Using Property Precedence Relations[1]

Abstract

Information integration is a challenging issue in information management. Generally, data integration is performed through schema mapping representing high level information between heterogeneous data sources. Such mappings are generated using direct correspondences between data elements of source and target schemas, while other semantic relations are neglected. In this chapter, we focus on semantic heterogeneity reconciliation in schema mapping generation and data integration. We first use local property precedence relations as fundamental semantic relations between properties within source and target to semantically enhance schema mappings. Then, we use global property precedence relations between source and target elements to achieve Configurable Data Integration (CDI). In this configurable setting, two different query expansion algorithms are proposed allowing tradeoff between accuracy and completeness in query answering. Experiments using a working prototype of CDI show the feasibility of using this approach in various data integration scenarios.

4.1 Introduction

Distributing data for security, efficiency, reliability, or other purposes has resulted in islands of data, each of which may hold only part of the data required to fulfill information requests. This has led to a growing need for information integration across heterogeneous sources. Two prevailing approaches in information integration are data

[1] An extended version of: Sekhavat, Y. A., & Parsons, J. (2012). Semantic schema mapping using property precedence relations. *Proceedings of ICSC'12 IEEE International Conference on Semantic Computing, Palermo, Italy*, 210-217.

exchange, in which data structured under a source schema is converted to data under target schema, and data integration, in which integration is performed by querying across multiple autonomous and heterogeneous data sources. Data integration and data exchange are usually performed based on logical expressions called *schema mappings* that represent the relationships between different data sources independent of implementation (Fagin et al., 2009a; Fuxman et al., 2006a; Fuxman et al., 2006b). Such mappings help to better understand and reason about the relationships between data sources, and they can automatically be compiled to executable scripts in either data integration or data exchange.

One important issue in data integration is properly handling semantic heterogeneities. Generally, data integration is performed in a heterogeneous environment including a set of data sources that are designed and managed independently. Such an environment typically results in semantic heterogeneity in which the same concept can be represented in different ways. As a result, there might be values in different data sources that seem different but are semantically equivalent (Sekhavat & Parsons, 2012). Inability to reconcile semantic heterogeneity results in improper interpretation of concepts that consequently prevents proper data integration. In spite of substantial progress in schema mapping techniques for data integration (Fagin et al., 2009; Popa et al., 2002), semantic heterogeneity reconciliation is not well understood. As argued in (Haas, 2006), the lack of a deep understanding of the meaning of data and operations to integrate data has been the main problem for semantic heterogeneity reconciliation. In addition, the lack of a theoretical foundation behind semantic heterogeneity reconciliation techniques has also

resulted in limited progress in this area (Parsons & Wand, 2003). In (Sekhavat, 2012), we provided theoretical foundations for semantic heterogeneity reconciliation using Bunge's ontology (Bunge, 1977). Then, we developed schema mapping techniques using these theoretical foundations (Sekhavat & Parsons, 2012). In this chapter, we show how data integration systems based on query processing can exploit such mappings to improve semantic data integration.

Generally, two types of information are required to generate schema mappings: relations between data items within a data source (through finding referential integrity constraints), and relations between data items among different data sources (direct correspondences between the same properties in different data sources) (Bonifati et al., 2011).

In this chapter, we argue that schema mapping techniques based on simple property correspondences are not capable of handling some forms of semantic heterogeneity. Such techniques have largely dealt with cases where correspondences between properties are established between two properties representing the same concept in two different data sources. Due to semantic heterogeneity, the same concepts can be shown using different representations. In addition, other semantic relations between properties are largely ignored. As shown in (Parsons & Wand, 2003) structurally different attributes can manifest the same higher level property that consequently can be used for semantic heterogeneity reconciliation. For example, "high GPA" and "large number of publications" can manifest "scholarship eligibility" even though these are conceptually two different properties. We argue both direct and indirect inferences can be used to bind

two different data sources. First, it is possible to infer an implicit property of a thing from its explicit, but completely different, properties. For example, the gender of a patient suffering from a gender specific disease can be inferred from that specific disease even though the gender property does not exist explicitly (e.g., patients who suffer from *prostatitis* are *male*). Second, it is also possible to infer a property of a thing from a property of another thing. For example, the specialty of a doctor (e.g., *eye specialist*) can be inferred from activities of that doctor (e.g., *lasik surgery*). We argue that identifying implicit properties of instances through inference can improve schema mapping relative to common schema mapping techniques that consider only explicit properties.

In this chapter, we first discuss how schema mapping algorithms can exploit such inferences to reconcile semantic heterogeneities. Since the syntactic representation of concepts in different data sources does not necessarily comply with the semantics of those concepts, using auxiliary information illustrating such correspondence is required. For this purpose, we employ one of the fundamental relations between properties based on Bunge's ontology (Bunge, 1977) called *property precedence* relations described in Section 4.3. Using such relations enables the creation of some new semantically enhanced mappings that are not achievable in existing techniques (Fagin et al., 2009; Miller et al., 2000; Popa et al., 2002). To address ambiguous mappings where it is not possible to decide which mappings should be employed solely based on metadata, we propose a data-metadata oriented technique that takes into account data level heterogeneities as well as schema level heterogeneities. In this technique, mapping expressions are specified in terms of specific properties (values assigned to properties) as well as class level

properties (e.g., *having color* is a class level property while *having red color* is a specific property).

Then, we use mappings enhanced using inferences to integrate data through query rewriting. In query rewriting, a query posed to the target is rewritten to sub queries applicable on the source without materializing the source data in the target. Unlike many common query rewriting techniques that look for similar structures in different data sources (Fagin et al., 2009), we propose a technique that binds similar properties in different data sources that are manifested at different levels of abstractions. In particular, we study what it means to accurately or completely answer a target query given that the property precedence relations between properties are specified between different data sources. An important implication of this approach would be a data integration system that allows tradeoff between accuracy and completeness. This approach is called Configurable Data Integration (CDI) in which accuracy can be compromised to achieve more complete results. Unlike many rewriting systems that are lossless and consist of statements asserting that some portion of data is *equal* to some other portion of the data, the configurable query rewriting algorithm proposed in this thesis allows users to balance between accuracy and completeness. Completeness can be an important consideration in applications where the results of a query undergo further processing or analysis to support business intelligence, during which time unsound results generated in initial integration phase may be removed. For example, an intelligent application looking for security threats may compromise accuracy to achieve a complete list of potential intrusions and threats.

In the following, we first review the common existing approach for data integration through property correspondences. In Section 4.3.1, we discuss ontological foundations for semantic heterogeneity reconciliation, elaborate property precedence relations, and show how different types of property precedence relations can be used to reconcile different kinds of semantic heterogeneities. Then, we propose mapping generation algorithms using auxiliary information in terms of property precedence relations. In Section 4.4.3, we propose query rewriting algorithms using these mappings. We conclude by evaluating, analyzing and discussing implications of the techniques proposed for schema mapping and query rewriting.

4.2 Data Integration based on Property Correspondences

Data integration is achieved through query answering that aims to compute the set of answers of a query posed to the target by rewriting this query to a set of subqueries applied on the source. The first step toward query rewriting is to find how the elements of source and target relate to each other. Typically, this relationship is provided through property correspondences in which an atomic element (property) of a source schema is mapped to an element of the target schema. A correspondence shows an equivalence relationship between two properties that can be provided manually by a user, or automatically through schema matching techniques (Bonifati et al., 2011). For example, as shown in Figure 4-1, $c_1 \ldots c_6$ are property correspondences between source and target where c_i represents an equivalency between two properties in the source and the target. Correspondences, as well as referential integrity constraints, in the source and the target are used to generate high level expressions between source and target in which groups of

elements in different tables (in the case of relational data sources) or nesting of records (in hierarchal structures such as XML) are mapped together. Such high level expressions are called schema mappings. In Figure 4-1, mp_1 and mp_2 are two examples of such schema mappings for the source and the target schema. Clio (Miller et al., 2000; Popa et al., 2002), HePToX (Bonifati et al., 2005) and Spicy (Bonifati et al., 2008) are examples of data integration systems that employ property correspondences to generate schema mappings.

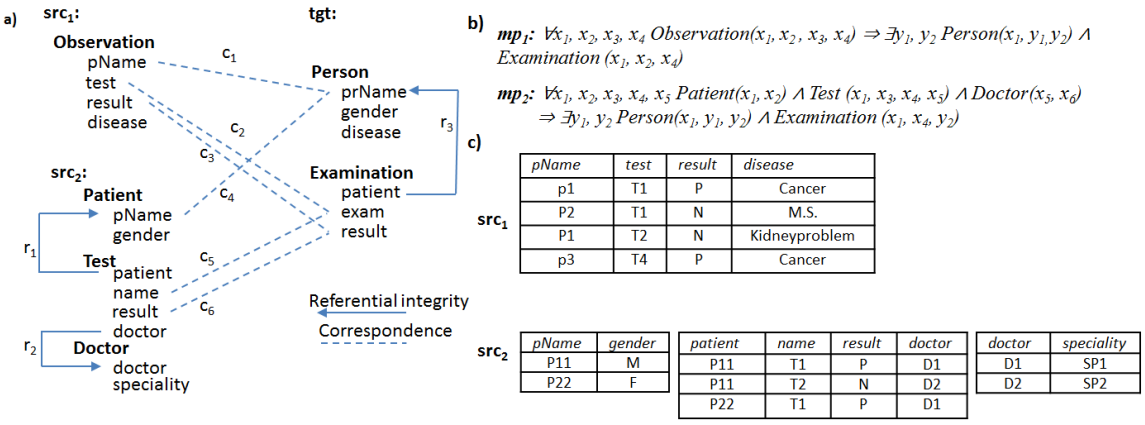


Figure 4-1: Two source schemas and a target schema (a), schema mapping expressions representing relations between sources and the target (b), and instances of src_1 and src_2

In this thesis, we consider data integration between relational schemas. We follow the formalized notions proposed by (Fagin et al., 2005a; Fagin et al., 2005b). Formally, a relational schema is a finite collection $R=\{R_1,..,R_k\}$ of relations (tables), where each relation has a fixed arity indicating the number of properties. Source and target schemas are two disjoint schemas shown as $S=\{S_1,..,S_n\}$ and $T=\{T_1,..,T_m\}$, respectively. An instance I of a schema S is a set of instances over relations of S where an instance of a

relation R_i:$\{A_1,...,A_k\}$ is a finite set of tuples in form of $R(A_1:v_1,...,A_k:v_k)$ whose arities match those of the relation symbols in R_i. We define an atomic formula as an expression of the form $R(\boldsymbol{x})$, where R is a relation with variable $\boldsymbol{x}$ which is a vector of variables $x_1,...,x_t$. For example, {*Observation*} in src_1 and {*Person, Examination*} in src_2 are examples of atomic formulas shown in Figure 4-1(a).

The basic algorithm for mapping generation is explored in (Fagin et al., 2009; Popa et al., 2002) and refined later in (Fuxman et al., 2006a) to support nested mappings. Such mappings are generated by identifying *path*s and *logical associations* in source and target schemas that are elaborated in the following. Definitions adapted from (Fagin et al., 2009) are indicated with an *.

Definition 1*(*primary and relative path*): A *primary path* regarding a schema root is a sequence of $R_1(\boldsymbol{x}_1),...,R_n(\boldsymbol{x}_n)$ where $\boldsymbol{x_i}= (x_i^1, x_i^2, ..., x_i^t)$ is a vector of variables representing a tuple of relation R_i. In this formula, R_1 is a relation depending only on the *root* (there is no variable in $\boldsymbol{x}_1$ referencing the primary key of another relation), and $R_i(i\geq 2)$ is only dependent on R_{i-1} (there is a variable in $\boldsymbol{x}_i$ which is the primary key of $\boldsymbol{x}_{i-1}$). On the other hand, in a *relative path*, R_1 is a relation depending on other relations rather than the root.

In Figure 4-1(a), primary paths at the first level are PT_1: $\forall x,y,z,t$ *Observation(x, y, z, t)* and PT_2: $\forall x,y,z,t,u$ *Person(x, y, z), Examination(x, t, u).* On the other hand, PT_3: $\forall x,y,z$ *Examination(x, y, z)* is a relative path because *x* in *Examination* is a variable depending on *Person*. Primary and relative paths are extracted in the mapping generation

process, as they represent relations between data items within a schema in terms of integrity constraints.

Definition 2*: A *referential integrity constraint* is an expression of the form: $\forall PT_i \exists PT_j, E$ where PT_i is a primary path, PT_j is either a primary or relative path. E is a conjunction of equalities of the form $e_1=e_2$, in which e_1 is an expression depending on one of the variables of PT_i and e_2 is an expression depending on one of the variables of PT_j.

An example of a referential integrity constraint is r_1 in Figure 4-1(a) which is defined as $(\forall x, y, z\ Patient(x, y), \exists x', y', z', t'\ Test(x', y', z', t'), x=x'$. Primary paths in association with referential integrity constraints specify the semantic relations between schema elements within a schema. In the source schema src_2, every record in *Patient* is related to one or more records in *Test* through referential integrity constraint r_1. Accordingly, every record of *Test* is also related to one or more records in *Doctor* considering r_2. This can result in the following new association:

$\forall x_1,...,x_6\ Patient(x_1,x_2), Test(x_1,x_3,x_4, x_5), Doctor(x_5, x_6)$

Such a maximal association that cannot be expanded anymore considering other schema elements is called a *logical association* (Fagin et al., 2009; Popa et al., 2002). Logical associations are generated using the *Chase* algorithm (Popa & Tannen, 1999) that is originally used to explore functional dependencies. This algorithm involves a set of chase steps in which a chase step is applied on a Path PT_k using the referential integrity constraint $(\forall PT_1 \exists PT_2, E)$. In each step, when PT_1 is subsumed in PT_k but PT_2 is not subsumed in PT_k, a new path including PT_2 and condition E is added to existing associations. The *Chase* algorithm continues until no chase step can be executed

regarding existing referential integrity constraints. To form a mapping expression, logical associations covering property correspondences between the source and the target are considered together. The details of this process are discussed in (Fagin et al., 2009a).

Definition 3* (*property correspondence*): A correspondence from a property p_1 in path PT_1 to property p_2 in path PT_2 is a triple $\langle PT_1, PT_2, p_1=p_2 \rangle$.

Typically, correspondences are generated automatically using schema matching techniques (which is a different research issue not addressed in this thesis). In Figure 4-1(a), $c_1,...,c_6$ are correspondences shown in dashed lines.

To find the maximal set of correspondences that can be interpreted together regarding relations between classes in the source and the target, it is necessary to indicate how a property correspondence is related to a pair of logical associations in the source and the target. Examples of two mapping expressions between source schemas and the target schema are shown in Figure 4-1(b). These mappings are created from property correspondences and referential integrity constraints in Figure 4-1(a). The semantics of these mappings states a containment assertion between a query over the source, Q_s, and a Query over the target, Q_t, where, for each tuple returned by Q_s, there must exist a corresponding tuple in Q_t. For example, mapping mp_1 states that for each *Observation* tuple in src_1, there must exist tuples in *Person* and *Examination* in the target.

The second step in data integration is using schema mappings for query rewriting to integrate data. A data integration system can be either LAV (local-as-view), in which a mapping statement relates an element of the source schema to a query (view) over the global schema, or GAV (global-as-view), in which a mapping statement relates each

element of the global schema to a query (view) over local schemas. GLAV (global-and-local-as-view) is the generalization of LAV and GAV, in which a query over the source schema is related to a query over the target schema (Halevy et al., 2006). As discussed in (Yu & Popa, 2004), GLAV mappings provide the language of sound but not necessarily complete assertions when restricted to relational schemas. In data integration through virtualization, a query posed to a target schema is rewritten based on mapping expressions such that the union of the evaluation of the rewritten queries returns the same result as evaluating the target query on the target. In (Yu & Popa, 2004), a query q is defined as an expression of the form:

q:= **in** *P* **where** *E* **return** *T*

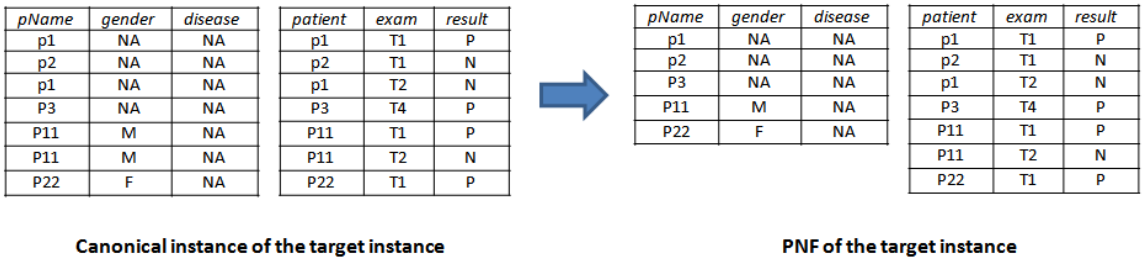
This formula indicates that given a path *P*, which is a sequence of relations $R_1(x_1),..., R_n(x_n)$, and a set of equality dependencies *E*, return a set of tuples corresponding to a sequence of properties $t_1,..,t_n$ in *T*, in which the conjunction of conditions implicitly indicated by path *P* is true. In this formula, both target and source elements can be used to form a query, and during query rewriting, a query is answered using source and target instances. For example, regarding the target schema in Figure 4-1(a), an example of a query is:

q_1:= **in** $\forall x_1,...,x_6$ *Person*(x_1, x_2, x_3), *Examination*(x_4, x_5, x_6)
where $x_1=x_4$
return *Person.prName, Examination.result*

This query, which is formed using target data elements, asks for name of patients and their examination results. It is also possible to state tuples of *Persons* and *Examinations* implicitly in a path (e.g., $\forall x_1,...,x_5$ *Person*(x_1, x_2, x_3), *Examination*$(x_1, x_4,$

x_5)), in which a variable of a *Person* is corresponded to a sequence of variables in *Examination*. Given source and target data sources, we are interested to answer the query q_1 using mappings between source and target schemas. Generally, there are two approaches to answer a target query. One involves finding target instance *J* that is consistent with mapping expressions and source instance *I*, and then performing the target query on this target instance. In this approach, which is called *data exchange*, the target instance *J* is materialized before query answering. In the second (or *data integration*) approach, a target query is rewritten into some source queries (applicable on the source) without need for materialization in the target. In the data exchange approach, in order to avoid data redundancies in the target instance a technique to generate a canonical instance has been proposed (Yu & Popa, 2004). Briefly, for each combination of tuples in the sources matching the left side of a mapping expression between source and target, tuples satisfying the right hand side of that mapping are added to the target. The atomic values that are added to the target instance are equal to the source values if correspondences between the source and the target exist. Otherwise, "unknown" (shown by NA) values are created for target properties. In our example, the canonical instance regarding mapping mp_1 and mp_2 and source instances in Figure 4-1(c) is shown in Figure 4-2(a). To remove some redundancies of canonical answers, a procedure is used to create an instance called Partitioned Normal Form (PNF) (Yu & Popa, 2004). Finally, as a post-processing step, target constraints are applied on the target solution to find a logically correct instance in the target. Gottlob & Nash (2008) propose an algorithm to compute the core solution in

polynomial time using target constraints. Finally, the target query is evaluated on this materialized target instance to find the result.



pName	gender	disease
p1	NA	NA
p2	NA	NA
p1	NA	NA
P3	NA	NA
P11	M	NA
P11	M	NA
P22	F	NA

patient	exam	result
p1	T1	P
p2	T1	N
p1	T2	N
P3	T4	P
P11	T1	P
P11	T2	N
P22	T1	P

Canonical instance of the target instance

pName	gender	disease
p1	NA	NA
p2	NA	NA
P3	NA	NA
P11	M	NA
P22	F	NA

patient	exam	result
p1	T1	P
p2	T1	N
p1	T2	N
P3	T4	P
P11	T1	P
P11	T2	N
P22	T1	P

PNF of the target instance

Figure 4-2: Canonical instance and PNF instance based on mappings in Figure 4-1(b) and source instances in Figure 4-1(c)

On the other hand, in the data integration approach, in which a target query is rewritten into some source queries, there is no need for materializing the target solution. More specifically, in this approach query answering is performed through query rewriting. In our example, query q_1 is rewritten using mapping mp_1 and mp_2 where the rewritten queries are:

r_1: **in** $\forall x_1,...,x_4$ *Observation*(x_1, x_2, x_3, x_4)
return *Observation.pName, Observation.result*

r_2: **in** $\forall x_1,...,x_6$ *Patient*(x_1, x_2), *Test*(x_3, x_4, x_5, x_6)
where $x_1 = x_4$
return *Patient.pName, Test.result*

Queries r_1 and r_2 are rewritten forms of q_1 based on mappings mp_1 and mp_2 .The union of evaluating r_1 and r_2 on the sources src_1 and src_2, respectively returns the same results as evaluating q_1 on the materialized target instance. Differences between data exchange and data integration approaches are already elaborated in many research papers

(Fagin et al., 2009; Hernández et al., 2008). In this thesis, we focus on semantic heterogeneity reconciliation in the data integration approach.

4.3 Theoretical Foundations for Semantic Heterogeneity Reconciliation

We argue that property correspondences representing equivalence relations between similar properties in heterogeneous data sources are not enough for complete semantic heterogeneity reconciliation in data integration. To address this, we employ *property precedence* relations as fundamental relationships between properties to reconcile semantic heterogeneities as elaborated in the following.

4.3.1 Ontological Foundations

Many problems in information integration have origins in specifications used to represent relationships between schemas. Relying solely on property correspondences stated in terms of equivalency between two properties is not enough to handle some semantic heterogeneity. In looking for more comprehensive relations between properties, we turn to ontology, the branch of philosophy that deals with the order and structure of reality in the broadest way possible. Ontological principles have been widely considered as a theoretical foundation for conceptual modeling both in theoretical analyses (Parsons & Wand, 2000; Wand et al., 1999) and in empirical studies (Gemino & Wand, 2004; Parsons, 2011). Ontology deals with describing the real world, and information systems represent models of this world. In particular, we adapt Bunge's ontology (Bunge, 1977) for this purpose. The rationale for this choice is that this ontology has already been applied to different types of systems. In addition, this ontology is well-formalized in

terms of set theory and extensively references other philosophical ontologies (Wand & Weber, 1990).

To reconcile semantic heterogeneity in data integration, we focus on properties and relations between properties in Bunge's Ontology (Bunge, 1977), as properties are the basic constructs in information systems. To this end, we study the *property precedence* notion of Bunge's ontology, and employ this fundamental type of relation between properties in data integration. Property p_1 precedes property p_2, if and only if, the set of things possessing p_2 is a subset of the set of things possessing p_1. For example, the property of "being female" precedes the property of "being mother" since the set of entities that are mother is a subset of entities who are female. In (Sekhavat & Parsons, 2012), we have shown the potential of using property precedence relations in semantic heterogeneity reconciliation.

4.3.2 Formalism

In Bunge's ontology (Bunge, 1977), a domain of interest is a set of things in which each thing possesses at least one property. In (Parsons & Wand, 2008) a domain of interest is defined as a set of instances (*I*) where each instance can be a material object, action, event, or any other phenomenon. In this definition, "property" is a statement about the characteristics of an instance. When a property depends only on one thing, it is called an intrinsic property representing an inherent characteristic (e.g., the height of a *person*). On the other hand, when a property depends on more than one thing, it is called a mutual property (e.g., *surgeryDate* that represents the date on which a *patient* is operated on by a *surgeon*). In our formalism, predicate *p(i)* indicates that the instance *i* possesses the

intrinsic property *p*. Accordingly, predicate $s(i_1, i_2)$ indicates that instances i_1 and i_2 jointly possess the mutual property *s*.

Definition 4: Let *D* denote the domain of values of property *p*. A *manifestation* of *p* is a specific value $v \in D$ of *p* assigned to *p* denoted: $m := \langle p,v \rangle$.

If an instance possesses a manifestation $m := \langle p,v \rangle$, it also possesses *p*. For example, possessing the manifestation $m := \langle color, red \rangle$ by an instance indicates also that the instance has *color*. In the case of intrinsic properties, the predicate $m(i)$ shows whether or not the instance *i* possesses the manifestation *m*. In the case of possessing a mutual property, $m(i_1, i_2)$ is the predicate indicating if the manifestation *m* is jointly possessed by instances i_1 and i_2.

Definition 5: The *scope* of a manifestation *m* is the set of instances possessing *m*.

According to this definition, if *m* is the manifestation of an intrinsic property, $scope(m)$ is the set of instances possessing this manifestation, and if *m* is the manifestation of a mutual property, $scope(m)$ is a set of pairs $\langle i_1, i_2 \rangle$ such that *m* is jointly possessed by i_1 and i_2. Using this concept, we define property precedence relations.

Definition 6: *Simple Property Precedence (SPP)*: Let m_1 and m_2 the manifestations of properties p_1 and p_2, respectively. m_1 is said to precede m_2, if and only if, $scope(m_2) \subseteq scope(m_1)$ which is denoted: $m_2 \rightarrow m_1$.

According to this definition, the manifestation of a property p_1 precedes the manifestation of property p_2, if and only if, every instance *i* possessing m_2, also possesses m_1. In this definition, m_2 is the preceded (inferring) property, and m_1 is the preceding (inferred) property. We categorize simple property precedence relations in three groups as

follows: Generalization (spp_1), which is a special case of simple property precedence in which a general property is inferred from a specific properly. For example, m_1:= *‹diseaseType, "leukemia"›*→m_2:= *‹diseaseCategory, "cancer"›* shows that "*cancer*" is a more general term than "*leukemia*" since it includes many other cancer types (e.g., lung cancer, thyroid cancer). The second group (spp_2) is a simple property precedence in which a specific manifestation of an intrinsic property precedes a subset of possible values of another property. For example, *‹sugarLevel, "[140, 200]"›* → *‹sugarDegree, "high"›* states that a particular manifestation *‹sugarDegree, "high"›* precedes a range of values of *sugarLevel*. Finally, in the third group (spp_3), property precedences are stated in terms of inference where the preceding property is inferred from the preceded property. For example, a specific disease (e.g., kidney failure) of a patient can be inferred from the type of a treatment (e.g., dialysis).

It is also possible that two properties simultaneously precede each other, which is termed co-precedence. Manifestations of two different properties co-precede each other if everything that possesses one manifestation also possesses the other manifestation. Similar to the "property correspondences" concept used in Clio (Miller et al., 2000; Popa et al., 2002), a co-precedence indicates two equivalent properties in different data sources. For example, *gender* and *sex* are two different representations of the same characteristic that simultaneously precede each other. However, co-precedence is a more general concept than property correspondence because co-precedence may also exist between two different properties. For example, *‹diseaseName, "kidneyFailure"›* co-precedes m_2:=

‹treatment, "dialysis"› that sates patients suffering from kidney failure receive dialysis treatment, and only patients with this problem receive such treatment.

It is also possible to infer a property of an instance from properties of another instance if these two instances jointly possess a mutual property. Unlike simple property precedence, which confines the existence of a precedence relation among properties of a particular instance, we propose it is possible to state property precedences among different instances. We term this type of precedence a "compound precedence".

Definition 7: *Compound Property Precedence* (CPP): Let i_1 and i_2 denote two instances in a schema. A compound property precedence exists between two manifestations m_1 and m_2 if possessing m_2 by i_2 can be inferred from possessing m_1 by i_1, and i_1 and i_2 mutually possess s. The compound precedence is denoted:

$$(m_1,i_1) \xrightarrow{s} (m_2,i_2).$$

Compound property precedence may exist between the same or different properties. In compound property precedence between the same intrinsic properties (cpp_1), which is denoted $[m_1{:=}\ \langle p_1, v_i\rangle, m_1(x_1)] \xrightarrow{s(x_1,x_2)} [m_1{:=}\ \langle p_1, v_i\rangle, m_1(x_2)]$, any specific manifestation m_1 of an intrinsic property p_1 of an instance x_2 precedes the same manifestation of the same property of another instance x_1, if x_1 and x_2 jointly possess the mutual property s. For example, the compound property precedence $[m_1{:=}\ \langle address,$ *"add"*$\rangle, m_1(x_1)] \xrightarrow{couple(x_1,x_2)} [m_1{:=}\ \langle address,$ *"add"*$\rangle, m_1(x_2)]$ implies that for any two persons x_1 and x_2 that are known as a *couple*, by knowing the address of x_1, we can infer the address of x_2. On the other hand, in compound property precedence between different

intrinsic properties (cpp_2), which is denoted $[m_1 := \langle p_1, v_1 \rangle, m_1(x_1)] \xrightarrow{s(x_1,x_2)} [m_2 := \langle p_2, v_2 \rangle, m_2(x_2)]$, a specific manifestation m_2 of an intrinsic property p_2 of an instance x_2 precedes the manifestation m_1 of a different property p_1 of another instance x_1, where x_1 and x_2 jointly possess the mutual property s. For example, suppose only heart specialists who have *open heart surgery* certification can conduct *heart valve leakage* surgery. The compound property precedence for this example can be shown as $[m_1 := \langle disease, heartValveLeakage \rangle, m_1(x_1)] \xrightarrow{surgery} (m_2 := \langle specialty, openHeartSurgery \rangle, m_2(x_2)]$. This compound property precedence relation states that the specialty of a doctor can be inferred from disease of a patient who has been under surgery by that doctor. In this example, surgery is a mutual property jointly possessed by a doctor and a patient.

4.4 Toward Semantically Enhanced and Configurable Data Integration

There are a number of unresolved issues regarding semantic heterogeneity reconciliation in data integration including: how to integrate the same concepts and properties represented by different manifestations; how to integrate similar concepts represented in different levels of abstraction; and how to use implicit properties inferred from existing explicit properties to bind two data sources? In the following, we discuss how these issues are addressed to enhance data integration. We argue that relying solely on mapping expressions generated from property correspondences (i.e., equivalence relations between data items in source and target) is not enough to handle all semantic heterogeneities in query rewriting.

The concept of semantic relations between data items in different data sources is ignored in many data integration techniques. Two different structures in different data sources may represent the same concept. However, because the structures are different, they do not appear in property correspondences. As a result, semantic relations between them are ignored during schema mapping generation. We employ property precedence relations to reconcile such semantic heterogeneities. Property precedence relations not only cover property correspondences and subsumption relations, but also make it possible to infer implicit properties and discover semantically richer relations between data sources. We show how data integration can be enhanced using semantic relations between properties stated in terms of property precedence. We assume that property precedence relations are already created automatically or manually by users (issues in developing a property precedence schema are outside the scope of this thesis – see (Parsons & Chen, 2008) for a discussion on creating precedence schemas).

We first use local property precedence relations in the source and the target to find implicit properties of instances that can be used to bind two different data sources. In particular, we propose two algorithms that employ simple and compound property precedence relations for this purpose. When there is no explicit and direct relation between two properties, exploring implicit properties may make it possible to bind two data sources to integrate data. Mapping generation based on property precedence relations is a non-trivial task that requires systematic extension of the schema mapping algorithms. The novelty of this technique is in considering implicit relations between schema elements as well as explicit property correspondences. In this technique, mappings can be

enhanced incrementally using auxiliary information in terms of precedence relations between properties (discussed in Sections 4.4.1 and 4.4.2).

In the second phase, we use global property precedence relations to enhance data integration techniques through query rewriting. Global precedence is used to rewrite target queries considering new implicit semantics. The technique we propose for query rewriting not only considers equivalency between properties through property correspondences, but also takes into account similar properties represented at different levels of abstraction. Binding properties of source and target schemas through this type of relation provide some flexibility in data integration in terms of accuracy and completeness discussed in Section 4.4.3.

4.4.1 Mappings Enhancement Using Local Property Precedence

In this section, we discuss how schema mappings can be enhanced semantically using property precedence relations. Mappings generated in our technique are manifestation-based associations in which mappings are accompanied with conditions in terms of possessing some specific values. Unlike the concept of logical association (representing semantic relations between data items within a schema) discussed in Section 4.2, we generate a new type of logical association called manifestation-based association by applying simple and compound property precedence relations on existing logical associations already created by the Chase algorithm.

Definition 8: A *manifestation-based association* is a pair *ma:‹AS, M›,* where *AS* is a logical association and *M* is a set of manifestations.

Unlike logical associations, in which equalities are stated between class level properties (e.g., *Observation.pName = Examination.patient*), in a manifestation-based association, equalities are the specific values assigned to properties (e.g., specialty = 'surgeon'). In a manifestation-based association, associations between data elements exist only if such specific values are assigned to properties according to *M*. Note that a manifestation-based association is a fine-grained association in which data and metadata are taken into account simultaneously to specify logical relations within a data source. The difference between manifestation-based associations and logical associations is that implicit properties of instances are taken into account in the former.

The main idea behind using constraints in terms of property precedence relations is to extract implicit properties of instances inferred from existing explicit properties for use in binding two data sources. In Figure 4-3(a), sp_1 and sp_2 are simple property precedences stating that the disease of a patient precedes the treatment for specific manifestations of that disease. In this example, *kidney failure* can be inferred from *dialysis*, and *cancer* can be inferred from *chemotropic*.

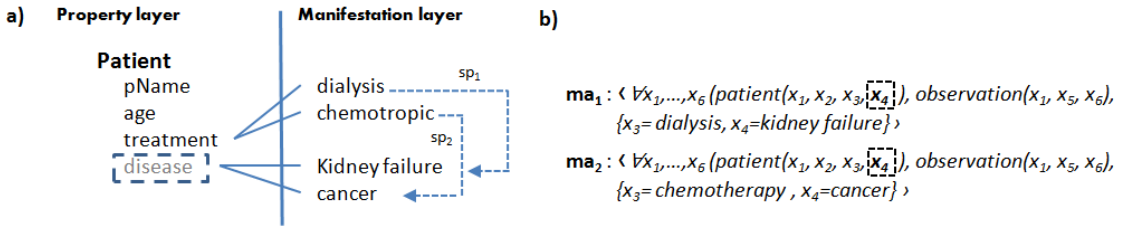


Figure 4-3: An example of applying simple property precedence

Algorithm I is proposed to enhance logical associations using simple property precedence relations. This algorithm uses a set of simple property precedence relations $SP = \{sp_1, sp_2, \ldots, sp_n\}$, in which $sp_k = m_i \rightarrow m_j$, $i \neq j$. When there exists $m_i \rightarrow m_j$ such that the

property of m_i is equal to one of the properties of relations in the schema, a new association $\langle a_i, M_T \rangle$ is generated. Accordingly, given $m_{i+1} \rightarrow m_{j+1}$, in which $m_{i+1} = m_j$, a new association $\langle a_{i+1}, M_{T+1} \rangle$ is created with a new manifestation set such that $M_{T+1} = M_T \cup m_{j+1}$. The Algorithm terminates when there is no simple property precedence sp_k that can be expanded using existing manifestations. Intuitively, for an acyclic and finite set of property precedence relations, Algorithm I finally terminates because each sp_k in *SP* generates at most one new association. The result of applying this algorithm on logical association A_1: $\forall x_1,...,x_5$ *patient*(x_1,x_2,x_3), *treatment*(x_1,x_4,x_5) and simple property precedences (sp_1,sp_2) is shown in Figure 4-3(b). Using this algorithm, property *disease* is added to the relation *patient.* In this example, *disease* did not exist originally in the *patient* relation, and it is added after applying property sp_1 and sp_2.

Algorithm I (SPP enhancer): Enhancing logical associations using SPP relations

---

Input: Set of logical associations AS
Set of simple property precedence relations SP
Output: Set of manifestation-based associations MA
GenerateSPPEnabledAssociations
1: MA←copy items in AS with null manifestation
2: **do {**
3: endFlag ←true
4: **foreach** association a_i **in** MA
5: create a new manifestation-based association $ma_i = \langle a_i, M_t = null \rangle$
6: R_i← Extract the relations of a_i
7: **foreach** relation r_i **in** R_i
8: create a new relation $r_t = r_i$
9: **foreach** property p_i **in** r_i
10: **If** there exist a simple property precedence
11: $m_1 := \langle p_1, v_1 \rangle \rightarrow m_2 := \langle p_2, v_2 \rangle$ in SP and $p_i = p_1$ **then**
12: add p_2 to relation r_t
13: add $\{m_1, m_2\}$ to M_t
14: endFlag ←false
15: **if** ($r_t \neq r_i$) **then** replace r_i with r_t in a_i
16: add ma_i to MA
17: **}While (**endFlag = false)

18: **return** MA

Constraints in terms of local compound property precedence relations can also be used to explore different manifestations of the same concepts. A compound property precedence $(m_1,i_1) \xrightarrow{s} (m_2,i_2)$ implies that, for each pair of instances i_1, i_2 that mutually possess property s, the predicate $m_2(i_2)$ can be inferred from $m_1(i_1)$. Possessing a mutual property can be represented using referential integrity constraints in a relational schema. Figure 4-4(a) shows how implicit properties can be inferred from compound property precedence relations. In this Figure, cp_1 and cp_2 are compound property precedences stating that the specific specialty of a doctor precedes some specific types of procedures performed by that doctor. For example, assuming that *lasik* surgeries can be performed only by eye specialists, it is possible to infer a specialty of doctor from the type of treatment that has performed by that doctor.

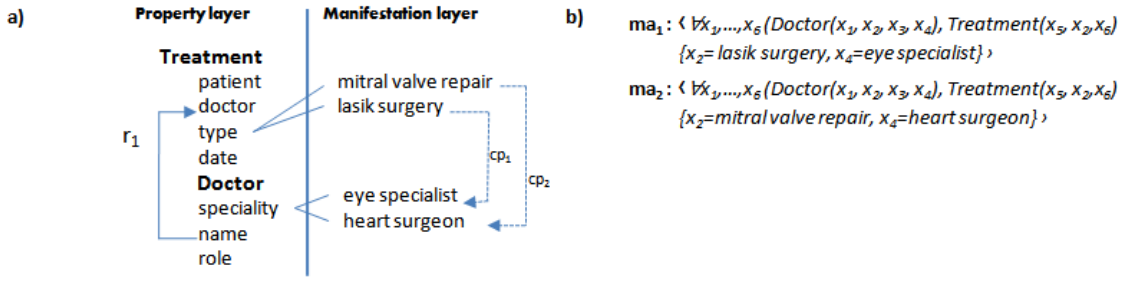


Figure 4-4: An example of applying compound property precedence

The details of activities to infer such implicit properties using compound property precedence relations are shown in Algorithm II. In this algorithm, the set of relations in an input association is checked to find pairs of relations (r_i, r_j) such that r_j is dependent on r_i, and there exists a compound property precedence $m_1{:=}\langle p_1,v_1\rangle \rightarrow m_2{:=}\langle p_2,v_2\rangle$ matching

these relations. When such pair is found, a new manifestation-based association is created in which specific properties (manifestations) of cp_i are included. The result of applying this algorithm on logical association A_i: $\forall x_1,...,x_6\ doctor(x_1, x_2, x_3, x_4), treatment(x_5, x_2, x_6)$ and compound property precedences cp_1 and cp_2 is shown in Figure 4-4(b).

Manifestation-based associations take into account specific values of properties as well as class level properties. These manifestation-based associations are used to find new mappings called manifestation-based mappings that are elaborated in the next section.

Algorithm II (CPP Enhancer): Enhancing associations using CPP

Input: Set of associations AS
Set of compound property precedences CP
Output: Set of manifestation-based associations MA
GenerateCPPEnalbledAssociations
1: MA←*null*
2: **foreach** association a_i in AS
3: create a new manifestation-based association $ma_i = \langle a_i, M_T = null \rangle$
4: $RS_i \leftarrow$ Extract the sequence of relations of a_i
5: **foreach** two sequential relations (r_i, r_j) in RS_i
6: **If** there exist a precedence $m_1 := \langle p_1, v_1 \rangle \rightarrow m_2 := \langle p_2, v_2 \rangle$ in CP
7: such that p_1 is in r_i and p_2 is in r_j **then**
8: add $\{m_1, m_2\}$ to M_T
9: **If** ($M_T \neq$ null)**then**
10: add ma_i to MA
11: **return** MA

4.4.2 Generating Semantically Enhanced Mappings

Logical associations enhanced using simple and compound property precedences are used in conjunction with a set of property correspondences to generate schema mappings. A property correspondence shows an equivalence relation between two properties in different data sources. We use manifestation-based associations to combine such correspondences in a meaningful way. The main idea behind the algorithm for mapping

generation (Algorithm III) is that correspondences from source to target schema are considered together to generate a complex mapping expression when the following conditions hold: (1) all elements of the sources in those correspondences all occur in an association of the source schema, and (2) all elements of the target in those correspondences all occur in the same association of the target schema. As a result, Algorithm III finds the maximal set of correspondences that can be interpreted together by ensuring the elements they match belong to the same logical association in the source and in the target schema. Note that the focus of this thesis is on semantic heterogeneities rather than structural heterogeneities. As discussed in (Fagin et al., 2009), conversion functions like $p_1=f(p_2)$ can be used instead of $p_1=p_2$ to reconcile such structural heterogeneities (e.g., concat(*creditCardNo*, *expDate*) = *creditCardInfo*).

To find the maximal set of correspondences that are related together, it is necessary to indicate how a property correspondence is related to a pair of manifestation-based associations in the source and the target. To that end, we take into account the concept of coverage among a manifestation-based association and a property correspondence. A pair of manifestation-based association $\langle MA^S, MA^T \rangle$ *covers* the property correspondence $c:\langle PT_1, PT_2, p_1=p_2 \rangle$ if MA^S covers PT_1 and MA^T covers PT_2. A manifestation-based association is a pair $\langle AS, M \rangle$ where *AS* is a logical association and *M* is a set of manifestations. As a result, when MA^S and MA^T cover PT_1 and PT_2, respectively, the sequence of relations in PT_1, and PT_2 is the subset of sequence of relations in the logical association related to MA^S and MA^T.

In Clio (Fagin et al., 2009; Miller et al., 2000; Popa et al., 2002), a mapping is defined as a triple $\langle A^S, A^T, E\rangle$ in which A^S and A^T are logical associations in the source and the target where E is a conjunction of equalities representing associations between class level properties of A^S and A^T. In this thesis, we introduce and generate manifestation-based mappings in which mappings are accompanied with specific properties (manifestations).

Definition 9 (*Manifestation-based mapping*): A manifestation-based mapping is a statement in form of [*mp*: MA^S ***in*** *S*, MA^T ***in*** *T* ***with*** *M*], where MA^S and MA^T are manifestation-based associations in source (S) and target (T) schemas, respectively. *M* is a set of manifestations representing some constraints in terms of possessing specific values by some properties in source and target schemas. In this definition, equalities regarding the referential constraints implicitly exist in MA^S and MA^T.

Algorithm III (Mapping generator)

Input: Manifestation-based associations of the source MA^S
Manifestation-based associations of the target MA^T
Correspondences between source and target CR
Output: A set of Mappings MP

```
A← manifestation-based associations in (MA^S) that cover at least
    one path in the source side of correspondences in CR
B← manifestation-based associations in (MA^T) that covers at least
    one path in the target side of correspondences in CR
foreach pair <a, b> in A×B
    C←{c| c∈CR and c is covered by<a, b>}
    If C = null then continue;
    Let C= {c1,...,cm}
    m^S← {manifestations of a}
    m^T← {manifestations of b}
    foreach ci in C
        let e the equality in ci
        If (m^T =null or m^T⊆m^S)
            update variables of a and b according to e
            mp: a in S, b in T with m^S
```

16: MP←MP∪{mp}
17: **Return** MP

The manifestation-based mapping [MA^S ***in*** *S*, MA^T ***in*** *T* ***with*** *M*] states that, for a path in MA^S regarding manifestations in *M*, a path exists in MA^T. Algorithm III shows the details of generating manifestation-based mappings from manifestation-based associations and a set of property correspondences. Unlike Clio (Fagin et al., 2009; Miller et al., 2000; Popa et al., 2002), which checks each pair of logical associations in the source and the target to find mutual correspondences, our proposed approach overcomes the problem of handling a large number of associations by pruning the set of associations and considering only those items covering at least one path of correspondences. Using this approach, the number of combinations of source and target associations checked to find a maximal set of correspondences is significantly reduced.

We use an example to illustrate how Algorithm III works. Suppose source schema S consists of relations *patient* and *observation*, and the target schema T includes relations *person* and *examination* as follows:

S: *patient(pName, disease), observation(patient, doctorName, type)*
T: *person(prName, gender, role), examination(pName, result)*

Suppose correspondences c_1 and c_2 show property correspondences between S and T as follows:

c_1: ‹ ∀*x,y,z patient(x, y),* ∀*x′,y′,z′ person(x′, y′, z′), x=x′*›
c_2: ‹ ∀*x,y,z observation(x, y, z),* ∀*x′,y′ examination (x′, y′), x=x′*›

Using Algorithms I and II, the following manifestation-based associations are generated that are inputs for Algorithm III. In this example, only correspondence c_1 can be applied on logical associations as the path in ma^T does not cover the path of c_2.

ma^S: ‹ ∀x,y,z *patient(x, y, z), {y='ovarian cancer', z= 'female'}*›
ma^T: ‹ ∀x',y',z' *person(x', y', z'), null*›

Finally, using these manifestation-based associations and property correspondences, the following mapping is generated:

mp_1: ∀x,y,z *patient(x, y, z)* ***in*** *S,* ∃x' *person(x, y ,x')* ***in*** *T*
with *{y='ovarian cancer', z= 'female'}*

However, for this example, ++Spicy (Marnette et al., 2011) (an open source implementation of Clio's algorithms) generates mapping mp'_1 in which the correspondence between the implicit property of *patient* (i.e., *sex*) and property *gender* in *person* relation is neglected:

mp'_1: ∀x,y *patient(x, y)* ***in*** *S,* ∃x', y' *person(x, x', y')* ***in*** *T*

As another example, considering manifestation-based associations ma^S and ma^T, compound property precedences c_1 and c_2 can be applied in Algorithm III such that:

ma^S: ‹(∀x,y,z,t,r *patient(x, y, z), observation (x, t, r)), {y='ovarianCancer', z= 'female'}*›
ma^T: ‹(∀x',y',z', t' *person(x', y', z'), examination (x', t')), null*›

Using such manifestation-based associations and property correspondences between *patient.pName* and *person.prName* and also between properties of *observation* and *examination*, the following manifestation-based mapping is created:

mp_2: ∀x,y,z,t,r *patient(x, y, z), observation (x, t, r)* ***in*** *S,*
∃x',y' *person(x, z, x'), examination (x, y')* ***in*** *T*

with *{y='ovarian cancer', z= 'female'}*

On the other hand, since ++Spicy (Marnette et al., 2011) does not consider the implicit property (*sex*) in the source and its correspondence to *gender* in the target, the following mapping is created in which source and target schemas are not bound regarding *gender* and *sex*:

mp'_2: $\forall x,y,z,t,r$ *patient(x, y), observation (x, z, t)* ***in*** *S,*

$\exists x', y', z'$*person(x, x', y'), examination (x, z')* ***in*** *T*

Using property precedence relations has two important implications that differentiate our approach from ++Spicy (Marnette et al., 2011). First, property correspondences are specified among class level properties in ++Spicy; consequently, mappings are stated in terms of class level properties. However, such class level properties are not expressive enough to capture the full semantics of concepts and relations. On the other hand, in the technique proposed in this thesis, mappings are expressed using data level properties that allow specifying some new mappings between explicit properties as well as implicit properties inferred from existing explicit properties. Second, unlike general mappings generated by ++Spicy (Marnette et al., 2011), the manifestation-based mappings generated in our approach are accompanied with some constraints that specify under what conditions a mapping expressions is applicable. Such conditions make it possible to define fine-grained mappings between source and target schemas.

4.4.3 Query Rewriting Using Manifestation-based Mappings

In the schema mapping generation phase, we used local property precedence relations in the source and the target to find implicit properties used to bind two data sources. In the query processing phase, we use global property precedence relations, including precedences between source and target items. This global precedence schema relates similar concepts represented at different levels of abstraction. Unlike matching equivalent properties (i.e., through property correspondences), matching properties at different levels of abstraction provides opportunities to obtain new potential answers to a query.

In this section, we propose a new approach for data integration through query answering in which different query rewriting algorithms can be performed depending on users' preference for complete or sound answers. In this approach, which is called *Configurable Data Integration (CDI)*, a general term in the target query (preceding property in a simple property precedence relation) can be replaced with a more specific term in the source (preceded property in that property precedence relation) to return tuples of the source matching this specific property. This type of query rewriting, which we call query expansion through specialization (Exp-Spc), increases the completeness of the answers without losing accuracy. On the other hand, in query expansion through generalization (Exp-Gen), a specific term in the target query is replaced with a more general term in the source, that consequently results in returning some new correct tuples as well as some unwanted incorrect tuples. Therefore, query expansion results in increasing the completeness of answers at the expense of compromising accuracy. Note that although conventional query rewriting techniques (Fagin et al., 2009; Miller et al.,

2000; Popa et al., 2002) employ equivalence relations (i.e., property correspondences) to generate sound rewritings, they may return incomplete answers as they ignore semantic relations between properties that are represented at different levels of abstraction. To provide comprehensive semantic relations between source and target properties, the technique proposed in this chapter (CDI) employs global property precedence schema for query rewriting including a set of simple property precedence relations.

In query rewriting, an input query posed to the target is transformed to a set of subqueries applied on source datasets where different representations of the concepts of the target are used in these subqueries (Bonifati et al., 2011). Based on a user's preference in terms of accuracy or completeness in data integration, query rewriting can be fulfilled using different query rewriting algorithms. The details of query rewriting based on Exp-Spc and Exp-Gen are elaborated in Algorithm IV. In Exp-Gen, by relaxing query predicates, which is stating the query predicate at a higher level of abstraction (through replacing a more specific property with a generic property), some new semantically correct tuples are returned at the expense of returning some unsound answers. Such query expansion is an important characteristic of CDI that allows finding hidden potential answers in applications in which completeness is critical. In other words, in some applications (e.g., identifying malicious behaviors and intrusions) finding a complete set of answers is crucial even if the answers include some false-positive results. Query expansion through generalization (Exp-Gen) allows exploring some *potential answers* that are not extracted using direct binding (property correspondence) between source and target properties. Additional processing can remove incorrect query results.

Algorithm IV (Query Rewriter): Query rewriting through Exp-Gen and Exp-Spc

---

Input: query constraint *c*, set of simple PP relations $\sum_i = \{\sigma_1, \sigma_2, \ldots, \sigma_t\}$
Where σ_i is in form of $m_1 := \langle p_1, v_1 \rangle \rightarrow m_2 := \langle p_2, v_2 \rangle$,
source database D_i , and the target database D_j
RT: //Rewriting Type (can be Exp-Gen or Exp-Spc)
Output: result (rewritten query predicate)
Begin
result← *c*
foreach manifestation *m* in *c* **do {**
if RT= Exp-Gen **then{**
if there exist a σ_i in $\sum_I$ such that $m_1 = m$ and m_1 is known in D_j then
result← $m_2 \cup (c - \{m_1\})$**}**
else if RT= Exp-Spc **then {**
if there exist a σ_i in $\sum_I$ such that $m_2 = m$ and m_2 is known in D_j then
result← $m_1 \cup (c - \{m_2\})$**}**
}
If (result = *c*) then
return null**;**
else
return result**;**
End

CDI provides two different options in rewriting of a target query. If the purpose of query rewriting is to maximize recall at the expense of possibly lower precision (because of creating some false positive results), then Exp-Gen as well as Exp-Spc algorithms are applied. On the other hand, if the purpose is to increase recall without losing precision, only Exp-Spc is employed in query rewriting.

Given the mapping mp_1 that was generated in Section 4.4.2,

mp_1: $\forall x,y,z$ *patient(x, y, z)* **in** *S*, $\exists x'$ *person(x, y ,x')* **in** *T*
with *{y='ovarian cancer', z= 'female'}*

Suppose the target query q_2 was:

q_2:= **in** $\forall x,y,z$ *Person*(x_1, x_2, x_3), *Examination*(x_4, x_5, x_6)
where $x_1 = x_4$, x_3= *"leukemia"*

return *Person.prName, Examination.result*

This query asks for the name of patients and examination that have been performed on the patients who suffer from *leukemia.* Using mapping mp_1, and property precedence relation m_1:= ‹*diseaseType, "leukemia"*›→m_2:= ‹*disease, "cancer"*›, the query rewriting algorithm (Algorithm IV) generates the following query using Exp-Gen:

q_2:= **in** ∀*x, y, z Observation*(x_1, x_2, x_3, x_4)
where x_4 = *"Cancer"*
return *Person.prName, Examination.result*

Using this rewritten query, and given the source instance in src_1 in Figure 4-1(c), {(p1, p), (p3, p)} is returned, while no result is returned using ++Spicy (Marnette et al., 2011) because no record matches the query condition (x_3= *"leukemia"*). Note that P1 and/or P3 may represent patients with other types of cancer. However, since the names of diseases are not shown in the source, P1 and P3 are potential answers representing patients that may have leukemia. In other words, although this result certainly includes those patients who suffer from Leukemia (if there exist such patients), it may include some unsound answers. This is a different approach for query answering compared to sound rewriting techniques in which accuracy may be compromised to find potential answers. Further analysis can be performed on the potential set of answers to prune unsound answers.

4.4.4 Theoretical Analysis of the Query Rewriting Algorithm

The quality of query rewriting algorithms is analyzed based on accuracy and completeness of answers returned. The need for measuring the quality of query answering

arises from the fact that query processing systems are built based on some semantic reconciliation techniques that may not exactly reconcile semantic heterogeneities. In this section, we analyze Algorithm IV based on precision (accuracy of results) and recall (completeness of results). Given an input target query *q*, we calculate the number of True Positives *TP* (i.e., tuples that satisfy the query predicate of *q*, and the query processor returns them), True Negatives *TN* (i.e., tuples that do not satisfy the query predicate of *q*, and the query processor does not return them), False Positive *FP*, (i.e., tuples that do not satisfy the query predicate, but the query processor returns them), and False Negative *FN*, (i.e., tuples that satisfy the query predicate, but the query processor does not return them). Based on these items, precision and recall are computed as: *precision = TP/(TP+FP)* and *recall = TP/(TP+FN)*.

Let *n* be the number of tuples that satisfy target query predicate in source src_1. Suppose $m_x := \langle p_x, v_x \rangle$, is a generic manifestation that precedes a set of manifestations $m_1 := \langle p_1, v_1 \rangle$,, $m_2 := \langle p_2, v_2 \rangle$,, … , $m_k := \langle p_k, v_k \rangle$. Assume that src_1 employs a generic manifestation $m_x := \langle p_x, v_x \rangle$, and *b* is the number of instances that satisfy m_x in this database. The target data source employs specific manifestations $m_1,\ldots,m_k$, and $a_1,\ldots,a_k$ are the number of instances that satisfy $m_1,\ldots,m_k$, respectively. Suppose a query with the predicate c_x that includes m_x is posed to the target. In Exp-Spc, the query rewriter engine rewrites c_x by replacing a generic manifestation m_x with specific manifestations. As a result, $c_1,\ldots, c_k$ will be the rewritten query predicates of c_x. In this case, Algorithm IV generates $c_1=(c_x\text{-}mx) \cup m_1$, $c_2=(c_x\text{-}m_x) \cup m_2,\ldots$, $c_k=(c_x\text{-}m_x) \cup m_k$. According to these rewritten queries, $TP_{Cx}=b$, $TP_{Cx,C1}=b+a_1$, $TP_{Cx,C1,C2}=b+a_1+a_2,\ldots$,

$TP_{Cx,C1,C2,...,Ck}=b+a_1+a_2+...+a_k$. Hence, $FN_{Cx}=n-b$, $FN_{Cx,C1}=n-(b+a_1)$, $FN_{Cx,C1,C2}=n-(b+a_1+a_2)$,..., $FN_{Cx,C1,C2,...,Ck}=n-(b+a_1+a_2+...+a_k)$. *FP* is zero in all cases since no extra results are created by replacing a general concept with a more specific one. Based on these items, precision is not affected using Exp-Spc. This is due the fact that the target query includes a generic manifestation where through query expansion, a specific manifestation (which is also a generic manifestation) is replaced with that generic manifestation. On the other hand, $recall_{Cx}=b/n$, $recall_{Cx,C1}=(b+a_1)/n$,..., $recall_{Cx,C1,..,Ck}=(b+a_1+...,+a_k)/n=1$ that indicate Exp-Spc increases the completeness of the answers. The amount of increase depends on the number of manifestations represented through property precedence relations. In the best case, when all specific manifestations ($m_1,\ldots,m_k$) of a generic manifestation (m_x) are identified and used for query rewriting, the recall is 1.

In query expansion through generalization (EXP-Gen), a specific manifestation in an input query predicate is replaced with a more generic manifestation. Let $a_1,\ldots,a_k$ be the number of instances that satisfy $m_1,\ldots,m_k$ in src_1, and $b_1,\ldots,b_k$ be the number of instances that satisfy $m_1,\ldots,m_k$ in target. Suppose the target query predicate includes one of the specific manifestations m_i where $1<i<k$. As a result, the rewritten query constraint will be: $c_x=(c_i-m_i) \cup m_x$. Consequently, $TP_{Ci}=a_i$ and $TP_{Cx,Ci}=a_i+b_i$, $FN_{Ci}=b_i$, $FN_{Cx,Ci}=0$, $FP_{Ci}=0$ and $FP_{Cx,Ci}= b_1+b_2+...+b_k-b_i$. Based on these items, precision and recall are computed as: $precision_{Ci}=1$, $precision_{Cx,Ci}=(a_i+b_i)/((a_i+b_i)+(b_1+b_2+...+b_k-b_i))$, $recall_{Ci}=a_i/(a_i+b_i)$ and $recall_{Cx,Ci}=(a_i+b_i)/((a_i+b_i)+0)=1$. According to these results, by replacing a generic manifestation with a specific manifestation, the recall is increased

from $a_i/(a_i+b_i)$ to *1*. However, this may result in a reduction in precision. We already expected such a decrease in precision because query rewriting through replacing a specific manifestation with a generic manifestation results in returning some unsound answers corresponding to other specific manifestations rather than the manifestation in the input query.

4.5 Evaluation

In this section, we report the experiments have been performed to evaluate CDI in terms of the quality of schema mapping and query rewriting. For this purpose, a working prototype of CDI is implemented using Java. Generally, evaluating mapping systems is a quite challenging task as there is no standard input methodology for mapping generation, and there are many factors unique to each mapping system. (Alexe et al., 2008) propose a benchmark to evaluate schema mapping techniques based on the ability to support different types of mappings. This basic mapping suite represents a minimum set of transformation functions that should be supported by a mapping system. However, evaluating the effectiveness of a schema mapping technique solely based on the notion of successful implementation of mapping scenarios does not guarantee the effectiveness of data integration because the frequency of these mapping scenarios may vary in different schemas.

4.5.1 Experimental Setting

CDI is tested in several real integration scenarios. For this purpose, we used data integration scenarios provided by AIM lab available at (http://aimlab.cs.uoregon.edu).

Each data integration scenario includes two data sets (each dataset can appear as source or target) as well as a set of mappings between these data sources. The experiments were conducted in two phases. We first evaluated the quality of schema mappings generated by CDI. In particular, we studied if the schema mappings enhanced using simple and compound property precedence relations can improve semantic heterogeneity reconciliation in schema mapping. The mapping component of CDI is compared with ++Spicy (Marnette et al., 2011), which is an open source implementation of Clio algorithms. Then, we studied the effect using different query expansion algorithms in query rewriting. Given a pair of datasets (db_1, db_2), we formed two scenarios, db_1-db_2 and db_2-db_1, where the first database represents the source and the second database represents the target.

4.5.2 Datasets and Queries

As discussed in (Köpcke & Rahm, 2010), to evaluate a mapping system data from different domains must be considered. For this purpose, six datasets in three different domains (two datasets as source and target in each domain). These datasets include (DBLP1, DBLP2), (Amalgam1, and Amalgam2) and (UTDB, UTCS). UTCS and UTDB are two databases about Computer Science department and the database group at University of Toronto. DBLP1 and DBLP2 are relational schemas for the DBLP bibliography representing this domain using different schemas. Amalgam datasets include different schemas of the same data in computer science bibliography domain developed separately by different people in the Clio project. These datasets form a heterogeneous environment in which different representations are used for the same concepts in a

particular domain. Considering these datasets, six data integration scenarios were formed including DBLP1-DBLP2, DBLP2-DBLP1, Amalgam1-Amalgam2, Amalgam2-Amalgam1 and UTCS-UTDB and UTDB-UTCS. The correct mappings in each scenario are already provided by AIM lab.

To evaluate the effectiveness of mappings generated and used by CDI, the following queries were used. For Amalgam datasets, we designed 12 queries ($Q_{A1},\ldots,Q_{A12}$) including: four queries referencing at least one property that is used in simple property precedence schema ($Q_{A1},\ldots,Q_{A4}$), four queries referencing at least one property that is used in compound property precedence schema ($Q_{A5},\ldots,Q_{A8}$), and four queries not referencing any property in simple and compound property precedence schemas ($Q_{A9},\ldots,Q_{A12}$). Accordingly, we generated 12 queries ($Q_{U1},\ldots,Q_{U12}$) for UTCS and UTDB. Among 12 queries for each pair of datasets, six queries (Q_1, Q_2, Q_5, Q_6, Q_9, Q_{10}) were used in db_1-db_2 scenario while the rest of queries (Q_3, Q_4, Q_7, Q_8, Q_{11}, Q_{12}) were used in the reverse scenario (db_2-db_1). Consequently, each data integration scenario is tested for six queries including: two queries referring simple precedences, two queries referencing compound precedences, and two queries not referencing simple or compound precedences. In the case of DBLP, we designed 12 queries ($Q_{D1},\ldots,Q_{D12}$) based on queries provided by SP^{2}Bench (Schmidt et al., 2012) in which some items were modified to be compatible with DBLP1 and DBLP2 schemas.

4.5.3 Evaluation of Semantically Enhanced Mappings

In the first phase, we aim to indicate to what extent manifestation-based mappings generated by CDI affect the quality of schema mapping. Direct comparison of two sets of

schema mapping expressions is not possible as mappings are high level expressions that show relations between source and target schemas. However, it is possible to compare the effect of using such mappings in a data integration or a data exchange scenario. For this purpose, we compared the target instance generated by CDI with the target instance generated by ++Spicy (Marnette et al., 2011). We followed a similar approach used in ++Spicy (Marnette et al., 2011) to generate a target instance from a set of schema mapping expressions with a modification that checks manifestations (as conditions of mappings) to generate a target instance. Given the target instances generated by ++Spicy (Marnette et al., 2011) and CDI, we performed queries elaborated in Section 4.5.2 to evaluate the quality of mappings.

The quality of query answering on a target instance is measured in terms of precision and recall regarding the answers returned by each system. As discussed earlier in Section 4.5.2, each data integration scenario includes six queries including two queries referencing a property in a simple property precedence relation, two queries referencing a property in a compound property precedence relation, and two queries without using any property referenced in simple or compound property precedence relations. Two simple and two compound property precedence relations in each scenario were manually extracted. Examples of simple precedence, compound precedence, and co-precedence relations extracted for Amalgam dataset are shown in Table 4-1.

Table 4-1: Examples of different types of property precedence relations extracted from Amalgam Database (SPP: Simple Property Precedence, COPP: Co-Property Precedence, CPP: Compound Property Precedence)

SPP_1	m_1:= ‹loc, "USA, WA"› → m_2:= ‹loc, "UnitedStates"›
SPP_2	m_1:= ‹Descriptor, "queryProcessing"› → m_2:= ‹class, "database"›
SPP_3	m_1:= ‹publisher, "ACM"› → m_2:= ‹Language, "english"›.

$COPP_1$	m_1:= ‹*loc*, v_i› co-proceeds m_2:= ‹*countryofOrigin*, v_i›
$COPP_2$	m_1:= ‹*type*, *"techRep"*› co-proceeds m_2:= ‹*type*, *"technicalReport"*›
CPP_1	[m_1:= ‹*countryOfPublication*, "v_i"›, $m_1(x_1)$] $\xrightarrow{located(x_1,x_2)}$ [m_2:= ‹*location*, v_2›, $m_2(x_2)$]
CPP_2	[m_1:= ‹*Source*, *"IEEE Int. Conf. Data Eng"*›, $m_1(x_1)$] $\xrightarrow{Recorded(x_1,x_2)}$ [m_2:= ‹*classificationCategory*, *"database"*›, $m_2(x_2)$]

Since schema mapping in CDI considers some implicit properties in addition to existing explicit properties, a more complete set of mappings between source and target is provided using CDI in comparison to ++Spicy (Marnette et al., 2011). Consequently, we expect to generate a more complete target instance using such mappings. The results of experiments for six data integration scenarios elaborated in Section 4.5.2 support this claim (Figure 4-5). The results show that using semantically enhanced mappings can increase the completeness of query answering without affecting the accuracy. This is due to the fact that no false negative answers are retuned because only tuples satisfying mapping expressions are returned. In the case of referencing simple property precedence relations (Q_{X1}, Q_{X2}, Q_{X3}, Q_{X4} where x can be D, A, and U representing DBLP, Amalagam and UTCS, respectively), CDI increases recall because some new tuples satisfying new query predicates are returned. However, the amount of this increase varies in different data integration scenarios. This difference in more considerable for DBLP1-DBLP2 and DBLP2-DBLP1 scenarios where a larger number of tuples satisfy the new query predicates. On the other hand, in Q_{A1}, Q_{A2} in $Amalgam_1$-$Amalgam_2$ scenarios, there is no improvement in recall because there is no tuple satisfying property precedence relations. This implies two conditions to increase the recall: (1) the query must reference a property for which there exists a property precedence relation; and (2) there must be tuples satisfying implicit properties obtained by applying property precedence relations. For the

same reason, there is no improvement in the recall for queries Q_{U7}, Q_{U8} in UTCS-UTDB scenarios where there is no tuple satisfying compound property precedences referenced in these queries. In the remaining scenarios, CDI improves query processing in terms of increasing the completeness. As shown in Figure 4-5, this improvement is achieved without negatively affecting precision.

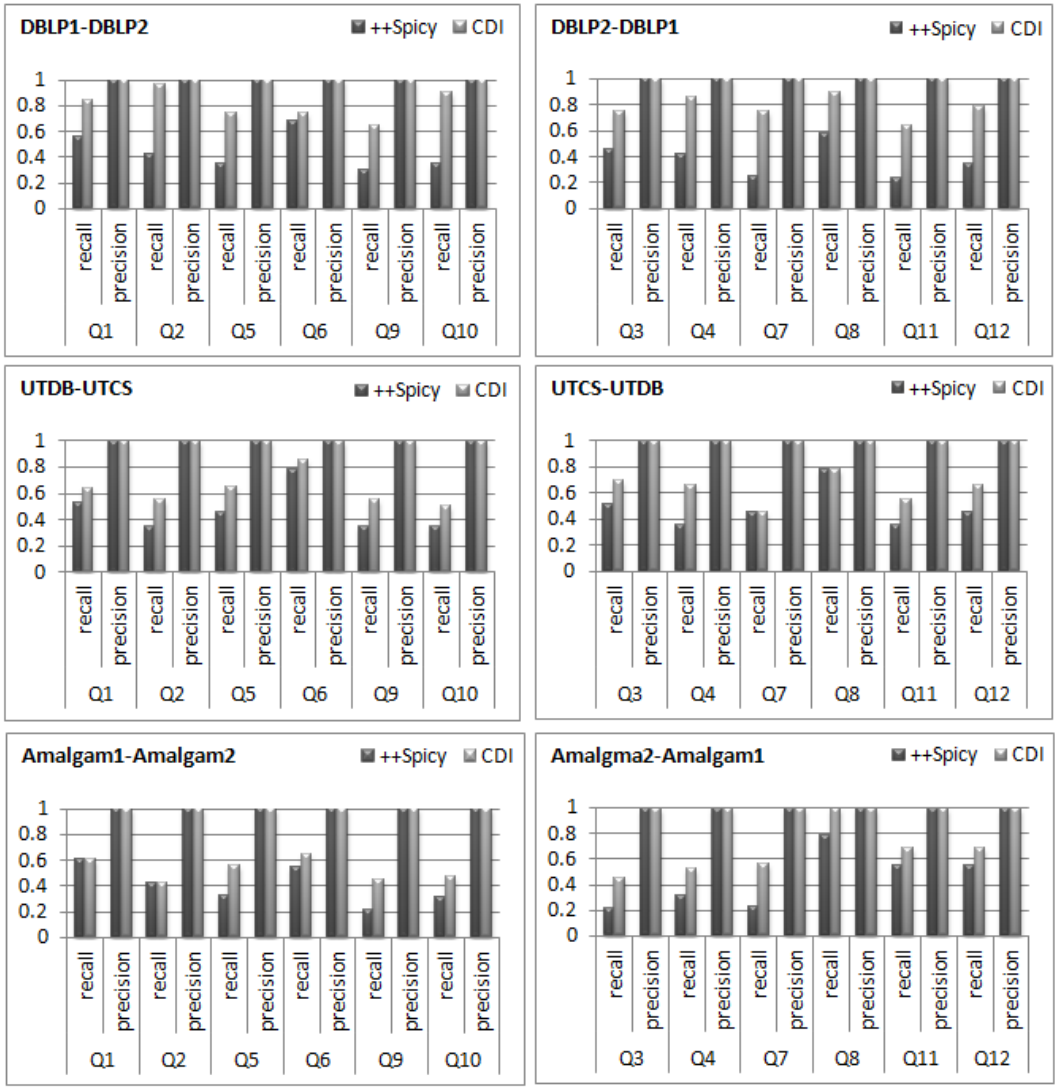


Figure 4-5: Precision and recall computed for six different data integration scenarios

4.5.4 CDI: The Tradeoff between Accuracy and Completeness

Experiments were performed to evaluate different types of query expansion techniques in CDI. These two techniques provide configurable query answering. To evaluate the query expansion techniques, we designed a data integration setting including three different scenarios. In the first scenario (Exp-Spc), only query expansion through specialization is used for query rewriting. In the second scenario (Exp-Gen), only query expansion through generalization is used. These scenarios are compared with a query rewriting without query expansion (None).

The Amalgam datasets were selected to study configurable query answering. We extracted 24 simple property precedence relations between source and target where, in the twelve of these precedences, a property in the source precedes a property in the target, and the remaining, a property in the target precedes a property of the source. Query answering scenarios were performed on the Amalgam1-Amalgam2 scenario including twelve queries already explained in Section 4.5.2. The results of experiments in different query expansion scenarios are shown in Figure 4-6 and Figure 4-7 that are elaborated in the following.

A. Query Expansion through Specialization (Exp-Spc)

As shown in Figure 4-6, in the case of using Exp-Spc recall for query rewriting, recall is increased without the loss of precision compared to the query rewriting without using Exp-Spc. This result is expected because in Exp-Spc, a generic property is replaced with a more specific property in the query predicate that still satisfies the query predicate of the original query. Since the final result is the union of the original query and rewritten

queries, some new tuples that satisfy a different manifestation of the original query predicate are added to the final result. However, there are some exceptions. In Q4, and Q9, there is no change in recall using Exp-Spc. The reason is that there is no data regarding property precedence relations corresponding to these queries. In other words, although existence of at least one property precedence relation between source and target ensures generating a new subquery, there may not exist tuples satisfying this new subquery in the source. As a result, this type of query expansion increases recall to the extent that such specific properties (preceded properties) exist in the source.

The amount of increase in recall using Exp-Spc varies in different queries depending on the number of precedences referenced in a query. The more complete a set of simple property precedence relations, the more increase in the recall for queries that reference a generic property. In the best case, when all specific manifestations of a generic property are identified, the maximum recall is achieved through query expansion. For example, in query Q2 that asks for all publications of a specific person, because all manifestations of publications (including journal paper, conference paper, manual, book, and technical report) are identified through property precedence relations between source and target schemas, the recall is increased to 1. As shown in Figure 4-6, precision is not changed using Exp-Spc because no unsound result is added to the final answers, and precision is not changed. Since precision remain intact by using Exp-Spc, increasing recall results in increasing f-measure (i.e., harmonic mean of precision and recall)

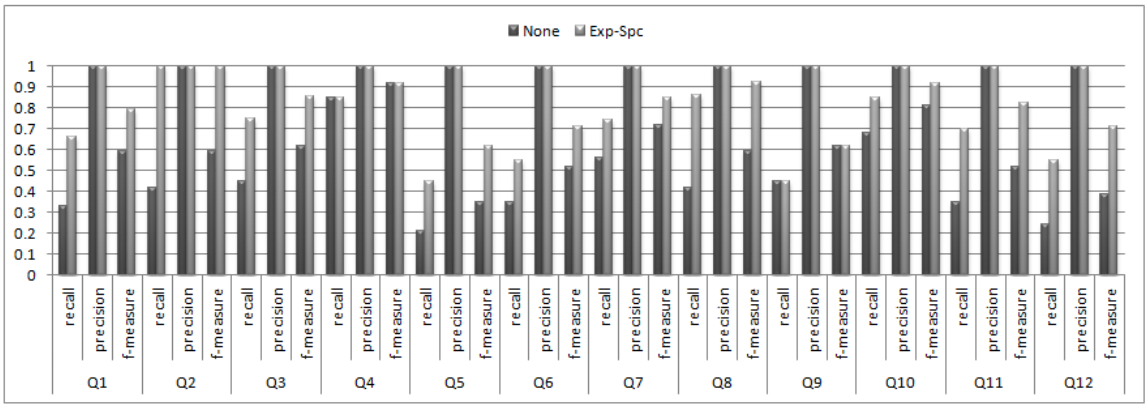


Figure 4-6: Precision and recall computed using Exp-Spc, and query rewriting without query expansion (None)

B. Query Expansion through Generalization (Exp-Gen)

In the query expansion through generalization, a specific term in a query predicate is replaced with a more general term. In the theoretical analysis, we showed query expansion through generalization can result in returning more complete results with a possibility of returning some unsound answers that may affect precision. In this section, we show how this important feature can be exploited in real data sources to provide configurable query answering through the tradeoff between accuracy and completeness.

The results of performing query expansion algorithm in the Amalgam1-Amalgam2 data integration scenario are shown in Figure 4-6. As expected, recall is increased for all queries in this scenario. Those potential tuples satisfying a query predicate that references a generic term are returned, as well as tuples satisfying the original query predicate (referencing a more specific term). For example in Query Q4, the original query addresses a specific property (i.e., select technical reports published in year 2000), where through query expansion, this property is replaced with a more general

query predicate (i.e., select articles published in year 2000). In Amalgam1, publications are classified into book, journal, manual and technical reports. On the other hand, in Amalgam2, publications are classified into books, journals and series. Without query expansion through generalization, no result is returned for the original target query because there is no direct property correspondence between *TechnicalReports* in Amalgam1 and *Series* in Amalgam2. However, this type of query expansion allows extracting publications in Amalgam2 that are potentially technical reports. The amount of increase in recall varies in different data integration scenarios depending on the number of tuples satisfying the specific property (indicated in query predicate) in comparison to the number of other specific properties preceded by the generic property.

Despite returning some new potential answers using Exp-Gen, some unsound answers are also returned in this technique. Such unsound answers can negatively affect the accuracy of results. For example, as shown in Figure 4-7, precision for all queries is reduced when this query expansion algorithm is used (except for Q5, and Q7 where there is no tuple satisfying rewritten queries in the source). The amount of this decrease varies depending on the number of property precedence relations between a generic property and specific properties as well as the number of tuples satisfying each specific property in a query. As shown in Figure 4-7, despite decreasing precision values by using Exp-Gen, f-measure is increased in most of the cases because of considerable improvements in recall values.

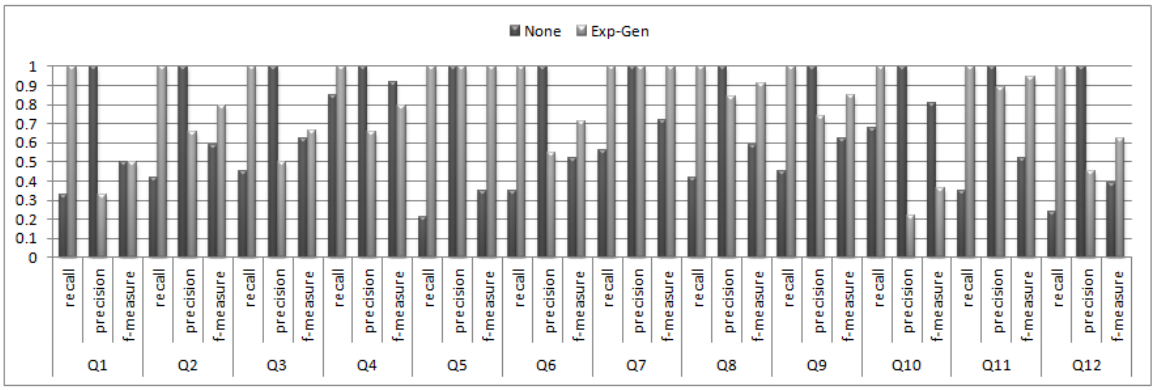


Figure 4-7: Precision and recall computed using Query Expansion through generalization (Exp-Gen) and query rewriting without query expansion (None)

To overcome the problem of unsound results, further processing and analysis is required to support business intelligence through which results are pruned using some other constraints. Extracting such constraints requires understanding the underlying data and queries. For example, in query Q4 that asks for technical reports published in 2000, the result of this query after query expansion includes some other publications such as manuals and book chapters. To prune such unsound results, it is possible to filter based on some exclusive properties of technical reports. For example, the name of the institution publishing a technical report is a property existing only for technical reports while this property is not used in journal papers and proceedings. Such analysis is one direction for extending our configure data integration technique in the future.

The tradeoff between completeness and accuracy is the main idea behind the configurable query answering where accuracy can be compromised to achieve more complete results. This approach can be useful in many applications where finding a complete set of results is critical even with generating some unsound answers. For example, identifying the complete list of potential diseases of a patient is crucial even

when some false positives results are returned. The results are required to go under further processing to reduce false alarms.

4.6 Related Work

All scenarios in which a data is accessed from multiple heterogeneous datasets entails schema mapping (Halevy, 2010). In this section we review related work in information integration.

In (Bonifati et al., 2011), a comprehensive survey of schema mapping and data integration techniques through query rewriting is proposed. Clio (Fagin et al., 2009; Miller et al., 2000; Popa et al., 2002), is one of the leading projects in data integration providing schema matching and schema mapping tools. In this system, a set of mapping expressions is generated using property correspondences and referential integrity constraints. ++Spicy (Marnette et al., 2011) is an open source implementation of Clio algorithms. In (Jiang, Ho, Popa, & Han, 2007) an extension of Clio is proposed in which mappings and data exchange processes are performed between XML data. An important challenge in XML data is addressing target constraints according to multilevel hierarchies in the target schema.

Semantic heterogeneity reconciliation in data integration is an important step for schema mapping and query answering. Different approaches have been proposed to reconcile semantic heterogeneities in schema mapping. A method to generate schema mappings through collaboration with users proposed in (Chiticariu et al., 2008). In this technique, intermediate schemas are refined based on feedback received from users. In (Fletcher & Wyss, 2006), users actively participate in mapping generation by providing

data examples. Mapping refinement using data examples is considered in (Alexe, Hernández, Popa, & Tan, 2010), where each data example is a partial specification of semantics to refine mappings. Another approach employed to add semantics in schema mapping is employing a global domain ontology (or conceptual model) to represent higher level mappings between a source and a target schema (Mena, Illarramendi, Kashyap, & Sheth, 2000). However, designing a global domain ontology or a conceptual model can be very expensive. An & Song (2008) propose a technique that looks for complex mappings and semantic associations between two conceptual models. This is performed through making a mapping graph from the cross product of graphs of each conceptual model.

As discussed in (Hassanzadeh et al., 2009), because a real world concept or characteristic can be represented via different syntactic representations in different data sources, higher level information such as domain knowledge is necessary for schema mapping. An attribute-based semantic heterogeneity reconciliation method proposed in (Parsons & Wand, 2003) that employs relations between properties instead of their actual meaning. They showed how structurally different attributes can manifest the same higher level property that consequently can be used for semantic heterogeneity reconciliation.

Ontological foundations for semantic heterogeneity reconciliation in data integration through query processing are proposed in (Sekhavat, 2012), in which a set of query rewriting rules are proposed based on property precedence relations. In (Sekhavat & Parsons, 2012a), we studied the fundamental semantic relations between properties from an ontological point of view and showed how inferring implicit properties from

existing properties using property precedence relations can be exploited to enhance schema mapping expressions. As argued in (Haas et al., 2009), schema mapping and data mapping are complementary techniques. As a result, when schema mapping and data mapping generate the same output, this represents the correctness of the mapping.

A data mapping technique is employed in (Kementsietsidis et al., 2003), in which mapping tables including a set of relations between data values in different data sources are used for semantic heterogeneity reconciliation. They proposed a method for inferring new associations from data by exploring existing associations.

Data integration through query rewriting has been widely studied for information interoperability. The basic query rewriting algorithms are already discussed in (Yu & Popa, 2004). An algorithm to rewrite a set of source to target mappings from which SQL scripts are generated to compute target instances is developed in (ten Cate et al., 2009). Schema mapping and query rewriting techniques based on the extension of classical logical formalisms such as Datalog are proposed in (Calì et al., 2009). In (Yu & Popa, 2004), algorithms proposed to rewrite a target query based on source schemas using mappings and target constraints. In the case of target constraints, data transferred using mapping expressions are checked against target constraints for consistency in the target.

There has been substantial interest for data integration in distributed database management systems. Piazza (Ives et al., 2004) is a distributed mapping system in which each peer stores mappings with other peers where each mapping is an equivalency or a subsumption between different queries. In this system, mappings as well as storage descriptions are used to rewrite a query. In Orchestra (Ives et al., 2008), mappings

generated in other systems are used to generate Datalog statements. This system performs keyword queries rather than structured queries.

Data integration with uncertainty is discussed in (Das Sarma et al., 2011) where correspondences between schema elements are accompanied with the probability of certainty. These probabilities represent to what extent a concept is similar to other concepts. In this probabilistic query rewriting, final query answers are also accompanied with probabilities that show to what extent the results are reliable. In (Halevy et al., 2006) an incremental data integration technique is proposed where sources are added with no conflict to the target, and then, the system continuously evolves to map between data sources. To deal with uncertainty, a technique that keeps top-K mappings between two schemas is proposed in (Gal, 2006) where each mapping has a probability between 0 and 1. Top-k schema mappings also used in (Magnani & Montesi, 2007; Magnani et al., 2005) to increase the recall of data integration. In this method, all possible schema mappings are created and used given probabilistic schema matching. A technique, in which query rewriting is provided through mapping between ontologies (corresponding to each data source) using SPARQL queries is proposed in (Correndo et al., 2010). In this technique, a list of entity alignments is used to rewrite a triple for fitting a new ontology.

Pottinger & Bernstein (2008) develop a method to create a mediated schema given a pair of two relational schemas and mappings between them. In this system, mappings are a set of select-project-join queries. Wang & Pottinger (2008) propose a technique to generate complex mappings that allows users to specific relations between schema elements in a generic and accurate way.

4.7 Conclusion and Future Work

In this chapter, we discussed the importance of semantic heterogeneity reconciliation in data integration, and proposed a data integration technique based on ontological foundations. We showed why current techniques relying only on equivalence relations and property correspondences are not enough to reconcile some semantic heterogeneity. To address the problem, we turned to ontology and used fundamental relations between properties called *property precedence* relations. First, we showed how schema mapping techniques can exploit auxiliary information in terms of local property precedence relations to enhance the mapping formalism. In particular, we used this information to find implicit properties of instances that can be used to bind different data sources. We introduced a new type of mapping expression called manifestation-based mapping, in which relations between source and target schemas are accompanied with some conditions in terms of possessing some specific values.

In the second part of this chapter, we proposed a query rewriting technique to rewrite target queries based on mappings expressions generated in the first phase. The technique rewrites a query posed to the target into a set of source queries where evaluating the union of these queries on the source returns the same result as running the target query on the materialized target instance. Using this information and also a set of global property precedence relations, we proposed a configurable query rewriting technique that allows tradeoff between accuracy and completeness in data integration. Two query expansion algorithms were proposed to provide this feature in query rewriting. In the case of Query Expansion through generalization, the results of experiments show that recall can be

increased when a query references a property for which there exists a property precedence relation, and also there are tuples satisfying implicit properties obtained by applying these property precedence relations. We also showed that the amount of increase in recall using query expansion through specialization depends on the number of property precedence relations referenced in a query.

Several open questions remain, including: Are property precedence relations comprehensive enough to capture all kinds of semantic heterogeneity? Is it possible to directly employ local property precedence relations for query rewriting? How can a combination of query expansion techniques proposed in this chapter be used to achieve certain levels of completeness or soundness?

4.8 References

Alexe, B., Hernández, M., Popa, L., & Tan, W. (2010). Mapmerge: correlating independent schema mappings. *Proceedings of the VLDB Endowment, 3*(1-2), 81-92.

Alexe, B., Tan, W., & Velegrakis, Y. (2008). Stbenchmark: towards a benchmark for mapping systems. *Proceedings of the VLDB Endowment*, *1*(1), 230-244.

An, Y., & Song, I. (2008). Discovering semantically similar associations (SeSA) for complex mappings between conceptual models. *Proceedings of the 27th International Conference on Conceptual Modeling*, Barcelona, Spain. 369-382. doi: 10.1007/978-3-540-87877-3_27

Bonifati, A., Chang, E. Q., Lakshmanan, A. V. S., Ho, T., & Pottinger, R. (2005). HePToX: marrying XML and heterogeneity in your P2P databases. *Proceedings of the 31st International Conference on Very Large Data Bases*, Trondheim, Norway. 1267-1270.

Bonifati, A., Mecca, G., Pappalardo, A., Raunich, S., & Summa, G. (2008). Schema mapping verification: the spicy way. *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, Nantes, France. 85-96. doi: 10.1145/1353343.1353358

Bonifati, A., Mecca, G., Papotti, P., & Velegrakis, Y. (2011). Discovery and correctness of schema mapping transformations. In Bellahsene, Z., Bonifati, A. & Rahm, E. (Ed.), *Schema Matching and Mapping* (pp. 111-147). Berlin, Heidelberg: Springer-Verlag.

Bunge, M. (1977). *Treatise on Basic Philosophy*: *the Furniture of the World*. Boston, MA: Reidel.

Calì, A., Gottlob, G., & Lukasiewicz, T. (2009). Datalog±: a unified approach to ontologies and integrity constraints. *Proceedings of the 12th International Conference on Database Theory*, St. Petersburg, Russia. 14-30. doi: 10.1145/1514894.1514897

Chiticariu, L., Kolaitis, P. G., & Popa, L. (2008). Interactive generation of integrated schemas. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Vancouver, Canada. 833-846. doi: 10.1145/1376616.1376700

Correndo, G., Salvadores, M., Millard, I., Glaser, H., & Shadbolt, N. (2010). SPARQL query rewriting for implementing data integration over linked data. *Proceedings of the EDBT/ICDT Workshops*, Lausanne, Switzerland. 4:1-4:11. doi: 10.1145/1754239.1754244

Das Sarma, A., Dong, X. L., & Halevy, A. Y. (2011). Uncertainty in data integration and dataspace support platforms. In Bellahsene, Z., Bonifati, A., & Rahm, E. (Ed.), *Schema Matching and Mapping* (pp. 75-108). Berlin, Heidelberg: Springer-Verlag.

Fagin, R., Kolaitis, P. G., Miller, R. J., & Popa, L. (2005a). Data exchange: semantics and query answering. *Theoretical Computer Science, 336*(1), 89-124. doi: 10.1016/j.tcs.2004.10.033

Fagin, R., Kolaitis, P. G., & Popa, L. (2005b). Data exchange: getting to the core. *ACM Transactions on Database Systems, 30*(1), 174-210. doi:10.1145/1061318.1061323

Fagin, R., Haas, L. M., Hernández, M., Miller, R. J., Popa, L., & Velegrakis, Y. (2009). Conceptual modeling: foundations and applications. In A. Borgida, er T., V. K. Chaudhri, P. Giorgini & E. S. Yu (Eds.), *Essays in Honor of John Mylopoulos* (pp. 198-236). Berlin, Heidelberg: Springer-Verlag. doi: 10.1007/978-3-642-02463-4_12

Fletcher, G. H., & Wyss, C. M. (2006). Data mapping as search. *Advances in Database Technology, 3896*(1), 95-111. doi:10.1007/11687238_9

Fuxman, A., Hernandez, M. A., Ho, H., Miller, R. J., Papotti, P., & Popa, L. (2006a). Nested mappings: schema mapping reloaded. *Proceedings of the 32nd International Conference on Very Large Data Bases,* Seoul, Korea. 67-78.

Fuxman, A., Kolaitis, P. G., Miller, R. J., & Tan, W. (2006b). Peer data exchange. *ACM Transactions on Database Systems, 31*(4), 1454-1498. doi: 10.1145/1189769.1189778

Gal, A. (2006). Managing uncertainty in schema matching with top-K schema mappings. *Journal on Data Semantics*, *4090*(1), 90-114. doi: 10.1007/11803034_5

Gemino, A., & Wand, Y. (2004). A framework for empirical evaluation of conceptual modeling techniques. *Requirements Engineering, 9*(4), 248-260. doi: 10.1007/s00766-004-0204-6

Gottlob, G., & Nash, A. (2008). Efficient core computation in data exchange. *Journal of the ACM, 55*(2), 9:1-9:49. doi: 10.1145/1346330.1346334

Haas, L. M. (2006). Beauty and the beast: the theory and practice of information integration. *Proceedings of the 11th International Conference on Database Theory,* Barcelona, Spain. 28-43. doi: 10.1007/11965893_3

Haas, L. M., Hentschel, M., Kossmann, D., & Miller, R. J. (2009). Schema AND data: a holistic approach to mapping, resolution and fusion in information integration. *Proceedings of the 28th International Conference on Conceptual Modeling,* Gramado, Brazil. 27-40. doi: 10.1007/978-3-642-04840-1_5

Halevy, A. Y., Rajaraman, A., & Ordille, J. (2006). Data integration: the teenage years. *Proceedings of the 32nd International Conference on Very Large Data Bases,* Seoul, Korea. 9-16.

Halevy, A. Y. (2010). Technical perspective schema mappings: rules for mixing data. *Communications of the ACM, 53*(1), 100-101.

Hassanzadeh, O., Kementsietsidis, A., Lim, L., Miller, R. J., & Wang, M. (2009). A framework for semantic link discovery over relational data. *Proceedings of the 18th ACM Conference on Information and Knowledge Management,* Hong Kong, China. 1027-1036. doi: 10.1145/1645953.1646084

Hernández, M. A., Papotti, P., & Tan, W. (2008). Data exchange with data-metadata translations. *Proceedings of the VLDB Endowment, 1*(1), 260-273.

Ives, Z. G., Halevy, A. Y., Mork, P., & Tatarinov, I. (2004). Piazza: mediation and integration infrastructure for semantic Web data. *Web Semantics: Science, Services and Agents on the World Wide Web, 1*(2), 155-175. doi: 10.1016/j.websem.2003.11.003

Ives, Z. G., Green, T. J., Karvounarakis, G., Taylor, N. E., Tannen, V., Talukdar, P. P., Pereira, F. (2008). The ORCHESTRA collaborative data sharing system. *SIGMOD Record, 37*(3), 26-32. doi: 10.1145/1462571.1462577

Jiang, H., Ho, H., Popa, L., & Han, W. (2007). Mapping-driven XML transformation. *Proceedings of the 16th International Conference on World Wide Web,* Banff, Alberta, Canada. 1063-1072. doi: 10.1145/1242572.1242715

Kementsietsidis, A., Arenas, M., & Miller, R. J. (2003). Mapping data in peer-to-peer systems: semantics and algorithmic issues. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* San Diego, CA, USA. 325-336. doi: 10.1145/872757.872798

Köpcke, H., & Rahm, E. (2010). Frameworks for entity matching: a comparison. *Data & Knowledge Engineering, 69*(2), 197-210. doi: 10.1016/j.datak.2009.10.003

Magnani, M., Rizopoulos, N., McBrien, P., & Montesi, D. (2005). Schema integration based on uncertain semantic mappings. *Proceedings of the 24th International Conference on Conceptual Modeling,* Klagenfurt, Austria. 31-46. doi: 10.1007/11568322_3

Magnani, M., & Montesi, D. (2007). Uncertainty in data integration: current approaches and open problems. *Proceedings of the VLDB Workshop on Management of Uncertain Data,* Vienna, Austria. 18-32.

Marnette, B., Mecca, G., Papotti, P., Raunich, S., & Santoro, D. (2011). ++Spicy: an open-source tool for second-generation schema mapping and data exchange. *Proceedings of the VLDB Endowment, 4*(12), 1438-1441.

Mena, E., Illarramendi, A., Kashyap, V., & Sheth, A. P. (2000). OBSERVER: an approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases, 8*(2), 223-271. doi: 10.1023/A:1008741824956

Miller, R. J., Haas, L. M., & Hernández, M. A. (2000). Schema mapping as query discovery. *Proceedings of the 26th International Conference on Very Large Data Bases,* Cairo, Egypt. 77-88.

Parsons, J., & Wand, Y. (2000). Emancipating instances from the tyranny of classes in information modeling. *ACM Transactions on Database Systems, 25*(2), 228-268. doi: 10.1145/357775.357778

Parsons, J., & Wand, Y. (2003). Attribute-based semantic reconciliation of multiple data sources. *Journal on Data Semantics, 2800*(1), 21-47. doi: 10.1007/978-3-540-39733-5_2

Parsons, J., & Chen, T. (2008). Using property precedence to enhance the effectiveness of queries on unstructured data. *Proceedings of 18th Workshop on Information Technology Systems,* Paris, France, 73-78.

Parsons, J., & Wand, Y. (2008). Using cognitive principles to guide classification in information systems modeling. *MIS Quarterly, 32*(4), 839-868.

Parsons, J. (2011). An experimental study of the effects of representing property precedence on the comprehension of conceptual schemas. *Journal of the Association for Information Systems, 12*(6), 1.

Popa, L., & Tannen, V. (1999). An equational chase for path-conjunctive queries, constraints, and views. *Proceedings of the 7th International Conference on Database Theory,* Jerusalem, Israel. 39-57.

Popa, L., Velegrakis, Y., Hernández, M. A., Miller, R. J., & Fagin, R. (2002). Translating Web data. *Proceedings of the 28th International Conference on Very Large Data Bases,* Hong Kong, China. 598-609.

Pottinger, R., & Bernstein, P. A. (2008). Schema merging and mapping creation for relational sources. *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology,* Nantes, France. 73-84. doi: 10.1145/1353343.1353357

Sekhavat, Y. A. (2012). Semantic heterogeneity reconciliation in data integration. *Proceedings of the PhD Workshop of 38th International Conference on Very Large Data Bases,* Istanbul, Turkey. 19-24

Sekhavat, Y. A., & Parsons, J. (2012). Semantic schema mapping using property precedence relations. *Proceedings of the IEEE 6th International Conference on Semantic Computing,* Palermo, Italy. 210-217. doi: 10.1109/ICSC.2012.24

ten Cate, B., Chiticariu, L., Kolaitis, P., & Tan, W. (2009). Laconic schema mappings: computing the core with SQL queries. *Proceedings of the VLDB Endowment, 2*(1), 1006-1017.

Wand, Y., & Weber, R. (1990). Mario Bunge's ontology as a formal foundation for information systems concepts. In Weingartner, P., Dorn, & G. J. W. (Ed.), *Studies on Mario Bunge's Treatise* (pp. 123-149). Atlanta: Rodopi.

Wand, Y., Storey, V. C., & Weber, R. (1999). An ontological analysis of the relationship construct in conceptual modeling. *ACM Transactions on Database Systems, 24*(4), 494-528. doi: 10.1145/331983.331989

Wang, T., & Pottinger, R. (2008). SeMap: a generic mapping construction system. *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, Nantes, France. 97-108. doi: 10.1145/1353343.1353359

Yu, C., & Popa, L. (2004). Constraint-based XML query rewriting for data integration. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Paris, France. 371-382. doi: 10.1145/1007568.1007611

Chapter 5 EDEX: Entity-Preserving Data Exchange: An Ontological Approach[1]

Abstract

Data exchange is the process of generating an instance of a target schema from an instance of a source schema adhering to source data in the target. Generally, data exchange is performed using schema mapping expressions that represent high level relations between the source and the target. In this chapter, we argue that confining the relations between source and target schemas in terms of relations between a set of classes prevents conveying the whole semantics in data exchange. We show such class level schema mappings cannot resolve some ambiguous data exchange scenarios, resulting in an incorrect data exchange. To address this problem, we propose Entity-Preserving Data Exchange (EDEX) method that reflects source entities in the target independent of classification of entities. We show EDEX can reconcile ambiguities while generates the *core* solution as the accurate and efficient solution. The experiments show EDEX outperforms other methods in terms of quality and efficiency of data exchange with a slight overhead of storage space.

5.1 Introduction

According to Ventana's report on Data Integration (San, 2012), the amount of conventional structured data is growing more than 30 percent annually in two thirds of the organizations surveyed. According to this report, more than half of organizations currently integrate six or more data sources. This shows the need for information

[1] An extended version of : Sekhavat, Y. A., & Parsons, J. (2013). EDEX: Entity Preserving Data Exchange. *Proceedings of DATA'13 International Conference on Data Management Technologies*, Reykjavík, Iceland (to appear).

integration with the goal of making high quality data available from various data sources. Accurate and complete information integration is crucial to ensure efficiency in business processes.

Information integration can be performed through *data integration* (virtualization), which is the process of querying across multiple autonomous and heterogeneous data sources, or through *data exchange* (materialization), which is taking data structured under a source schema, and generating an instance of a target schema that adheres to the structure of the target schema. In this chapter, we focus on data exchange in which an instance of the target schema reflecting the source instance is created. The prevailing approach for this process has been based on schema mappings that are high level specifications to describe relationships between database schemas (Bellahsene, 2011; Bonifati et al., 2010; Popa et al., 2002). These specifications are usually represented in a logical formalism capturing the relationships between database schemas independent of implementation details.

Many leading projects, such as Clio (Fagin et al., 2009; Miller et al., 2000; Popa et al., 2002), are based on schema mapping. Because of semantic heterogeneities among different data sources, there are some ambiguous cases that cannot be handled using the schema mapping based approach. We argue that the main problem of schema mapping based data exchange is that it only deals with schema level relations between source and target schemas. However, as argued in (Haas et al., 2009), ignoring data level relations may result in incomplete capture of relations between data sources, leading to incorrect data exchange. From the data level point of view, there has been research on data centric

heterogeneity reconciliation in data exchange called *entity resolution* (a.k.a., entity matching, duplicate identification, record linkage and reference reconciliation) (Talburt, 2011). Generally, entity resolution is used to clean data and create a consistent view of data from multiple heterogeneous and conflicting representations by identifying entities referring to the same real world object. For example, in a customer relationship management system that includes information of customers from different enterprises, finding different references pointing out to the same customers through entity resolution is crucial for effective customer management.

In spite of independent progress in schema level and data level approaches for data exchange, semantic heterogeneities are not completely resolved because of the gap between these two approaches. Such semantic heterogeneities may result in ambiguous cases in schema mappings, and consequently, improper data exchange. We contend that inability to resolve ambiguous cases in the schema mapping based approach emerges from the assumption of *inherent classification* in schema mapping. Many difficulties in information system management such as schema integration and schema evolution can be attributed to this assumption (Parsons & Wand, 2000). According to the assumption of *inherent classification*, every thing modelled in a domain of interest is treated as an instance of a class or entity (e.g., in an Object-Oriented model or Entity Relationship model). Although classification is an effort to organize knowledge about existing things, real world objects do not inherently belong to classes. According to ontological foundations about the nature of things in the real world, things (specified in terms of a set of properties) exist prior to and independent of their classification (Bunge, 1977).

We argue that, since schema mapping expressions are bounded in class definitions, they do not convey the whole semantics of data exchange. Although data exchange based on schema mapping has many advantages in data exchange, neglecting entities and data level heterogeneities can be problematic. To fill the gap between these two streams, we propose an *entity preserving* approach that focuses on preserving source entities in the target independent of classification. More specifically, given a set of entities in the source, we search for the best host relations in which source entities can be resided. An entity resides in a relation (or class) when it is a tuple of that relation (or a member of that class). Such entity residing is performed in such a way that preserves source entities without redundancy. This is different from entity resolution that determines whether two references are referring to the same real-world objects for the sake of data cleaning (Talburt, 2011).

In conventional data exchange through schema mapping, simple value correspondences (a.k.a., property correspondences) between properties, as well as integrity constraints in the source and the target, are used to generate schema mapping expressions. Then, such mappings are used to generate a target instance. However, in the *entity preserving* approach, value correspondences are directly used to find the best relations, which can reside source entities without generating schema mappings. We argue although schema mappings can describe relationships between database schemas independent of implementation details, they are not expressive enough to convey the whole semantics in data exchange. The entity preserving approach proposed in this thesis

addresses this problem by considering data level relations between entities as well as property correspondences.

In (Sekhavat & Parsons, 2013a), to resolve ambiguous mappings we proposed a schema mapping algorithm in which conceptual models are used to semantically enhance schema mappings. Although the experiments shows promising improvement in handling ambiguous cases, the quality of the final result is strongly dependent on the quality of conceptual models and matching between conceptual models and relational databases. However, the entity preserving approach proposed in this thesis is independent of conceptual models relying solely on source and target schemas, actual data (entities) in the source, and correspondences between source and target properties. In particular, the contributions of this chapter are as follows: first we show how data exchange techniques based on schema mapping expressions are not capable of handling ambiguous cases. Then, we propose Entity preserving Data EXchange (EDEX) approach which is a hybrid of data level and schema level approaches. This chapter is an extended version of (Sekhavat & Parsons, 2013b) including more details about EDEX, theoretical discussions on the quality of data exchange, and a comprehensive set of experiments. A set of algorithms is proposed to demonstrate the feasibility of implementing this approach. We show EDEX generates a core solution (Fagin et al., 2005b) in data exchange as the most efficient solution. Finally, EDEX is evaluated from two points of view. First, a set of experiments is performed to assess the quality of data exchange and to compare with two similar works. Then, the efficiency of the algorithms is evaluated in terms of executions times, scalability and storage space.

5.2 Overview and Background

5.2.1 Motivating Example: Ambiguity in Data Exchange

In practice, usually human intervention is required to analyze and validate ambiguous schema mappings. Consequently, inspecting mappings and selecting preferred ones is the responsibilities of users. A mapping expression denotes an ambiguous case when it can be interpreted in more than one way, and as a result, there is no unique way to generate the target instance based on it (Alexe et al., 2008a). One of the important ambiguous cases in data exchange occurs when a generalization relation is implemented using different techniques in source and target schemas. As shown in Figure 5-1, a generalization relation can be realized through: G1) allocating separate tables for super class and subclasses such that there is a foreign key in each subclass referencing the super class, G2) allocating a separate table for each subclass, and repeating the properties of the super class in each subclass, G3) collapsing attributes of subclasses in to the super class and allocating a single table including all properties or G4) collapsing attributes of subclasses in to the super class, which is accompanied by an additional property that explicitly indicates the subclass. A generalization relation can result in ambiguous cases for data exchange through schema mapping techniques because other types of relations including functional dependencies (e.g. a doctor observes a patient) and self-reference relations are also realized through the same technique (i.e., key/foreign key). In the following, we elaborate an important case in which schema mapping based algorithms generate ambiguous schema mappings.

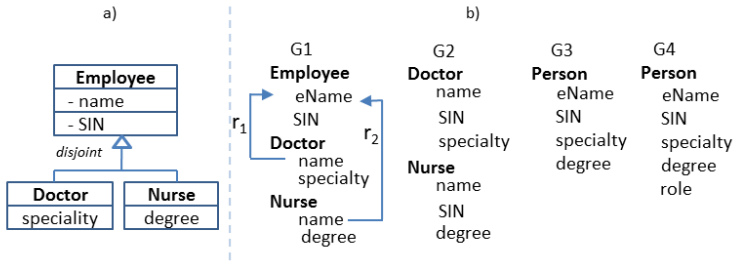


Figure 5-1: An example of a generalization relation (a) and its different implementations in the relational database (b)

One important type of ambiguous schema mapping occurs when a class in the source simultaneously refers to more than one class in the target while only one of them can be acceptable in data exchange based on the values of properties. For example, as shown in Figure 5-2, each course is taught by an instructor. On the other hand, in the target schema, a professor or a graduate student can be considered as an instructor of a course (arrows represent value correspondence between source and target properties, and dashed lines shows the referential integrity constraints). Given the source and target schemas shown in Figure 5-2, the following schema mapping expressions m_1, m_2 and m_3 are generated by ++Spicy (an open source implementation of Clio's schema mapping algorithms) (Marnette et al., 2011).

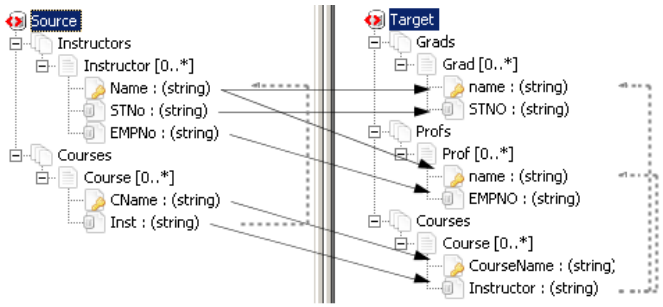


Figure 5-2: An example of a data exchange setting including source and target schemas in ++Spicy

***m₁**: for each x_1, x_2, x_3: Instructor (Name: x_1, STNo: x_2, EMPNo: x_3) → Grad (name: x_1, STNO: x_2).*
***m₂**: for each x_1, x_2, x_3: Instructor (Name: x_1, STNo: x_2, EMPNo: x_3) → Prof (name: x_1, EMPNO: x_3).*
***m₃**: for each x_1, x_2, x_3, x_4: Instructor (Name: x_4, STNo: x_2, EMPNo: x_3), Course (CName: x_1, Inst: x_4) → Course (CourseName: x_1, Instructor: x_4).*

Using these mappings, given a source instance [*Instructor(I_1, st_1, null), Instructor(I_2, null, emp_1), Course(C_1, I_1), Course(C_2, I_2)]*, ++Spicy generates the target instance as [*Grad(I_1, st_1), Grad(I_2, null), Prof(I_1, null), Prof(I_2, emp_1), Course(C_1, I_1), Course(C_2, I_2)*]. One obvious problem in the mappings generated by ++Spicy is that for a given tuple in *Instructor*, two different mappings are generated while only one of them is acceptable according to *STNo* and *EMPNo*. In other words, when *STNO* exists for an instructor in the source, the corresponding record must be generated in the *Grad* table in the target. On the other hand, when *EMPNO* exists for an instructor in the source, the corresponding record must be generated in the *Prof* table in the target. This ambiguity between m_1 and m_2 results in generating redundant information in the target while *Grad(I_2, null)* and *Prof(I_1, null)* are unacceptable. This is one of the ambiguous cases where data exchange techniques based on schema mapping expressions are not able to handle ambiguity because the values of properties determine which mappings are applicable. Relying only on schema centric information cannot resolve such ambiguities. In this thesis, we show how our entity preserving approach takes into account data and semantic heterogeneities to resolve such ambiguities.

5.2.2 Background and Formalism

This thesis follows the formalized notions proposed by (Fagin et al., 2005a; Fagin et al., 2005b). In the data exchange problem, databases are considered as sets of relations. A relational schema is a finite set $R=\{R_1,..,R_k\}$ of relations in which R_i has a fixed arity (number of properties). In the context of data exchange in this paper, the source and target schemas are disjoint, shown as $S=\{S_1,..,S_n\}$ and $T=\{T_1,..,T_m\}$, respectively. An instance I of a schema S is a set of instances over relations of S, where an instance of a relation $\{A_1,...,A_k\}$ is a finite set of tuples in form of $R(A_1:v_1,...,A_k:v_k)$. Generally, data exchange systems work based on a set of different types of dependencies to specify mappings between source and target. These dependencies include tuple generating dependencies (tgds) and equality generating dependencies (egds) (Bonifati et al., 2011). Tuple generating dependencies include source-to-target tuple generating dependencies (s-t tgds), and target tuple generating dependencies (target tgds). A s-t tgds is a dependency in which source relations are used in the premise and target relations are used in the conclusion. This type of dependency is used to specify how a tuple is created in the target given a tuple in a source (e.g., m_1, m_2 in Figure 5-1are s-t tgds). On the other hand, in a target tgd, only target symbols are used that are typically employed to specify foreign-keys in the target (e.g., : $\forall x_1,x_2\ Course(x_1,x_2) \rightarrow \exists x_3\ Prof(x_1,x_3)$). Target equality generating dependencies (egds) are used to show primary key constraints in the target (e.g., : $\forall x_1,x_2\ Prof(x_1,x_2),Course(x_3, x_2) \rightarrow x_1=x_3$).

In (Bonifati et al., 2011), a desirable target instance is defined as a legal instance satisfying correspondences between source and target and integrity constraints in the

target. Such an instance contains all source information while no piece of information is reported twice. In the schema mapping based data exchange, a mapping scenario is denoted $M=(\boldsymbol{S}, \boldsymbol{T}, \Sigma_{st}, \Sigma_t)$, where $\boldsymbol{S}$ is a source schema, $\boldsymbol{T}$ is a target schema, Σ_{st} is a set of s-t tgds (i.e., source-to-target dependencies) and Σ_t is a set of target constraints.

If I is an instance of $\boldsymbol{S}$ and J is an instance of $\boldsymbol{T}$, then J is called a solution for M and I, if I and J satisfy Σ_{st}, and J satisfies Σ_t. Formally, this is shown in the form of $J \in Sol(M, I)$ iff $\langle I, J \rangle$ satisfies dependencies in $\Sigma_{st} \cup \Sigma_t$. In particular, the data exchange problem is defined as the materialization of the target instance J over target schema T given an instance I of the source schema S, such that Σ_t is satisfied by J and Σ_{st} is satisfied simultaneously by I and J. Given $M= (\boldsymbol{S}, \boldsymbol{T}, \Sigma_{st}, \Sigma_t)$, multiple solutions may exist for a source instance because each tgd only states an inclusion constraint without indicating the content of a target instance. The concept of *universal solution*, which is a solution with several good properties justifying its selection as a target solution, is proposed in (Fagin et al., 2005a).

To formalize the notion of universal solution, we need to introduce the concept of homomorphism among two solutions. In the data exchange formalism proposed in (Fagin et al., 2005a), the set of all constant values that may occur in source instances is denoted $\underline{Const}$, and an infinite set of variables (called *labeled nulls*) are denoted $\underline{Var}$ such that $\underline{Var} \cap \underline{Const} = \emptyset$. In generating target solutions, variables are used to create new values in the target that do not already exist in the source. Each element of a tuple $t=\{a_1,a_2,...,a_n\}$ over a relation from an instance is a member of $\underline{Const} \cup \underline{Var}$. In the following, we review some

concepts in the context of data exchange. Definitions adapted from Clio (Fagin et al., 2005a) are indicated with *.

Definition 1 (homomorphism)*. Let K_1 and K_2 denote two instances over a relational schema R with values in $\underline{Const} \cup \underline{Var}$. A homomorphism $h: K_1 \rightarrow K_2$ is a mapping from $Const \cup Var\ (K_1)$ to $Const \cup Var\ (K_2)$ such that: (1) $h(c) = c$ for every $c \in Const$; (2) for every fact $R_i(t)$ of K_1, we have that $R_i(h(t))$ is a fact of K_2 where, if $t = (a_1,...,a_n)$, then $h(t)= (h(a_1),..., h(a_n))$.

Definition 2 (universal solution)*. Consider a data exchange setting $(\boldsymbol{S}, \boldsymbol{T}, \Sigma_{st}, \Sigma_t)$. If I is a source instance, then a *universal solution* for I is a solution J such that for every solution J' for I, there exists a homomorphism $h: J \rightarrow J'$

Several good properties of a universal solution justify its choice as a good target solution in the data exchange problem. This solution is considered the most general possible solution because it can be homomorphically mapped to any arbitrary solution. In addition, as discussed in (Fagin et al., 2005a), universal solutions of a given source instance are homomorphically equivalent. Generally, the Chase algorithm (Popa & Tannen, 1999) is used to generate universal solutions. This algorithm starts with a source instance and an empty target instance. In each Chase step, source-to-target dependencies and target dependencies are applied on the source and target instances as long as they satisfy these dependencies.

Among universal solutions, the solution with smallest size is called the *core solution* (Fagin et al., 2005b). Because of the minimality and uniqueness of the core

solution among universal solutions, this solution is considered as an ideal solution for data exchange. Formally the core solution is defined as:

Definition 3 (core solution)*. A target instance *J* among universal solutions is called a *core solution* if there is no proper subinstance $J' \subseteq J$ such that there is a homomorphism $h: J \rightarrow J'$.

A general technique to generate the core solution is generating a universal solution using the Chase algorithm, and then post processing the universal solution to remove redundancies. A greedy algorithm is developed in (Fagin et al., 2005b) to generate the core solution where given a source instance *I*, a universal solution *J* is generated. Then, each tuple is checked against source-to-target dependencies and target constraints to find redundant tuples. Note that a core solution is still a universal solution with the important feature that the core is the minimum sized universal solution.

5.3 Towards the Entity Preserving Approach

According to Bunge's ontology (Bunge, 1977), a domain of interest includes a set of things each possessing at least one property. An "entity" is defined as a "thing" which can be distinctly identified (Chen, 1976). For example, a specific person, company, or event is an entity. In relational database theory, a tuple (row) of a table (i.e., an ordered sequence of values, which is a finite function that maps attributes to values) can represent a particular entity, where a primary key uniquely identifies a tuple within tuples of a relation. In practice, information about one thing (physical object or a concept) may be split over several relations. For example, information about characteristics of a student can be stored in three different relations including student, department and university. On

the other hand, information about several things can be combined in a single (unnormalized) relation. For example, information about the program and courses of a student can be stored in a single profile relation. This different configuration for relations between tuples and entities is the consequence of different classifications used in various schemas. Such differences add complexity in data integration, where a data integrator needs to deal with different classification structures in data sources. In (Parsons & Wand, 2000), this problem is attributed to the assumption of *inherent classification*, where every thing in the domain of interest is treated as instance of a class.

To overcome the problem of different classification structures in the source and target, we propose a solution that preserves entities in the source regardless of classification. For this purpose, our technique identifies existing entities in the source, and then finds the best host (or hosts) for these entities with the goal of maximum information preservation and minimum redundancy.

A relational schema can be represented using a directed graph $G=(V, E)$ where $V=\{R_1,\ldots,R_n\}$ is a set of vertices representing relations (tables), and E is a set of edges. Each edge shows a directed relation from the referencing table to the referenced table.

Definition 4 (schema graph). Given a schema S, a schema graph $G=(V, E)$ is a directed graph that defines relation joinability according to foreign key-primary key relationships in S. It has a vertex R_i for each relation $R_i \in S$ and an edge from R_i to R_j for each foreign key to primary key relationship from R_i to R_j in S.

In a schema graph G, each node has a name representing a table in S, and a set of properties specifying that table. Each edge from property p_1 of R_i to property p_2 of R_j is

labelled with a pair ‹p_1, p_2› where p_1 references p_2. An example of a schema graph for the relational schema in Figure 5-3(a) is shown in Figure 5-3(b). Representing a schema using a directed graph, indirect properties of relation R_i can be found in an acyclic graph representing ancestors of R_i that we refer to as a Relation Ancestors Tree (see Definition 6 below).

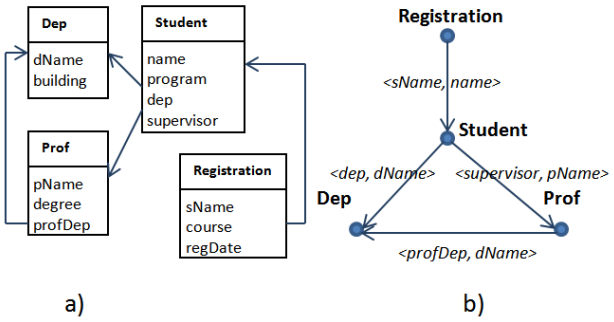


Figure 5-3: A relational schema (a) and the schema graph of this relational schema (b)

Registration		
sName	**course**	**regDate**
S1	C1	dt1
S2	C2	dt2
S3	C1	dt3
S1	C2	dt4

Prof		
pName	**degree**	**profDep**
Prof1	deg1	D1
Prof2	deg1	D1
Prof3	deg2	D2

Student			
name	**program**	**dep**	**supervisor**
S1	P1	D1	Prof1
S2	P2	D2	Prof2
S3	P3	D2	Prof3

Dep	
dName	**building**
D1	B1
D2	B1
D3	B2

Figure 5-4: An instance of the relational schema shown in Figure 5-3

Definition 5 (Neighbour). We define neighbours of a relation *r* denoted *N(r)* as a set of relations that are referenced directly by *r*. Consequently, there is an edge from *r* to any relation in *N(r)*. Accordingly, we define neighbours of a tuple *t* as set of tuples referenced by *t* denoted *N(t)*.

For example, *N(Student)* is {*Dep, Prof*}, and given *t* as the first tuple of student instance in Figure 5-4, *N(t) = {[(dName: D1), (building: B1)],[(pName: prof1), (degree: deg1), (profDep:D1)]}*.

Definition 6 (Relation Ancestor Tree). A *Relation Ancestors Tree (RAT)* of a relation *r* denoted *RAT(r)* is a sub graph of schema graph *G* with the root of *r* and all paths from *r* to *N(r)*, all paths from each relation r_i in *N(r)* to $N(r_i)$, and so on until adding a path does not result in a cycle.

RAT(r) represents all ancestors of *r* that can be extracted using the breath-first-search technique and traversing from relation *r* to all ancestors of *r*, where that node is not already visited. Relation Ancestor Tree for each relation of the schema and its schema graph in Figure 5-3 is shown in Figure 5-5.

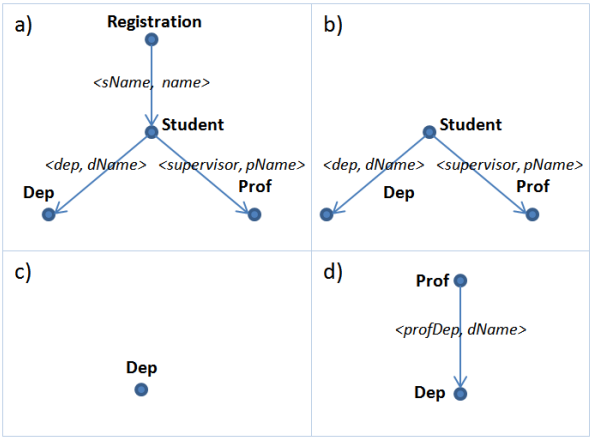


Figure 5-5: Relational Ancestor Trees for each relation of the schema in Figure 5-3

We distinguish between class level (generic) and instance level (specific) properties. A relation in a data model is represented in terms of a set of generic properties while tuples of a relation possess specific properties represented as a set of property and value pairs (‹*p, v*›). An entity possessing a specific property manifests possessing a

generic property. For example, *(gender, 'male')* and *(gender, 'female')* are two manifestations of *gender* as a generic property. We use *P(r)* to show properties of relation *r* (i.e., a set of generic properties) and *P(t)* to represent properties of tuple *t* (i.e., a set of specific properties $\langle p_1, v_1 \rangle$ specifying *t*).

Drilling down a tuple from its foreign key(s) to corresponding tuples in referenced relations (tables), it is possible to extract some indirect properties represented in neighbour relations. This way, we can extract indirect properties of *r*. For example, given the *Student* relation in Figure 5-4, *{(name, s1), (program, p1),* (*dep, D1), (supervisor, Prof1)}* is the set of properties of the first tuple of *Student*. On the other hand *Student* is referencing *Department* through (*dep, D1)* where *{(dName, D1), (building, B1)}* can be considered as indirect properties of this student. Accordingly, the properties of the supervisor {*(pName, prof1), (degree, deg1), (profDep, D1)*} can also be considered as indirect properties of this student. In addition, each *professor* tuple references a particular *department*, where properties of that department can also be considered as indirect properties of that professor.

Definition 7 (Tuple Ancestors Tree). A *Tuple Ancestors Tree* of a tuple *t* denoted *TAT(t)* is a tree with root *t* and all paths from *t* to *N(t),* all paths from each t_i in *N(t)* to $N(t_i)$, and so on until adding a path does not result in a cycle.

Using the concept of indirect properties, we introduce and define the concept of *super entity*.

Definition 8. A *Super Entity* of a tuple *t* from relation *r* (i.e., denoted *SE(t)*) is a set of specific properties of *t* as well as all indirect properties of *t* that are accessible from

TAT(t). Formally, $SE(t) = P(t) \cup P(TAT(t))$ where $P(TAT(t))$ is the set of all specific properties in *TAT(t)*.

Intuitively, if *t* is a tuple of relation *r* with no referring relation, then super entity of *t* has the same set of properties as *t*. A super entity shows complete information of a tuple including all direct and indirect properties regardless of the classification in a schema. According to Bunge's ontology (Bunge, 1977), null properties should not be considered as properties. As a result, given two different tuples of a relation, they may have different number of properties. Generating super entities can be considered as a declassification process that shows information content regardless of any structure and using a set of properties. We argue that such flat structures (in terms of a set of properties) can be used for data exchange without difficulties in handling the structure of classes in the source. Since one of the important characteristics of a good data exchange technique is the complete translation of source data in the target, we use a set of super entities of all tuples in the source for this purpose. . Regardless of which class (relation) to which entity belongs, this technique preserves entity information without redundancies.

5.4 EDEX: Entity preserving Data EXchange

In the following, we elaborate the Entity-preserving Data EXchange (EDEX) approach. The data exchange process is performed in four steps. First, we extract all super entities regarding all relations in the source schema. In the second step, redundant entities are pruned to avoid data redundancies and to speed up the data exchange process. In the third step, the best host relations for these entities in the target schema are selected given

property correspondences between source and target properties. Finally, the pruned super entities are moved to their proper host tables.

5.4.1 Step 1: Super Entity Generation

The first step towards data exchange in EDEX is extracting all super entities, as they hold the complete information regarding source entities independent of classification. According to the definition of a super entity, the super entity of a tuple is the union of properties of that tuple and properties of its Tuple Ancestor Tree. In a schema graph, an edge between node v_1 to node v_2 is a foreign key from a column of v_1 to a primary key of v_2. Each foreign key of a tuple references at most one tuple of the table referenced (In cases where more than one foreign key reference the same table, the tree includes more than one edge with different labels between those nodes).

A super entity regarding a tuple *t* is a flat structure that can be defined as a view over all ancestors of *t*. Relation Ancestor Tree *RAT(r)* is the structure that shows how this view can be built regarding tuples of relation *r*. To build *RAT(r)*, node *r* is selected as the root of this tree. Then, using the schema graph, all outgoing edges from *r* and their corresponding nodes *R* are connected to *r*. Then, for each node r_i in *R*, their outgoing edges and corresponding nodes are added to r_i if they do not already exist in the *RAT(r)*. This process continues until adding a new edge results in a loop in *RAT(r)*.

Once Relation Ancestor Trees of all relations in a schema are extracted, super entities can be extracted using view statements that can be generated by post-order traversal of these trees. In each step, leaves are joined with parent nodes. The output of this traverse is a nested view statement representing how nodes are joined. A Relation

Ancestor Tree is traversed in post-order manner such that in each step, a join between a child and its parent is formed. For the four relational ancestor trees shown in Figure 5-5, the following view statements are generated.

***Dep**: Dep*

***Prof**: Prof ⋈ Dep*

***Student**: (Student ⋈ Dep) ⋈ Prof*

***Registration**: Registration ⋈ ((Student ⋈ Dep) ⋈ Prof)*

Applying these view statements on the source instance (Figure 5-4) results in generating the following set of super entities:

e_1: {(dName, D1), (building, B1)}, src = {Dep} *e_2: {(dName, D2), (building, B1)}, src = {Dep}* *e_3: {(dName, D3), (building, B2)}, src = {Dep}*
e_4: {(name, S1), (program, P1), (dep, D1),(dName, D1),(building, B1), (supervisor, prof1), (pName, Prof1),(degree, deg1), (profDep, D1)}, src = {Student} *e_5: {(name, S2), (program, P2), (dep, D2), (dName, D2), (building, B1), (supervisor, prof2), (pName, Prof2),(degree,deg1), (profDep, D1)}, src = {Student}* *e_6: {(name, S3), (program, P3), (dep, D2), (dName, D2),(building, B1), (supervisor, prof3), (pName, Prof3),(degree, deg2), (profDep, D2)}, src = {Student}*
e_7: {(sName, S1), (name, S1), (program, P1), (dep, D1),(dName, D1),(building, B1), ((supervisor, prof1), (pName, Prof1), (degree, deg1), (profDep, D1), (course,C1), (regDate, dt1)}, src ={Registration} *e_8: {(sName, S2), (name, S2), (program, P2), (dep, D2),(dName, D2), (building, B1), (supervisor, prof2),(pName, Prof2), (degree,deg1), (profDep, D1), (course, C2), (regDate,dt2)}, src ={Registration}* *e_9: {(sName, S2), (name, S3), (program, P3), (dep, D2),(dName, D2), (building, B1), (supervisor, prof3), pName, Prof3), (degree, deg2), (profDep, D2), (course, C1), (regDate,dt3)}, src ={Registration}* *e_{10}: {(sName, S1), (name, S1),(program, P1), (dep, D1),(dName, D1), (building, B1), (supervisor, prof1),(pName, Prof1), (degree, deg1), (profDep, D1),(course, C2), (regDate, dt4)}, src ={Registration}*
e_{11}: {(pName, Prof1), (degree, deg1), (profDep, D1), (dName, D1), (building, B1)}, src ={Prof} *e_{12}: {(pName, Prof2), (degree, deg1), (profDep, D1), (dName, D1), (building, B1)} , src ={Prof}* *e_{13}: {(pName, Prof3), (degree, deg2), (profDep, D2), (dName, D2), (building, B1)} , src ={Prof}*

In addition to ‹property, value› pairs generated as the output of performing view statements, the source of each entity is also indicated in each super entity. Note that properties including null values are removed from tuples since they do not add extra information.

5.4.2 Step 2: Pruning Redundant Information

To prevent data redundancy in data exchange, the set of super entities must be pruned to eliminate repeated information. For this purpose, we introduce and use the concept of distinct super entity. A distinct super entity is a super entity possessing at least one property that does not exist in other super entities of an instance. In spite of the simple definition, finding such a distinct set of instances is not a trivial process. To extract a list of distinct super entities, a pruner algorithm (Algorithm I) is proposed to check if all elements of a super entity (the set of ‹property, value› pairs specifying that super entity) exist in at least one other super entity. A brute force search requires checking all super entities for a given super entity to check property inclusion. To avoid this naive search, the pruner algorithm for a given super entity checks only super entities extracted from neighbours of the source relation of that super entity. Recall the source relation corresponding to each super entity is already indicated for each super entity. As a result, given a schema graph the algorithm searches for inclusion only among super entities tagged as neighbours of the source of that super entity. Consequently, the search space is reduced such that only potential super entities are checked for inclusion. For example, for super entities extracted from *Dep*, only instances of *Student* and *Prof* are checked (as these are the only relations referencing *Dep*). Accordingly, only super entities extracted

from *Registration* are checked for each super entity extracted from *Student*. Nothing is checked for super entities of *Registration* as there is no relation referencing *Registration*.

One potential problem in the pruning process is the order of checking super entities for inclusion because, different checking orders may result in different output. For example, a super entity *x* including all items of super entity *y* may be removed by super entity *z*, if *z* includes all items of *x* and is processed before *y*. To address this problem, once an inclusion is found, instead of physical deleting that entity, the item is marked as "deleted". As a result, actual deleting is performed at the end of algorithm once all inclusion tests are performed.

Algorithm I (Super Entity Pruner)

Input: a list of super entities *suprEnt*
a schema graph regarding a source schema *G=(V, E)*
Output: a pruned list of super entities

```
1:  foreach super entity e1 in suprEnt
2:      src1 = the source of e1
3:      refNeighbors = a set of nodes in G referencing src1
4:      // there is no node vi in V such that vi is referencing src1
5:      If (refNeighbors == null)
6:          continue;
7:      foreach super entity e2 in suprEnt
8:          src2 = the source of e2
9:          If (refNeighbors contains src2)
10:             If (e1 is included in e2)
11:                 mark e1 as "deleted"
12: foreach super entity e1 in suprEnt
13:     If (remove e1 from suprEnt if e1 is marked as "deleted")
```

In our example, the Super Entity Pruner algorithm removes super entities e_1 as this super entity is completely included in e_4. Accordingly, e_2 is removed because of inclusion in e_5 (and e_6). Using this algorithm, $\{e_1, e_2, e_3\}$ are checked for inclusion in $\{e_4, e_5, e_6, e_{11}, e_{12}, e_{13}\}$. In the same way, $\{e_4, e_5, e_6\}$ are checked for inclusion in $\{e_7, e_8, e_9, e_{10}, e_{12}\}$.

Nothing is checked for e_7, e_8, e_9, e_{10} because their source (i.e., *Registration*) is not referenced by a relation in the schema graph.

5.4.3 Step 3: Host Selection

After generating the list of super entities in the source, we need to find appropriate relations (tables) in the target to reside these super entities. Selecting target host relations requires considering some important issues. First, the same concepts can be shown using different representations and as a result, two different properties can represent the same concept in the source and the target. To connect source and target schemas, we use property correspondences (a.k.a. value correspondence) in form of $\langle p_1, p_2 \rangle$ representing correspondence between property p_1 in the source and property p_2 in the target. Each correspondence shows that an attribute of the target is semantically related to an attribute in the source. Such correspondences can be created manually by users in a visual interface, or automatically by schema matchers using structural, statistics, string similarities and other schema matching techniques (Bellahsene, 2011). Value correspondences are already used in data exchange techniques based on schema mapping. However, in our approach, value correspondences are directly used to select best hosts for source entities regardless of schema mapping.

The second issue is the conditions for selecting best host for source entities. We propose two conditions as important. *Completeness* means residing hosts must be able to recover all properties of source entities in the target. *Non-redundancy* ensures no repetitive information is transferred to the target. To satisfy these two conditions, a host selection algorithm (Algorithm II) is proposed that is elaborated in the following. We use

the target schema shown in Figure 5-6 and the following set of value correspondences *{name ↔ stName, program ↔ prog, dName ↔ dpt, supervisor ↔ supervisor, course ↔ courseName, regDate ↔ date}* between this schema and source schema in Figure 5-3 to explain the host selection algorithm.

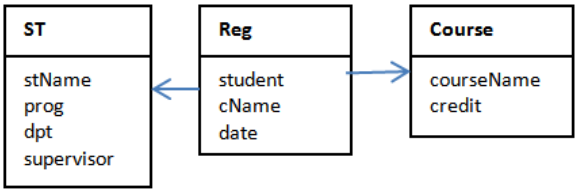


Figure 5-6: The target schema residing the source instance

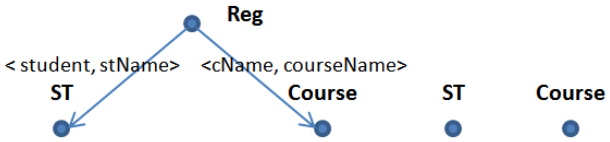


Figure 5-7: RATs constructed for each relation in the target

To select the best hosts for entities extracted from the source, we need to take into account the structure of the target schema. For this purpose, we consider Relation Ancestor Trees in the target as structures to reside super entities of the source. For this purpose, we need to extract a RAT corresponding to each relation in the target. Then, we need to check which structure can properly reside super entities of the source. Regarding the target schema shown in Figure 5-6, the RATs constructed for each relation are shown in Figure 5-7.

We assume the existence of a unique property name for each property (to ensure this, each property can be named using the triple <*dbName, tableName, proprtyName*>). We form a hash table from value correspondences where, for each correspondence ‹p_1,

p_2›, the corresponding property of a given property is accessible. The best Relation Ancestor Tree to reside a given super entity would be the one that has the maximum number of properties (class-level properties) matching the properties (specific properties) of that super entity. Among those *RATs* with the maximum number of matching properties, a *RAT* with the minimum number of total properties is selected as this *RAT* holds the minimum number of unrelated properties. This host selection process is summarized in Algorithm II.

Algorithm II (Host RAT selector)

Input: A list of source super entities *suprEnt*
Schema graph regarding the target schema $GT=(V_T, E_T)$
Hash Table *htCorr* with target properties as keys and source properties as values
Output: The list of super entities marked with host names

```
tgt_RAT_collection ← the set of RATs for each relation in GT
tgt_property_collection = a collection holding a set of
                          properties for each RAT(r)
foreach super entity e in suprEnt
    matchingCount = new HashTable();
    foreach property p in e
        foreach RAT r in tgt_RAT_collection
            If  htCorr[p] in tgt_property_collection[r]
                matchingCnt[r]++; //values are initialized with 0
        If there is a single maximum value in matchingCnt
            assign RAT (r) corresponding to this value to e
        Else If more than one r exists with max matchingCnt
            select RAT (r) with minimum number of properties and
            assign it to e
return suprEnt
```

When no RAT with a matching property is found, this indicates the target schema is not capable of hosting this super entity. Otherwise, matching at least one property ensures residing that super entity. In the prototype implemented, such cases are shown to the data exchange designer as warning messages to reconfigure value correspondences or

to modify the target schema. Given the set of super entities extracted in Step 2, the following host RATs are selected for super entities as follows:

$e_3 \rightarrow ST$, $e_4 \rightarrow ST$, $e_5 \rightarrow ST$, $e_6 \rightarrow ST$,
$e_7 \rightarrow Reg$, $e_8 \rightarrow Reg$, $e_9 \rightarrow Reg$, $e_{10} \rightarrow Reg$,
$e_{11} \rightarrow ST$, $e_{12} \rightarrow ST$, $e_{13} \rightarrow ST$

5.4.4 Step 4: Entity Residing

The final step of EDEX is residing super entities extracted from the source in host RATs indicated in Step 3. Although super entities completely included in other super entities are considered redundant, and they are removed by the pruner algorithm in Step 2, some super entities may contain information about the same entity. For example, super entity e_1 and e_2 contain information about the same department. This is an important issue in data exchange because this information should refer to the same entity in the target to avoid entity redundancy. To address this problem, we use target *egds* (equality generating dependencies) used to encode primary key constraints in the target (Casanova et al., 1982). We use these constraints to avoid inserting the same entities with different identifications by checking primary keys. For this purpose, when a request to insert in the target is made, the algorithm first checks if information regarding the unique properties already exists. If the values regarding an entity already exist, the insertion command is aborted and the primary key of the tuple already referring to that entity is returned. Otherwise the insertion is performed.

One important issue that must be considered in residing super entities is that information regarding each ancestor must be inserted before inserting information of its descendants, as each child has at least one property referring to the primary key of its

parent. In particular, the structure of the host *RAT* can provide the proper order of information insertion. For this purpose, a post-order traversal of each host RAT ensures insertion in ancestors before insertion in descendants. The details of generating insertion statements to reside a super entity in a host RAT are shown in Algorithm III. Insertion statements generated by this algorithm can be easily transformed to SQL insertion statement as is performed in our EDEX prototype. The host *RAT* is traversed in the post-order manner, and the nested structure for insertion is created. For example, given a host *RAT* for *Reg* relation, the following expression is generated representing the order of insertions.

ex_1:Reg(student (ST: stName, prog, dpt, supervisor), cName (Course: courseName, credit), date)

This structure shows the order of inserting properties, given properties of a super entity. To generate insertion statements, we start by inserting greatest ancestors. In our example, *ex_1* prescribes three insertion statements with the order of ST, Course and Reg. First, two sets of properties P_1 = (*stName, prog, dpt, supervisor*) and P_2 = (*courseName, credit*) are inserted as two nested sets of properties. For example, for super entity *e_7* (*e_7: {(sName, S1), (name, S1), (program, P1), (dep, D1),(dName, D1),(building, B1), ((supervisor, prof1), (pName, Prof1), (degree, deg1), (profDep, D1), (course, C1), (regDate, dt1)}*) with *Reg* as a target host, the algorithm first checks if there is a common property between properties of *e_7* and P_1. In this case, *{stName, prog, dpt, supervisor}* are selected as common properties. Note that value correspondences are taken into account for finding common properties (e.g., *dName* in source corresponds to *dpt* in target). Then,

regarding the primary key of the corresponding target relation *ST* (i.e., *stName*), the *ST* relation is checked to see if information related to this student is already inserted in this table. If it is not already inserted, a tuple covering these properties is inserted and the primary key of this tuple is returned as a reference. If this tuple is already inserted, then no insertion statement is generated, and *stName* is returned as a reference. In the same way, for the second nested set of properties (*Course: courseName, credit*), *courseName* is identified as common property between e_7 and ex_1. Then, *Course* table is checked to find if information regarding $cName=C_1$ is already inserted to *Course*. Finally, since there is no other nested statement, an insertion statement for *Reg* is generated. In this case, the values for *student* and *cName* reference *ST* and *Course* relations, respectively, that are already processed as nested expressions.

Algorithm III (Entity Residing)

Input: Super entity $e = \{\langle p_1, v_1\rangle\langle p_2, v_2\rangle,...\}$
Host Relation Ancestor Tree *RAT(r)*
ex = the nested expression generated from the post-order
traversal of *RAT(r)*
Seq = the order of relations from *ex* for insertion such that
inner parentheses come before outer parenthesis
HtReferences = null
foreach relation *r* in *Seq*
CP= common properties of *e* and *r*
If *CP* is null **then**
continue;
Else If information regarding *CP* is already inserted in *r*
return related reference from *HtReferences*.
Else
insert the tuple related to *e* in *r*, add the reference to
HtReferences, and return this reference

Note that, although applying target egds in data exchange prevents inserting redundant tuples referencing the same entities, it may result in losing some entities of the

source. The reason is that inserting a tuple into a table may not be possible due to integrity constraint checking for primary keys. For example, because information about students and departments is stored in the same table with *stName* as a primary key, the existence of a department depends on existence of a student. In our example, e_3 cannot be inserted to *ST* because there is no student who is assigned to this department. This is a tradeoff between data consistency and data completeness, where a designer may relax some target egds to gain complete data exchange. The algorithm proposed in this chapter prioritizes integrity constraints and does not allow breaking any target egd constraint. The most important benefit of this feature is ensuring generation of the *core solution* as an efficient solution in data exchange discussed in the next section.

5.5 EDEX and Core Solution

While a large number of techniques have been proposed to generate universal solutions, only a few have considered generating the core solution. Such solutions are schema mapping based solutions in which it is assumed that class level relations between source and target already exist in terms of schema mapping expressions. On the other hand, EDEX is a schema mapping independent technique that generates the core solution. To prove this, we first need to show EDEX generates a valid solution regarding target constraints and value correspondences between source and target. Then, we show the instance generated by EDEX is a universal solution. Finally, we discuss why the instance generated by EDEX is the minimal universal solution (i.e., core solution). Lemma I, Lemma II, and Theorem I elaborate these propositions.

Lemma I: *Given a source instance I, and a set of value correspondences* $\Sigma_{st} = V$*, EDEX generates a valid target solution.*

Proof: An instance *K* is a valid solution iff $\langle I, K \rangle$ satisfies $(\Sigma_{st} \cup \Sigma_t)$. In this definition, Σ_{st} is a set of source to target dependencies, where each $\sigma \in \Sigma_{st}$ represents a relation between one or more classes in the source and one or more classes in the target. In EDEX, Σ_{st} is a set of simple value correspondences (*V*) that are independent of classification. Since no schema mapping expression is used to generate *K*, the only items in Σ_{st} that must be satisfied are these value correspondences. In Algorithm III, given a tuple from the source, an insertion statement is exactly generated based on value correspondences. As a result, given a tuple *t* inserted to table *T*, a corresponding property is retrieved from the hash table of properties. Consequently, if there is a constant value in the source, that value is inserted to its corresponding column in the target. Otherwise, if a corresponding property is not found, nothing is inserted. In either case, $\langle I, K \rangle$ satisfies (Σ_{st}). In addition, $\langle I, K \rangle$ satisfies (Σ_t) because Σ_t includes egd relations where each insertion statement checks primary keys and unique values before insertion. Consequently, $\langle I, K \rangle$ satisfies Σ_{st} and Σ_t which is denoted $\langle I, K \rangle$ satisfies $(\Sigma_{st} \cup \Sigma_t)$. As a result, the instance *K* generated by EDEX is a valid solution.

Lemma II: *Given a source instance I, EDEX generates a universal solution in the target.*

Proof: According to (Fagin et al., 2005a), a solution *K* is a *universal* solution if it is a valid solution, and it is derived from the source instance. The first condition is already satisfied using Lemma I. The second condition ensures that no extra information (that

does not exist in the source) is generated in the target solution. A solution K is a universal solution if there is a homomorphism from K to all other valid solutions. Homomorphism is a constant-preserving mapping used to turn an instance into a subset of another instance. Suppose there is a solution K_1 such that there is no homomorphism from $K \rightarrow K_1$. This means that there exists a tuple t in K for which there is no tuple t_1 in K such that $h(t)=t_1$. In this case, for each item in tuple t, one of the following statements is true: 1) there exists a constant value c such that c is mapped to a labelled null; or 2) c is mapped to c_1 such that c_1 is not in I. In the first case, Algorithm III ensures that constant c must be mapped to constant c_1, if there is a value correspondence between c and c_1. As a result, c cannot be mapped to a labelled null. In the second case, this means K_1 is not derived from the source I and as a result, K_1 is not a solution. Consequently, there is a homomorphism from K to every possible solution.

Theorem I*: The solution generated by EDEX is a core solution.*

Proof: A core solution K is the minimum solution among all possible universal solutions. The greedy algorithm proposed in (Fagin et al., 2005b) generates a core solution from a universal solution by stoping removing redundant tuples from the input universal solution when there is no tuple $R(t)$ in the input solution such that $k\text{-}\{R(t)\}$ satisfies Σ_{st}. Assume that, for a given source instance I, a universal solution K is generated by EDEX. Now, suppose K is not a *core* solution. In this case, there is a $R(t_1)$ such that there exists a homomorphism h: $R(t) \rightarrow R(t_1)$. Suppose $R(t)$ includes $Const_1 \cup Var_1$ and $R_1(t)$ includes $Const_2 \cup Var_2$. Since there is a homomorphism from $R(t)$ to $R(t_1)$, each constant c in $Const_1$ is mapped to c in $Const_2$. On the other hand, a variable v in Var_1 is

mapped to a variable v_2 in Var_2 or a constant c_2 in $Const_2$. $R(t_1)$ is removed only when there is a mapping between variable v and constant c in t. As a result, there is $\sigma \in \Sigma_t$ such that c references a primary key. This means the primary key of $R(t)$ is a variable, while the primary key inserted for a $R(t_1)$ is a constant. However, Algorithm III prevents this situation. As a result, there is no $R(t_1)$ satisfying the homomorphism $h: R(t) \rightarrow R(t_1)$. Consequently, K is a *core* solution.

5.6 Ambiguity Resolution in EDEX

In this section, we discuss how EDEX can address ambiguous cases in data exchange that are not resolved in many schema mapping based solutions. In the data exchange example discussed in Section 5.2, we showed how ambiguous schema mapping expressions result in generating an incorrect target solution. In particular, we showed how ++Spicy (Marnette et al., 2011) (as a representative of schema mapping based algorithms) is not capable of handling ambiguous cases when a class in the source simultaneously refers to more than one class in the target, while only one of them is acceptable. For this example, we show how EDEX can resolve such ambiguities.

For the source instance [$Instructor(I_1, st_1, null)$, $Instructor(I_2, null, emp_1)$, $Course(C_1, I_1)$, $Course(C_2, I_2)$] in the example in Section 5.2, the following super entities are generated in the first step of EDEX.

$e_1 = \{(Name, I_1), (STNO, st_1)\}, src = \{Instructor\}$
$e_2 = \{(Name, I_2), (EMPNO, emp_1)\}, src = \{Instructor\}$
$e_3 = \{(CName, C_1), (Inst, I_1), (Name, I_1), (STNO, st_1)\}, src = \{Course\}$
$e_4 = \{(CName, C_2), (Inst, I_2), (Name, I_2), (EMPNO, emp_1)\}, src = \{Course\}$

In step 2, the pruner algorithm of EDEX removes e_1 and e_2 as they are completely included in e_3 and e_4 respectively. Consequently, e_3 and e_4 are super entities containing all information of the source instance. In step 3, the host selection algorithm identifies the following RATs in Figure 5-8 as candidate structures for residing e_3 and e_4.

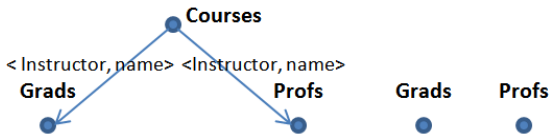


Figure 5-8: RATs for the target schema

Given value correspondences (shown by arrows in Figure 5-2), the number of properties matching e_3 and *RAT(Course)* is four, between e_3 and *RAT(Grad)* is two, and between e_3 and *Rat(Prof)* is two. Consequently, *RAT(Course)* is selected as a host RAT for e_3. Accordingly, *RAT(Course)* is selected for e_4. In Step 4, Algorithm III generates a nested structure by post order traversal of *RAT(Course)* which is:

ex_1: Course(CourseName, Instructor[(Grad: name, STNO), (Prof: name, EMPNO)])

In the case of residing super entity e_3 in *RAT(Course)*, *(Grad: name, STNO)* and *(Prof: name, EMPNO)* are two substructures at the same level that can be a choice for residing e_3. The number of matching properties between properties of e_3 and *(Grad: name, STNO)* is two, while the number of matching properties between properties of e_3 and *(Prof: name, EMPNO)* is one. Consequently, *Course (CourseName, Instructor [(Grad: name, STNO)])* is selected for residing e_3. Accordingly, *Course (CourseName, Instructor [(Prof: name, EMPNO)])* is selected for residing e_4. Finally, the entity residing algorithm inserts *[(name, I_1), STNO(st_1)]* in the *Grad* followed by inserting [*(CourseName, C_1)* and *(Instructor, I_1)*] to *Course*. In the case of residing e_4, the entity

reside algorithm inserts [*(name, I_2), EMPNO(emp_1)*] in *Prof* and then [*(CourseName, C_1)* and *(EMPNO, emp_1)*] is inserted to *Course*. Unlike ++Spicy (Marnette et al., 2011) that generates two distinct values for *Prof* and *Grad* (because of ambiguity in interpreting two mapping expressions at the same level), EDEX generates a correct target solution by identifying proper corresponding entities in the source. This example shows the importance of considering data level heterogeneities in addition to schema level heterogeneities to resolve ambiguous mappings in data exchange.

5.7 Evaluation

EDEX, including the set of algorithms proposed in this chapter, is implemented in a working prototype using Java. Generally, evaluating data exchange systems is challenging as these systems work based on mapping expressions where there is no standard methodology for mapping generation and evaluation. In addition, there are factors unique to each mapping system in which different measures are used to assess performance (Alexe et al., 2008b). In this section, we report the results of a wide variety of experiments performed to evaluate EDEX. Our prototype is compared with ++Spicy (Marnette et al., 2011), one of the leading data exchange systems based on schema mapping (which is an open source implementation of Clio) and also with SESM (Semantic Enrichment of Schema Mapping), which is an earlier version of our data exchange prototype based on schema mapping in which mappings are enhanced using conceptual models (Sekhavat & Parsons, 2013a). Post-processing techniques are not considered in our comparisons because of the deficiency of these techniques to generate core solution. As discussed in (Mecca et al., 2012), starting around 10k tuples, the

experiments take on average several hours as they exhaustively look for homomorphism among tuples generated in the canonical solution in order to remove variables.

The experiments cover the quality of data exchange as well as the performance of algorithms. In the first phase of evaluations, we study the quality of data exchange in terms of accuracy and completeness of the target instance generated by EDEX. In the second phase, we evaluate performance of EDEX in terms of execution time, scalability to handle different types and sizes of data sources, and finally, storage space required to perform EDEX. Various real and synthesized data source are used in the experiments elaborated in the following.

5.7.1 Experimental Settings

EDEX stores source and target instances as well as intermediate super entities extracted during data exchange process in SQL Server 2012. However, our prototype is implemented using a layered architecture in which the data access layer is implemented as an independent component. In this architecture, the underlying database engine can be replaced with any other database engines by modifying the data access layer. In the second phase of the experiments in which the focus is on performance, the experiments are also performed on a version of EDEX that employs our Sliced Column Store database (SCS) (Sekhavat & Parsons, 2012) to improve performance. We will discuss how EDEX can exploit some unique features of SCS to efficiently process intermediate super entities, and consequently, speed up the data exchange process.

The overall experimental setting, including components, inputs and outputs of ++Spicy (Marnette et al., 2011), SESM (Sekhavat & Parsons, 2013a) and EDEX is shown

in Figure 5-9. ++Spicy and many other schema mapping techniques exploit a graphical user interface by which users can specify and manipulate attribute correspondences. Then, these correspondences are stored in XML files. On the hand, in SESM and EDEX, value correspondences are specified and stored manually in XML files that are loaded into memory before data exchange. EDEX does not have a graphical user interface to define correspondences, as our focus is on studying the effect of using an entity preserving approach on ambiguity resolution. ++Spicy and SESM store mapping scenarios in XML files while mappings are not generated in EDEX, and value correspondence are directly used to generate a target instance. However, EDEX requires an auxiliary storage to store super entities temporarily generated during data exchange. To reduce the extra space required to store super entities, EDEX employs a "process-as-generate" technique elaborated in Section 5.7.4.

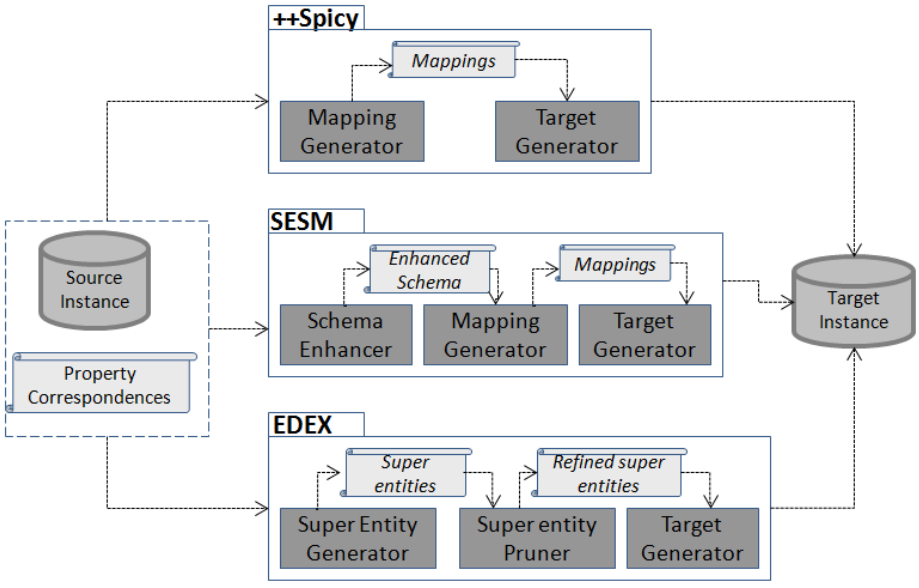


Figure 5-9: The overall architecture of entity preserving approach

5.7.2 Datasets

Datasets from variety of domains are used in the experiments. In addition, a set of synthesized data sets is used to measure the scalability of EDEX. In particular, six pairs of data sources including (UTDB, UTCS), (DBLP1, DBLP2), (NetworkA, NetworkB), (Amalgam1, Amalgam2), (Mondial1, Mondial2), (NBA_Official, NBA_Yahoo) are used where each pair includes two datasets in the same domain storing data using two different schemas. These data sources are relational data sources provided by AIM lab in computer Science department of University of Oregon. UTCS and UTDB are two databases about the Computer Science department and the database group at University of Toronto. DBLP1 and DBLP2 are relational schemas for the DBLP bibliography representing this domain using different schemas. NetworkA and NetworkB are two relational data sources about network configuration describing nodes and connections in a local area network. Amalgam1 and Amalgam2 are test datasets developed in Clio project. Mondial1 and Mondial2 are two databases including countries and their geographical features. Some of SQL scripts to generate these data sources had syntax problems that are manually fixed before materialization. The details of these data sources are shown in Table 5-1. Schema mapping expressions between data sources in each pair are also provided by AIM lab.

In the second phase of experiments, to study the scalability and storage space required to perform EDEX we used a modified version of our data generator (MUNDgen) (Sekhavat & Parsons, 2012) to provide datasets of varied complexity and size. This data generator takes into account many factors, such as average number of tuples in the base and dependent tables and redundancy in non-unique properties.

Table 5-1: Characteristics of datasets employed in the experiments

Database	*Number of tables*	*Avg # of prop in each table*	*Total # of tuples*	*Avg # of tuples in each table*	*Avg depth of RATs*	*Max depth of RAT*	*Avg # of edges*
UTCS	8	4.00	632	79	1.00	2	1.50
UTDB	13	4.23	3242	249	0.90	2	1.23
DBLP1	22	3.04	928	42	1.09	3	2.13
DBLP2	8	2.62	538	67	1.75	3	2.37
NetworkA	18	4.33	1446	80	0.94	2	1.11
NetworkB	19	2.63	1521	80	1.84	3	2.47
Amalgam1	15	6.73	570	38	0.46	1	0.93
Amalgam2	27	1.96	5130	190	0.62	1	0.81
Mondial1	28	3.64	1825	65	0.50	2	1.00
Mondial2	26	4.57	1812	69	1.61	4	2.30
NBA_Official	3	7.60	473	157	0.66	1	1.00
NAB_Yahoo	4	6.50	668	167	1.25	2	1.50

5.7.3 Phase 1: Quality of Data Exchange

In Section 5.2.1, we discussed how different implementations of a generalization relation can result in ambiguous interpretation for data exchange. In the first phase of evaluation, we performed a set of experiments to assess the quality of the data exchange process provided by EDEX in terms of the quality of target instance generated. Through this study, we have two important goals. First, we aim to determine to what extent EDEX is capable of handling ambiguity in comparison to ++Spicy and SESM when generalization relations are implemented using different techniques in relational data sources. Second, we want to know if such ambiguities exist in real data sources, and if they exist in to a sufficient degree to make a considerable improvement in data exchange. In (Sekhavat & Parsons, 2013a), we studied the importance of handling ambiguities in schema mapping in a sample healthcare data source, and we showed how ignoring this ambiguity can result in incorrect schema mappings that consequently generate incorrect target instances. Then, we showed how SESM can address this problem by semantically enriching schemas using

conceptual models. The main drawback of the approach used in SESM is that the quality of target instance is strongly dependent on the quality of conceptual models. However, the need for properly mapping between a conceptual model and a relational schema might limit the usage of this approach. Note that SESM is only a schema mapping generator tool rather than a data exchange tool. As a result, we used for this evaluation an enhanced version of SESM such that generates a target solution given a source solution and schema mappings generated by this system.

Two ambiguous scenarios sc_1 and sc_2 based on different implementation of generalization relations were identified. In scenario sc_1, a generalization relation is realized through technique G3 in the source and technique G1 in the target (see Figure 5-1). In scenario sc_2, a generalization relation is realized through technique G4 in source, and technique G1 in the target. These are error prone scenarios that result in generating ambiguous mappings. Given six pairs of data sources in Table 5-1, we formed a setting including 12 scenarios where each data source can appear as a source or a target. Each scenario is shown in the form of db_1-db_2 where db_1 and db_2 represent source and target, respectively. Inputs of each scenario are a source instance (including data and schema), a target schema and a set of property correspondences between source and target. The output is compared with the existing target data source, which serves as a gold result. In the case of ++Spicy and SESM, mappings already provided by AIM lab are used as best schema mapping expressions between source and target schemas.

A. Measures

In our experiments, accuracy and completeness is measured at two different levels. First, as shown in Table 5-2, we consider what portion of tables in the target schema is affected in the case of existing ambiguity caused by different interpretation of generalization relations. Then, at a fine grained level, we measure what portion of tuples of a data source is affected by this ambiguity (Table 5-2). These experiments allow studying to what extent ambiguity resolution can improve data exchange quality.

The quality of data exchange at the second level is measured in terms of precision and recall with respect to the target instance generated by a system and the target instance already existing as a gold result. For a given pair of datasets, let P denote the set of tuples existing in the target (in all existing tables) without considering the mappings and data exchange. Let R the set of tuples generated in the target given a source instance using a data exchange method (including correct and incorrect tuples). For each table, precision is computed as $(P \cap R)/R$ and recall is computed as $(P \cap R)/P$. A tuple is considered to be correct if its properties exactly match the properties of its corresponding tuple in the golden result. A key issue to validate EDEX is to show this system can increase precision and recall in the case of ambiguities while still generating correct instances for the rest of the cases similar to ++Spicy and SESM.

B. Results and Discussion

The number of error prone scenarios identified in each source-target scenario is shown in Table 5-2. Among 12 scenarios in this table, six scenarios include ambiguous cases that affect tuples generated in the target. For example in UTCS, *staff* is realized through

technique G4 where *adminStaff, academicStaff* and *technicalStaff* represent three different subtypes of staff. On the other hand, in UTDB, technique G1 is employed to realize this class, where a single *faculty* class is used for this purpose. In this case, ++Spicy generates three ambiguous mappings (mapping *adminStaff* to *faculty*, mapping *academicStaff* to *faculty*, and mapping *technicalStaff* to *faculty*) that must be resolved manually by a mapping designer. On the other hand, EDEX directly materializes each tuple of *adminStaff, academicStaff* and *technicalStaff* in the target based on property correspondences without need to deal with such ambiguous mappings. In this example, SESM generates three different mappings, including specific properties, that indicates which mapping must be applied regarding the values of those properties.

Existence of an ambiguous scenario in a db_1-db_2 setting does not entail that the reverse scenario (db_2-db_1) necessarily includes ambiguity. Similar to EDEX, SESM is also capable of handling ambiguous scenarios sc_1 and sc_2 and therefore, the result of experiments on SESM is similar to EDEX. As shown in Table 5-2, existence of an ambiguous scenario negatively affects recall as well as precision. However, the amount of this effect is different on various data sources. For example, in scenario 5, (NetworkA-NetworkB*)* although 26% of tables are negatively affected by scenario sc_1, this has 3% and 2% negative effect on precision and recall, respectively. On the other hand, in the reverse scenario (NetworkB-NetworkA), although only 16%of tables are negatively affected, precision and recall are reduced to 78% and 75%, respectively. This is due the fact that the tables affected in scenarios 5 have less tuples in comparison to the number of tuples in remaining tables. However, in scenario 6, problematic tables possess a large

number of tuples. This is an important observation indicating that the amount of improvement in the quality of data exchange using EDEX is a function of the number of ambiguous scenarios, the number of tables affected, and the number of tuples in these tables in comparison to the total number of tuples in a data source.

Table 5-2: The number and percent of tables negatively affected in ambiguous scenarios as well as the effect of error prone scenarios on precision and recall

Scenario #	*Source-Target*	*# of tables affected*		*% of tables affected*		*Avg of precision for each table*		*Avg of recall for each table*		*# of error prone scenarios*	
		++Spicy	**EDEX**	**++Spicy**	**EDEX**	**++Spicy**	**EDEX**	**++Spicy**	**EDEX**	**Sc_1**	**Sc_2**
1	*UTCS-UTDB*	4	-	%30	-	0.86	1	0.81	1	-	2
2	*UTDB-UTCS*	3	-	%37	-	0.88	1	0.87	1	-	1
3	*DBLP1-DBLP2*	2	-	%25	-	0.85	1	0.89	1	3	-
4	*DBLP2-DBLP1*	-	-	-	-	1	1	1	1	-	-
5	*NetworkA-NetworkB*	5	-	%26	-	0.97	1	0.98	1	2	-
6	*NetworkB-NetworkA*	3	-	%16	-	0.78	1	0.75	1	1	-
7	*Amalgam1_Amalgam2*	-	-	-	-	1	1	1	1	-	-
8	*Amalgam2_Amalgam1*	3	-	%20	-	0.81	1	0.83	1	-	1
9	*Mondial1-Mondial2*	-	-	-	-	1	1	1	1	-	-
10	*Mondial2-Mondial1*	-	-	-	-	1	1	1	1	-	-
11	*NBAOfficial-NBAYahoo*	-	-	-	-	1	1	1	1	-	-
12	*NABYahoo-NBAOfficial*	-	-	-	-	1	1	1	1	-	-

5.7.4 Phase 2: Efficiency

In the second phase of evaluations, we first study the execution time of EDEX to generate target instances for various real world data sources. We aim to see if ambiguity resolution in EDEX is achieved with an extra cost in terms of execution time. Then, we study the scalability of EDEX to determine to what extent EDEX can handle various sizes of data sources, and if EDEX scales with respect to a large number of tuples. All experiments were performed on a PC with an AMD (Athlon) 64 X2 dual core processor 2.71GHz CPU and 2GB RAM. Results reported in each scenario is the average of three run performed on a fresh start of the program.

A. Execution times for target instance generation

In the first phase of evaluations, we showed EDEX can improve the quality of data exchange. In this section, we determine whether this improvement is achieved with an extra expense of processing time, or whether EDEX can also improve efficiency in terms of execution time. For this purpose, the experiments were performed on the data sources elaborated in Table 5-1 and compared with the results of experiments on ++Spicy and SESM. The scenarios elaborated in Table 5-2 include ambiguous and unambiguous data exchange scenarios to find if the existence of an ambiguity in a data exchange scenario negatively affects processing time. From a theoretical point of view, we do not expect a considerable difference in the behaviour of EDEX in the case of existing or not existing ambiguity because EDEX follows the same approach in either case. However, we expect to see a difference in the behavior of EDEX to process different data sources since each schema can result in generating different number of intermediate super entities. In particular, we expect that the number of Relation Ancestor Trees in source and target and also the depth of them affect the execution time as they directly affect the number of super entities. Specifications of data source in terms of number of tables, average depth of RATs, the maximum depths of RATs, and average number of edges in each RAT, are shown in Table 5-1. In the case of using ++Spicy and SESM, execution time is calculated as the total time to generate a target instance including generating schema mapping expressions and generating the target instance.

Execution times for 12 data exchange scenarios in Table 5-2 is shown in Figure 5-10. These scenarios include ambiguous and unambiguous cases. As expected, there is

no difference between the behaviour of EDEX to deal with scenarios including or not including ambiguity. According to Table 5-2, scenarios 3, which is an ambiguous scenario, has less execution time in comparison to its reverse scenario (scenario 4), which is an unambiguous scenario. On the other hand, scenario 8, which is an ambiguous scenario, has more execution time in comparison to its reverse scenario (scenario 7), which is an unambiguous scenario. These observations support the idea that existence of ambiguity in a data exchange scenarios does not affect the execution time of EDEX.

In the case of SESM, preprocessing schema mapping expressions to enrich schemas with conceptual models before target instance generation is an extra overhead added to execution time compared to ++Spicy. As shown in Figure 10, the experiments support this claim where the execution time of SESM is more than the execution time of ++Spicy. In Scenarios 11 and 12, this difference is negligible due to few numbers of tables and relations in the data sources of these scenarios.

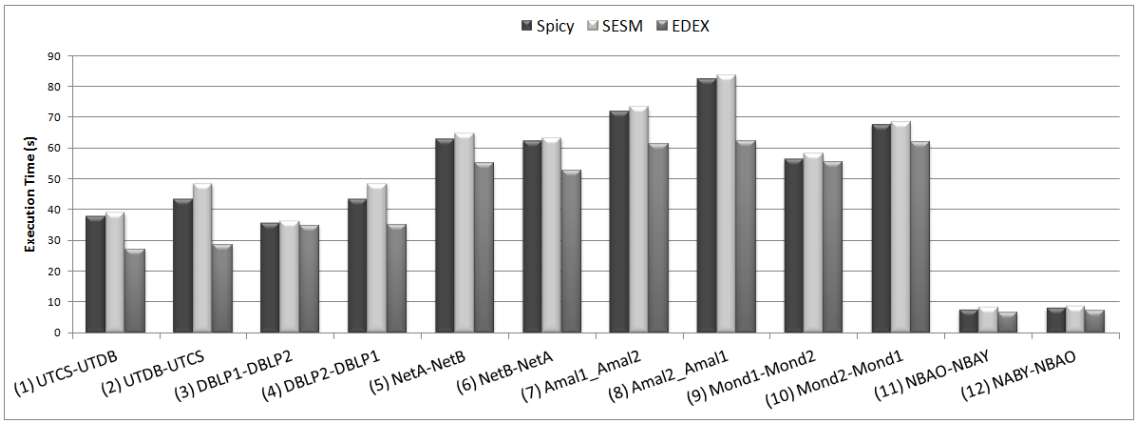


Figure 5-10: Execution time of ++Spicy, SESM and EDEX in Data exchange scenarios shown in Table 5-1

As shown in Figure 5-10, EDEX has less execution time compared to ++Spicy and SESM where in eight scenarios (1, 2, 4, 5, 6, 7, 8) this difference is more

considerable. Generally, core computation in schema mapping based approaches is known as an expensive process due to the need for post processing instances that are generated by applying mapping expressions. However, there is no need for this expensive process in EDEX because instead of an intermediary step to remove homomorphism from a canonical solution, EDEX directly generates the core solution without post processing of the canonical solution or preprocessing schema mapping expressions. This improvement is achieved with the cost of extra space required to store super entities as discussed and elaborated in Section D.

As discussed earlier, EDEX is capable of generating the target solution even without using the redundancy pruner component. Because unique values are checked before insertion in the target using target egds, redundant tuples are not inserted when a super entity is included in another super entity. The main purpose of the redundancy pruner component is improving the efficiency by reducing the processing time for checking redundant super entities. However, during the experiments, we noticed that in some cases, the time spent to check redundancy is more than the time required to check duplicate tuples. This may happen when the number of tuples in independent tables (not referencing other tables) is much less than the number of tuples in dependent tables, which consequently results in little redundancy detection. To study this problem, we performed a set of experiments using a simple data schema in the source including a master and a details table where details table references a master. This data source was mapped to a single flat table in the target. We measured execution times for the complete version of EDEX and the version without the redundancy pruner component. For a

constant number of tuples in the master table, we varied the number of tuples in dependent tables to study how redundancy remover algorithm can affect the overall performance of EDEX. The results of this experiment are shown in Figure 5-11.

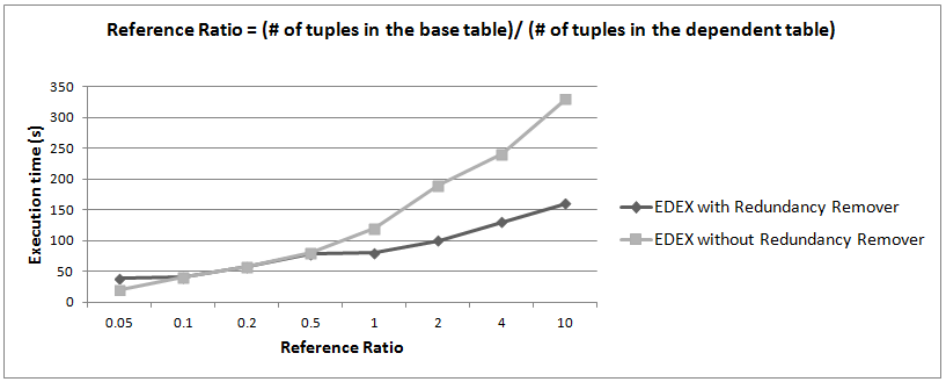


Figure 5-11: The effect of using redundancy remover component on execution time of EDEX

An interesting observation in Figure 5-11 is that removing redundant tuples can significantly reduce the data exchange time because redundancy checking (where each checking is a query on the database) is not performed. However, as shown in Figure 5-11, using the redundancy pruner component may negatively affect the performance when the *reference ratio* (which we define as (number of tuples in the base table)/(number of tuples in the dependent table)) is lower than 0.5. In this state, there are no considerable redundancies while checking super entity inclusion is an extra overhead. In addition, as shown in Figure 5-11, for lower values of reference ratio, this negative effect is substantial. To deal with this problem, we added a tuner module to EDEX such that, before application of the redundancy pruner algorithm on a specific set of super entities, the systems checks the reference ratio given the number of tuples in master and details tables to decide if it is worth applying the redundancy pruner component.

As shown in Figure 5-10, there is no considerable improvement in the execution time of EDEX in scenarios 3, 9 and 10 compared to ++Spicy and SESM. We attribute this deficiency to the high average depth of RATs in datasets of these scenarios that consequently results in a large number of intermediate super entities. In addition, due to the low reference ratio in these data sources, the tuner component does not prescribe running the redundancy pruner algorithm.

B. EDEX on SCS (Sliced Column Store Database)

Although EDEX outperforms two other systems in execution time, we believe there is still room to reduce execution time. EDEX is an I/O bound application due to the large number of database queries. We decided to replace the database engine with a read-optimized engine to improve query performance for processing super entities. Originally, EDEX stores all intermediate super entities in a SQL Server 2012 database engine which is row-store database. As argued in (Stonebraker et al., 2005), column-store databases can improve read oriented queries. As a read-optimized database, we used our Sliced Column Store (SCS) (Sekhavat & Parsons, 2012) engine to improve query processing in EDEX. In SCS, property columns are horizontally partitioned to slices where each slice stores id of tuples possessing a particular value of that property. The data access layer of EDEX modified such that all super entities are stored and processed using SCS.

Not surprisingly, as shown in Figure 5-12, using SCS results in a considerable improvement in data exchange performance. The improvement is more significant in scenarios 3, 4, 9 and 10 where a large numbers of super entities are generated due to the large number of RATs and also higher depths of RATs compared to other data sources. In

addition, due to the large number of categorical properties used in these data sources, SCS can better exploit the advantages of slicing for these data sources.

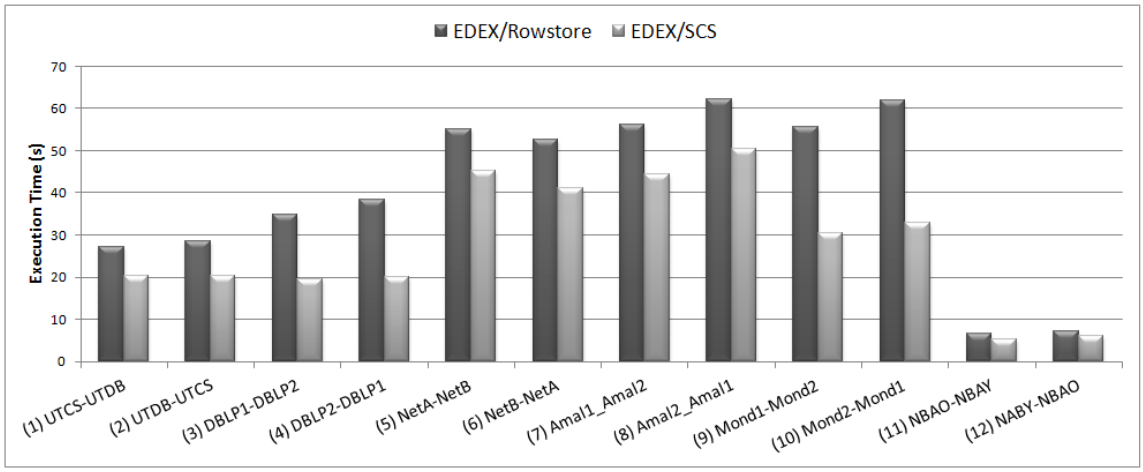


Figure 5-12: Execution time of EDEX using SQL Server 2012 (row-store) and SCS (Sliced Column Store)

C. Scalability

To evaluate the scalability of EDEX, we designed experiments to test this system in terms of the number of tuples, average depth of RATs and average number of edges in RATs. For this purpose, we used a modified version of our data generator (MUNDgen) to generate data sources of various sizes and specifications (Sekhavat & Parsons, 2012). In the first set of experiments, to evaluate the scalability of EDEX in terms of the number of tuples, we generated a set of synthetic data sources with different sizes based on the NBA Schema. Similar to (Mecca et al., 2012) we performed EDEX on datasets including 10k, 100k, 250k, 500k, and 1M tuples. The results of applying ++Spicy, SESM and EDEX on these data sources are shown in Figure 5-13(a). These experiments show that execution times scales up to 1M tuples. As shown in this figure, ++Spicy and SESM show almost

the same behaviour as the extra cost of processing schemas based on a simple conceptual model including 4 tables is negligible.

We were also interested to study the behaviour of EDEX to handle data sources with high depths of RATs. For this purpose, we formed five scenarios including 100k tuples with the same number of RATs, while we varied the average depths of RATs from 1 to 5. As shown in Figure 5-13(b), the result of experiments shows that the execution time of EDEX increases linearly with increasing the average depth of RATs. Similar to previous experiments, SESM follows the same trend as ++Spicy. Although the execution time of EDEX is less than the execution time of ++Spicy and SESM in all scenarios, the experiments show that increasing the average depth of RATs can be problematic for EDEX because of the increase in the number of super entities generated and processed.

We also tested EDEX over schemas with different average number of edges in RATs. For this purpose, we generated data sources with 100k tuples, while we varied the averages of edges from 2 to 10. As expected, increasing the number of edges in RATs directly results in increasing the number of super entities and consequently increasing the processing time. In spite of the negative effect of this parameter on the execution time of EDEX, this system still outperforms ++Spicy and SESM in all scenarios. Note that in real data exchange scenarios for data sources in Table 5-2, the maximum average depth was 1.75, while in these scalability evaluation experiments, the behaviour of EDEX is evaluated for data sources with average depths up to 5. Accordingly, the maximum of average number of edges in Table 5-1 is 2.37, while as show in Figure 5-13(c), EDEX can properly scale up to 10.

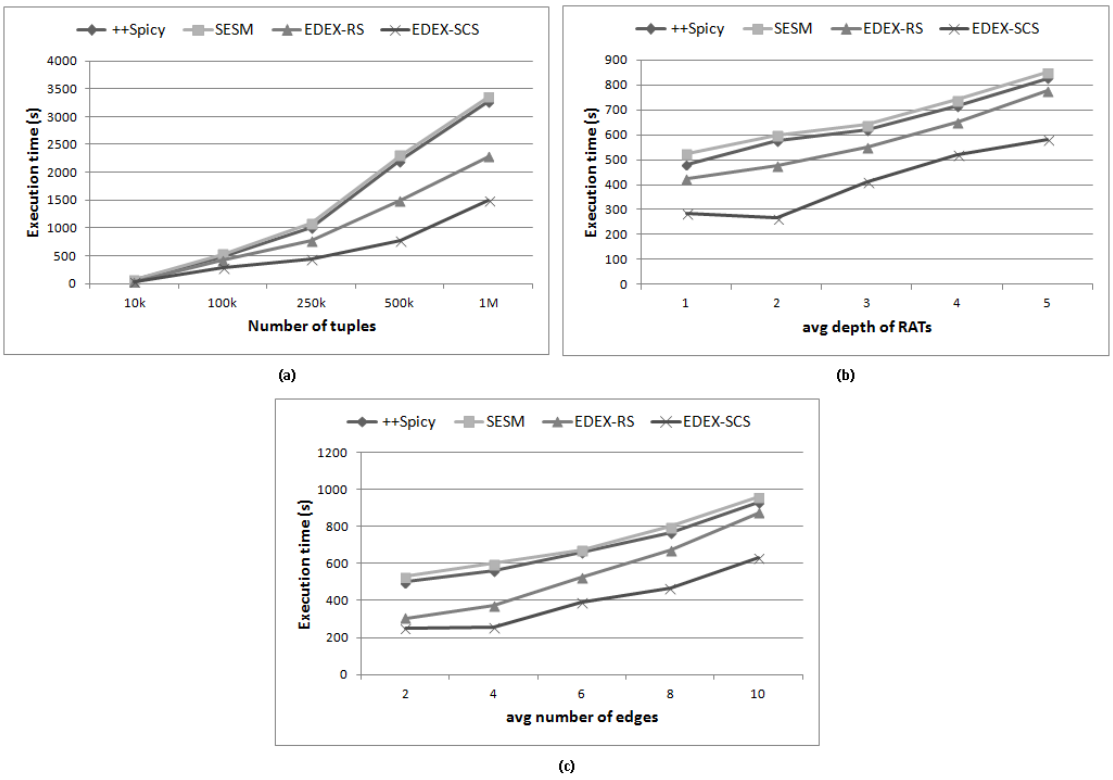


Figure 5-13: Execution times of EDEX on data sources of different size (a), different average depth of RATs (b) and average number of edges(c)

D. Storage Space

The ability to handle ambiguous scenarios, as well as improvement in the efficiency of EDEX in comparison to ++Spicy and SESM, is achievable with the cost of extra space required to store intermediate super entities. In this section, we measure how much extra space is required to perform EDEX in various data exchange scenarios and discuss techniques to mitigate the negative effects of this drawback. To reduce the extra space required to store super entities, we employed a "process-as-generate" technique in which instead of generating all super entities altogether and then processing them, super entities are processed as generated. In order to find if a super entity is included in other super entities, we only need to check super entities extracted from descendents of the source of

that super entity. As a result, super entity generation and entity residing in the target can be performed concurrently. Once a set of super entities is finalized (there is no need for redundancy check), these super entities are entered to the entity residing component.

In spite of the techniques used to reduce the space used by EDEX, this system requires extra space compared to ++Spicy and SESM. We measured space required to perform each data exchange scenario for all 12 scenarios in Table 5-2. The results of experiments for these scenarios using row store database (SQL Server 2012) and SCS (Sliced Column Store) are shown in Figure 5-14. In the experiments, the space required to operate EDEX on these scenarios is measured based on the maximum space allocated during performing each scenario. As shown in Figure 5-14, the version of EDEX using SCS allocates more space due to extra space required to store metadata regarding property tables. This is a tradeoff between better performance and extra space which is a decision made by a data exchange designer.

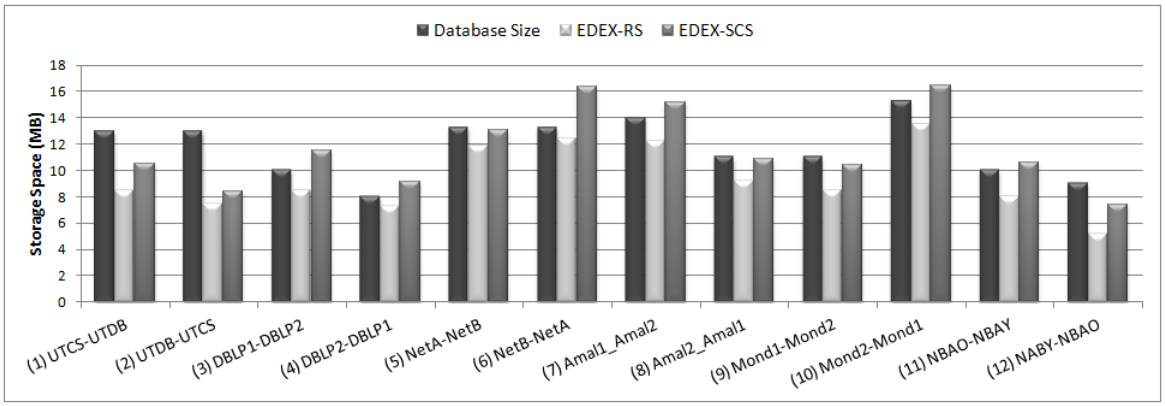


Figure 5-14: Storage space required to run EDEX on Scenarios in Table 5-2 using row-store (RS) and Sliced Column Store (SCS)

An important observation in these experiments is that the maximum space required to perform EDEX on each scenario is less than the size of the source instance in

the case of using EDEX with a row-store database. We used the data sources from scalability evaluation to study what features of data sources may affect the space required to perform EDEX. For this purpose, we performed three different set of experiments. As shown in Figure 5-15(a), first we varied the average number of tuples in the source instance for a constant number of RATs in the same simple schema. The results show there is a polynomial trend in space required to perform EDEX as the number of tuples grows. We already expected this trend as the number of super entities generated for each RAT is directly dependent on the number of tuples. In the second set of experiments, we varied the average depth of RATs, while the average numbers of tuples in each table as well as the average number of edges were kept constant. Not surprisingly, as shown in Figure 15(b), the space required to perform EDEX in all cases is almost the same because using "process as generate" technique avoids generating all super entities. Finally, we varied the average depth of RATs over a constant average number of tuples in each relation. As shown in Figure 5-15(c), space required to perform EDEX is increased with increasing the average depth of RATs. However, this is not considerable relative to the size of the dataset. To sum up, the experiments demonstrate that it is feasible to use EDEX on data sources of various sizes, with a reasonable extra space which would be as large as the size of the source dataset in the worst case.

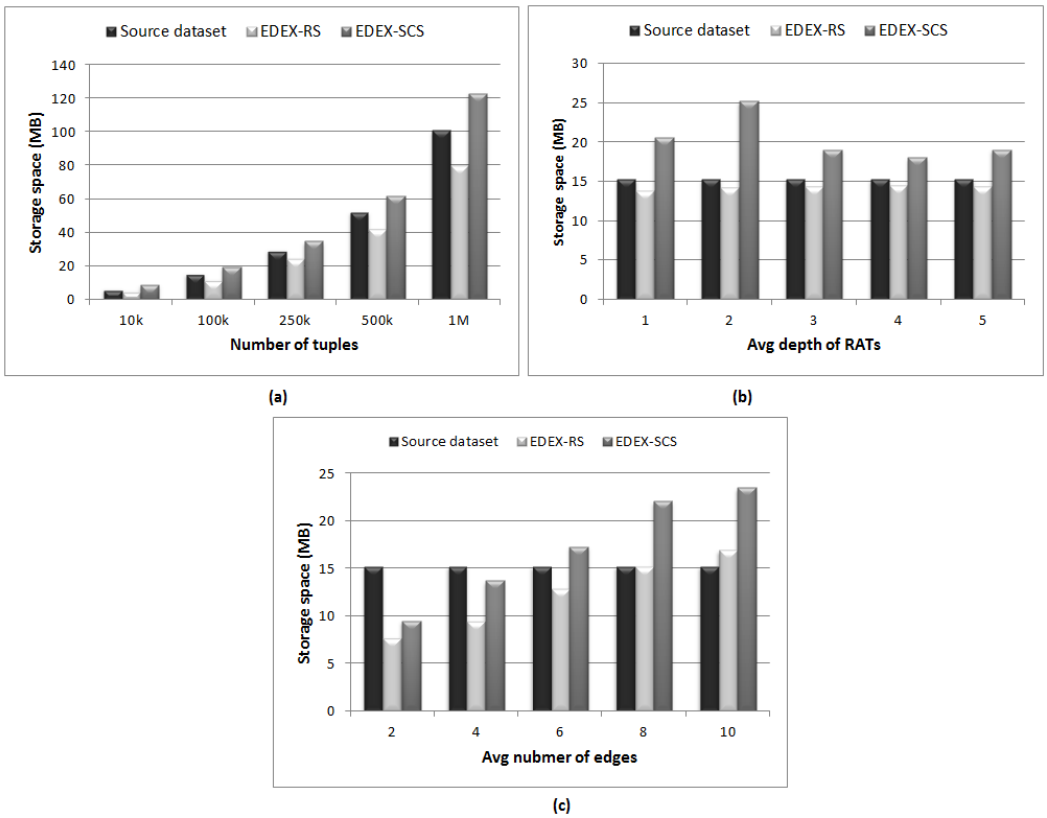


Figure 5-15: Storage space required to run EDEX on data sources of different size (a), different average depth of RATs (b) and different number of edges(c)

5.8 Related Work

The prevailing approach in data exchange has been based on schema mapping, in which schema mapping expressions are used to generate the target instance. Clio (Fagin et al., 2009; Miller et al., 2000) has pioneered this approach, and many subsequent research prototypes such as (Bonifati et al., 2010; Hernández et al., 2008) are proposed based on this project. Alongside studies on practical tools and algorithms for schema mapping generation, there have been theoretical studies on data exchange to provide a solid theoretical foundation for data exchange (Fagin et al., 2005a; Fagin et al., 2005b). The introduction and formalization of *universal* and *core* solution concepts has been the

key contributions of theoretical studies in data exchange. Generated by many schema mapping systems such as Clio (Fagin et al., 2009; Miller et al., 2000) and HePToX (Bonifati et al., 2005), universal solutions are preferred as they are the most general solution covering the entire space of valid solutions. On the other hand, generating the core solutions as a minimal universal solution is considered a natural requirement in data exchange. Two different algorithms (greedy and block) are proposed in (Fagin et al., 2005b) to compute the core solution. A polynomial algorithm providing a general answer to the problem of computing core solutions in data exchange is proposed in (Gottlob & Nash, 2008). An implementation of the core-computation algorithm using SQL is proposed in (Pichler & Savenkov, 2010).

Few data exchange systems based on schema mapping have considered producing executable scripts to compute the core solution. There have been two different approaches for generating the core solution. In the post-processing approach, the target solution generated by a system is pruned and processed to generate the core solution (Gottlob & Nash, 2008; Pichler & Savenkov, 2010). As argued in (Marnette et al., 2010), this technique may result in high redundancies that consequently impairs efficiency and quality of a data exchange system. On the other hand, in pre-processing approaches such as ++Spicy (Marnette et al., 2011), schema mapping expressions are rewritten such that refined mappings directly generate the core solution. The concept of homomorphism among mapping expressions is used to find mappings that may result in generating redundant tuples. In (ten Cate et al., 2009), schema mapping expressions are rewritten

into a laconic schema mapping (specified by first-order s-t tgds). Such laconic schema mappings can directly generate the core solution.

One important issue in data exchange based on schema mapping is ability to resolve ambiguous cases. As discussed in (Alexe et al., 2008), a schema mapping is ambiguous if it specifies in more than one way how an atomic target schema element can be created. Muse (Alexe et al., 2008) allows a mapping designer to select desired mapping among alternative interpretations of an ambiguous mapping.

As argued in (Qian et al., 2012), providing examples of target instance would be simpler for users rather than establishing matching between corresponding properties in source and target schemas. They proposed a sample-driven schema mapping technique that automatically constructs schema mappings from sample target instances provided by users. This technique generates mapping expressions given each pair of sample source and target instances. Then, the validity of these mappings is checked within the limit of acceptable noise. In EIRENE (Alexe et al., 2011), data examples are used to refine schema mappings rather than generating mapping expressions.

Another approach employed to gain the semantics of the data for ambiguity resolution is employing a domain ontology (or conceptual model) to represent higher level mappings between a source and a target schema. Sekhavat & Parsons (2013a) propose a technique in which schema mapping expressions are enhanced using conceptual models. The main drawback of this approach is the difficulty of designing a global domain ontology and conceptual model. In addition, maintaining mappings

between global and local domain ontologies is expensive when data sources are designed and maintained independently.

5.9 Conclusion and Future Work

In this chapter, we showed that schema mapping expressions representing relations between a set of classes in the source and the target are not able to handle many ambiguous cases in data exchange. We attributed this problem to the assumption of inherent classification in schema mapping. To address this problem, we proposed an entity preserving approach (EDEX) for data exchange in which the focus is on preserving source entities in the target no matter to what class they belong in the source. We introduced the concept of super entities to capture indirect properties of entities. We showed that, unlike many schema mapping based data exchange systems, EDEX can resolve ambiguous cases. In addition, EDEX can directly generate the core solution as a desirable solution for data exchange. There are interesting issues that remain open. Developing a mapping language expressing relations between source and target independent of classification in source and target is one of particular interest.

5.10 References

Alexe, B., Chiticariu, L., Miller, R. J., & Wang-Chiew Tan. (2008a). Muse: mapping understanding and design by example. *Proceedings of the IEEE 24rd International Conference on Data Engineering,* Cancún, México. 10-19. doi: 10.1109/ICDE.2008.4497409

Alexe, B., Tan, W., & Velegrakis, Y. (2008b). Stbenchmark: towards a benchmark for mapping systems. *Proceedings of the VLDB Endowment, 1*(1), 230-244.

Alexe, B., ten Cate, B., Kolaitis, P. G., & Tan, W. (2011). EIRENE: Interactive design and refinement of schema mappings via data examples. *Proceedings of the VLDB Endowment, 4*(12), 1414-1417.

Bellahsene, Z., Bonifati, A. & Rahm, E. (2011). *Schema Matching and Mapping*. Berlin, Heidelberg: Springer-Verlag.

Bonifati, A., Chang, E. Q., Lakshmanan, A. V. S., Ho, T., & Pottinger, R. (2005). HePToX: marrying XML and heterogeneity in your P2P databases. *Proceedings of the 31st International Conference on Very Large Data Bases,* Trondheim, Norway. 1267-1270.

Bonifati, A., Chang, E., Ho, T., Lakshmanan, L. V., Pottinger, R., & Chung, Y. (2010). Schema mapping and query translation in heterogeneous P2P XML databases. *The VLDB Journal, 19*(2), 231-256. doi: 10.1007/s00778-009-0159-9

Bonifati, A., Mecca, G., Papotti, P., & Velegrakis, Y. (2011). Discovery and correctness of schema mapping transformations. In Bellahsene, Z., Bonifati, A. & Rahm, E. (Ed.), *Schema Matching and Mapping* (pp. 111-147). Berlin, Heidelberg: Springer-Verlag.

Bunge, M. (1977). *Treatise on Basic Philosophy*: *the Furniture of the World*. Boston, MA: Reidel.

Casanova, M. A., Fagin, R., & Papadimitriou, C. H. (1982). Inclusion dependencies and their interaction with functional dependencies. *Proceedings of the 1st ACM SIGACT-SIGMOD Symposium on Principles of Database Systems,* Los Angeles, California. 171-176. doi: 10.1145/588111.588141

Chen, P. P. (1976). The entity-relationship model-Toward a unified view of data. *ACM Transactions on Database Systems, 1*(1), 9-36. doi: 10.1145/320434.320440

Das Sarma, A., Dong, X. L., & Halevy, A. Y. (2011). Uncertainty in data integration and dataspace support platforms. In Bellahsene, Z., Bonifati, A., & Rahm, E. (Ed.), *Schema Matching and Mapping* (pp. 75-108). Berlin, Heidelberg: Springer-Verlag.

Fagin, R., Kolaitis, P. G., Miller, R. J., & Popa, L. (2005a). Data exchange: semantics and query answering. *Theoretical Computer Science, 336*(1), 89-124. doi: 10.1016/j.tcs.2004.10.033

Fagin, R., Kolaitis, P. G., & Popa, L. (2005b). Data exchange: getting to the core. *ACM Transactions on Database Systems, 30*(1), 174-210. doi:10.1145/1061318.1061323

Fagin, R., Haas, L. M., Hernández, M., Miller, R. J., Popa, L., & Velegrakis, Y. (2009). Conceptual modeling: foundations and applications. In A. Borgida, er T., V. K. Chaudhri, P. Giorgini & E. S. Yu (Eds.), *Essays in Honor of John Mylopoulos* (pp. 198-236). Berlin, Heidelberg: Springer-Verlag. doi: 10.1007/978-3-642-02463-4_12

Gottlob, G., & Nash, A. (2008). Efficient core computation in data exchange. *Journal of the ACM, 55*(2), 9:1-9:49. doi: 10.1145/1346330.1346334

Haas, L. M., Hentschel, M., Kossmann, D., & Miller, R. J. (2009). Schema AND data: a holistic approach to mapping, resolution and fusion in information integration. *Proceedings of the 28th International Conference on Conceptual Modeling*, Gramado, Brazil. 27-40. doi: 10.1007/978-3-642-04840-1_5

Hernández, M. A., Papotti, P., & Tan, W. (2008). Data exchange with data-metadata translations. *Proceedings of the VLDB Endowment, 1*(1), 260-273.

Marnette, B., Mecca, G., & Papotti, P. (2010). Scalable data exchange with functional dependencies. *Proceedings of the VLDB Endowment, 3*(1-2), 105-116.

Marnette, B., Mecca, G., Papotti, P., Raunich, S., & Santoro, D. (2011). ++Spicy: an open-source tool for second-generation schema mapping and data exchange. *Proceedings of the VLDB Endowment, 4*(12), 1438-1441.

Mecca, G., Papotti, P., & Raunich, S. (2012). Core schema mappings: scalable core computations in data exchange. *Information Systems, 37*(7), 677-711. doi: 10.1016/j.is.2012.03.004

Miller, R. J., Haas, L. M., & Hernández, M. A. (2000). Schema mapping as query discovery. *Proceedings of the 26th International Conference on Very Large Data Bases*, Cairo, Egypt. 77-88.

Parsons, J., & Wand, Y. (2000). Emancipating instances from the tyranny of classes in information modeling. *ACM Transactions on Database Systems, 25*(2), 228-268. doi: 10.1145/357775.357778

Pichler, R., & Savenkov, V. (2010). Towards practical feasibility of core computation in data exchange. *Theoretical Computer Science, 411*(7-9), 935-957. doi: 10.1016/j.tcs.2009.09.035

Popa, L., & Tannen, V. (1999). An equational chase for path-conjunctive queries, constraints, and views. *Proceedings of the 7th International Conference on Database Theory*, Jerusalem, Israel. 39-57.

Popa, L., Velegrakis, Y., Hernández, M. A., Miller, R. J., & Fagin, R. (2002). Translating Web data. *Proceedings of the 28th International Conference on Very Large Data Bases,* Hong Kong, China. 598-609.

Qian, L., Cafarella, M. J., & Jagadish, H. V. (2012). Sample-driven schema mapping. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Scottsdale, AZ, USA. 73-84. doi: 10.1145/2213836.2213846

San, R. (2012). *Ventana research 2012 value index for data integration.* (Research). CA, USA: Ventana Research.

Sekhavat, Y. A., & Parsons, J. (2012). Sliced column-store (SCS): ontological foundations and practical implications. *Proceedings of the 31st International Conference on Conceptual Modeling,* Florence, Italy. 102-115. doi: 10.1007/978-3-642-34002-4_8

Sekhavat, Y. A., & Parsons, J. (2013a). SESM: semantic enrichment of schema mappings. *Proceedings of 4th ICDE International Workshop on Data Engineering Meets Semantic Web,* Brisbane, Australia (to appear).

Sekhavat, Y. A., & Parsons, J. (2013b). EDEX: Entity Preserving Data Exchange. *Proceedings of DATA'13 International Conference on Data Management Technologies,* Reykjavík, Iceland (to appear)

Stonebraker, M., Abadi, D. J., Batkin, A., Chen, X., Cherniack, M., Ferreira, M., Lau, E., Lin, A., Madden, S., O'Neil, E., O'Neil, P., Rasin, A., Tran, N., Zdonik, S. (2005). C-store: a column-oriented DBMS. *Proceedings of the 31st International Conference on Very Large Data Bases,* Trondheim, Norway. 553-564.

Talburt, J. R. (2011). *Entity Resolution and Information Quality,* Burlington, MA: Morgan Kaufmann.

ten Cate, B., Chiticariu, L., Kolaitis, P., & Tan, W. (2009). Laconic schema mappings: computing the core with SQL queries. *Proceedings of the VLDB Endowment*, 2(1), 1006-1017.

Chapter 6 Sliced Column-Store (SCS): Ontological Foundations and Practical Implications[1]

Abstract

Advances in business intelligence systems based on processing large data volumes are driving efforts toward read-optimized databases. Recently, the use of column-store approaches as a solution for such databases has become quite popular. The main idea behind the column-store approach is reducing I/O requirements through vertical partitioning of data in which only those attributes that are required to answer a query are read. This chapter offers two contributions to column-store data models. First, we show that such models can be grounded in ontological foundations that provide a theoretical basis for column-store databases based on representational adequacy. Second, we use these ontological foundations as the basis to propose an extended model of the column-store model called Sliced Column Store (SCS), and show that this model outperforms column-store models for read-oriented queries.

6.1 Introduction

The performance of query-intensive systems is strongly dependent on the performance of underlying databases and query processing engines. These systems require read-optimized database engines in which efficiently answering read-oriented ad-hoc queries has priority over write-oriented queries. Relational databases are optimized for Online Transaction Processing (OLTP) in which handling a large number of small inserts and

[1] Sekhavat, Y. A., & Parsons, J. (2012). Sliced column-store (SCS): Ontological foundations and practical implications. *Proceedings of ER'12 International Conference on Conceptual Modeling*, Florence, Italy. 102-115.

updates in an acceptable time is more important than reducing query answering time for complex read-oriented queries. These databases are implemented based on the row-store approach in which records (rows of tables) are contiguously stored in memory such that a single disk write is usually enough to write all fields of a record on a disk, and provides acceptable query performance in sequential access to the data. However, relational databases are not optimized for read-oriented tasks such as querying data warehouses. In terms of read-oriented queries, such architecture usually requires a full table scan where many data columns are projected out since not all attributes are required to answer a query. Consequently, many irrelevant properties are read, even though they are not necessary to process a query. As a result, relational systems are not I/O efficient because they use I/O bandwidth for reading unnecessary data. Based on this fact and the simple idea of reading only those attributes that are required to answer a query, the Column Store (CS) approach has been proposed. Unlike the row-store model in which the properties of a record are stored contiguously in memory, in the column-store approach, the values of each property for different records are stored contiguously. Recent years have witnessed the introduction of many database systems based on the column-store approach (Boncz et al., 2005; Larson et al., 2011; MacNicol & French, 2004; Stonebraker et al., 2005).

In addition to improving the performance of read-oriented queries, some research has also shown the potential of the column-store approach in addressing other problems in database management. Vertical table partitioning that uses the column store approach, is one of the major techniques used in database management to address difficulties in managing large database systems. As discussed in (Herodotou et al., 2011), table

partitioning can increase the manageability of database systems by allowing parallel access to different properties, easier backup, and a fine grained access control through providing facilities to enforce different access-right policies for different partitions. In addition to query efficiency and database manageability, the column-store approach provides other advantages such as supporting multi-value attributes, handling null values and efficient handling of wide tables with sparse data.

Based on these promising advantages of the column-store model, in this chapter we explore a theoretical foundation that provides a basis for the column-store approach. While prior research treats the model only as an ad-hoc approach used for physical data storage, we examine the question whether it is possible to provide a theoretical foundation that can further improve the performance of query processing. Answering this question is crucial since the answers can direct further efforts to improve data storage models based on the column-store approach.

We first discuss ontological foundations behind the column-store approach. We show that, unlike the row-store approach, the column-store approach is more compatible with a particular ontological view of the nature of reality represented in information systems. In particular, we show how the assumption of *inherent classification* (Parsons & Wand, 2000) has permeated the design of row-oriented model. We show the column-store model represents one step forward towards addressing this problem by vertical partitioning of data. Using these theoretical foundations, we show the column-store model can be improved by column slicing. We suggest two different query independent column slicing technique for nominal and string attributes. We argue that Sliced Column Store

(SCS), which is a step towards full data partitioning, is more compatible with the nature of data. The main advantage of column slicing techniques is that only those values specified in query predicates are read. We show how column slicing results in reducing the cost of selection operations and consequently speeding up join operations compared to the pure column store model.

The horizontal reorganization of data in column-store approach is already considered in terms of database cracking (Idreos et al., 2007; Idreos et al., 2011). This approach is a type of partial sorting in which each request for a particular result set (through posing queries) is an advice for partitioning columns to smaller parts. The rationale behind this technique is that future queries are somehow similar to previous queries that are already posed by users, and as a result, they are more likely to have similar query predicates. We argue that reorganizing physical storage of data based on input queries has some drawbacks. First, business intelligence applications usually involve many new ad-hoc queries that are not based on previous queries. For such queries, reorganization of columns may even worsen the query performance because of the extra effort required for reorganization of data, while future queries may not take advantages of this reorganization. Second, reorganizing before or during query answering not only requires extra time that negatively affects query answering, but also provides many concurrency and consistency problems that require additional consideration. Moreover, inserting new data requires reorganization of data in columns by shifting data. The proposed column slicing technique in this chapter is an effort to address these issues in which instead of sorting, the list of instances possessing the same value of a property

are physically stored in different slices. Our experiments show the effectiveness of using column slicing techniques to improve query performance as well as to address many problems of database cracking and column sorting.

6.2 SCS: Ontological Foundations and Practical Implications

Although reducing I/O overhead to improve query performance is considered as the main technical motivation behind column-oriented databases, to our knowledge no theoretical foundation has been proposed for this model. In this section, we propose a theoretical foundation behind the column store model and show why this model is appropriate for processing read-oriented queries. We also aim to find whether it is possible to improve query performance by refining the column store model by adopting a suitable theoretical foundation.

6.2.1 From Row Store to Column Store

Ontological Foundations. To explore the theoretical foundations behind the column store approach, we turn to formal ontology, the branch of philosophy that deals with the order and structure of reality in the broadest way possible. Ontology has been widely used as a theoretical foundation for conceptual modeling, both in theoretical analyses (Parsons & Wand, 2000) and in empirical studies (Gemino & Wand, 2004). The rationale behind using ontological principles is that it provides a meta-model of existing things in the real world, and database systems represent knowledge or facts about the real world. As a result, understanding the actual components and relations between things in the real world helps in designing databases that better reflect this reality. More specifically, we show

how the row-oriented approach has been the consequence of the *assumption of inherent classification* in information system modeling (Parsons & Wand, 2000). Many difficulties in information system management, such as schema integration, schema evolution and ability to exchange information between heterogeneous data sources, can be attributed to this assumption. According to the assumption of inherent classification, everything that is modeled in a domain of interest in an information system is treated as an instance of a class in an object-oriented model (or an entity belonging to an entity type in the Entity Relationship model). Contrary to this assumption, although classification is one of the rudimentary abilities of humans in understanding the things of interest in any domain, real world objects do not inherently belong to classes; rather, classification is a consequence of an effort to organize knowledge about existing things. Inherent classification is incompatible with ontological assumptions about the nature of things in the real world where things and their properties exist prior to and independent of their classification (Parsons & Wand, 2000).

From this perspective, the row-store model can be viewed as a consequence of the assumption of inherent classification that pervades database design. An implementation consequence of this assumption is that information about instances that belong to the same class is stored contiguously in memory. However, based on ontological foundations, data is not inherently classified; rather, classification is an outcome of humans' efforts to organize information. Thus, there is no fundamental reason why instances belonging to the same class should be stored contiguously in memory. The row-store model (i.e., based on contiguously storing records that are in the same class) adds complexity to data

extraction that consequently affects query answering time. This complexity is the result of extra operations required to filter data based on query predicates. We argue that query answering in a database is nothing but a set of data partitioning and data combining operations based on query predicates. For example, in the relational model, data partitioning is provided through selection, projection, intersection, difference, and division operations. On the other hand, combinations are provided through operations such as join, union, and Cartesian product. In the row-store model, the properties of instances belonging to the same class (tuples in a relation) are stored contiguously in memory. However, to answer many queries, there is no need for simultaneously extracting these properties. Therefore, extra selections and projections are required to answer these queries.

To address this problem of relational databases, an Instance-Based Data Model (IBDM) is proposed in (Parsons & Wand, 2000) that separates the definition of instances (records) from the definition of classes. IBDM has a two-layer architecture, in which the first (instance) layer specifies the existing things in a domain by defining their properties or attributes independent of their membership in a class. The second (class) layer specifies which properties belong to each class, where one property may belong to more than one class. From an implementation point of view, instances are represented through unique identifiers plus pointers to intrinsic properties (those depending on only one instance, such as ‘gender’ of a patient), and mutual properties (those depending on two or more instances, such as ‘date of surgery’ depending on a patient and a doctor). In particular, intrinsic properties can be stored as a set of (InstanceID, Value) pairs, and

mutual properties as a set of (InstanceId1, InstanceId2, Value) triples. In this model, each intrinsic property can be represented as a binary table where the name of the table is the unique identifier of that property. The first column of this table is the unique identifier of the instances (InstanceID) that possess this property, and the second column is the value of that property for each instance (value). Accordingly, a mutual property can be represented through a three-column table in which the first and second columns are unique identifiers of the instances that jointly possess this mutual property, and the third column is the value of that mutual property. From this description, IBDM can be viewed as a column-store model originally proposed as a solution for the problem of inherent classification.

Practical Implications. As argued in (Abadi et al., 2008), "There is in fact something fundamental about the design of column-store systems that makes them better suited to data warehousing workloads." From the implementation point of view, relational databases are based on the row-store approach in which data is stored in two dimensional relations (tables) including a set of properties (columns) for each record of data (row). Since data rows are stored contiguously in memory, this approach wastes I/O bandwidth as all attributes of a table are read even if not all of them are required to answer a query. The column-store approach is proposed as a solution to address this problem by limiting the number of attributes to those required to answer a given query. As shown in Figure 6-1, regarding a query that only address property A and property B in its query predicate, the whole table is scanned for the row-store model while columns C and D are not scanned in the column-store model. As discussed in (Harizopoulos et al., 2006) reducing

the amount of data read has a significant effect on reducing the time required to execute queries where I/O is the main bottleneck.

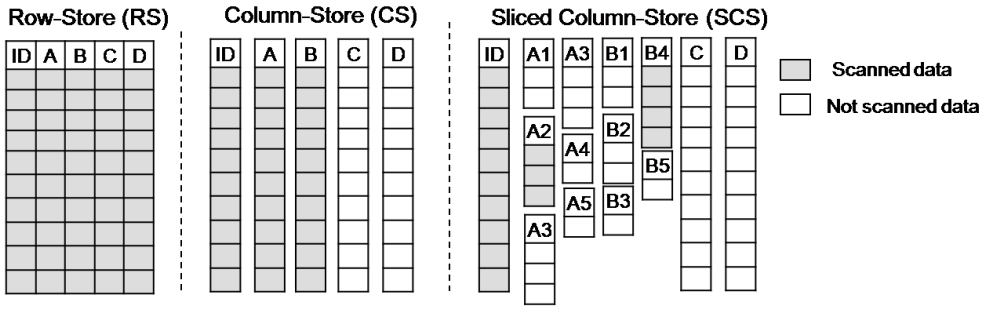


Figure 6-1: Scanned data in row-store, column-store and sliced column-store for a query including particular values of A and B

In the architecture of column-store databases, every n-ary relation is represented by a group of binary relations that are stored in form of two-column tables. The tuples of each binary relation are stored physically adjacent to speed up scanning data in these tables. Recently, there has been considerable effort devoted to improving the column-store based database systems (Boncz et al., 2005; Larson et al., 2011; MacNicol & French, 2004; Stonebraker et al., 2005). C-store (Stonebraker et al., 2005) is one of the leading column-store projects in which data is stored in groups of correlated columns called projections. In this architecture, the same property may appear in different projections with different sorting orders. Each projection is stored in a different physical storage structure. In this model, relations between different properties of the same record (data row) are provided through implementing join indexes. These join indexes are necessary to reconstruct the original table from existing projections.

6.2.2 From Column Store to Sliced Column Store (SCS)

Ontological Foundations. From the ontological point of view, characteristics of existing things in the real world are represented by 'properties,' which are the basic constructs in data models (Wand et al., 1999). To study the relations between properties, we use the property precedence notion of Bunge's ontology (Bunge, 1977). According to this notion, property P_1 precedes property P_2 if and only if the set of things possessing P_2 is a subset of things possessing P_1. For example, the property of 'having color' precedes the property of 'having red color' since the set of instances that are red is a subset of the set of instances that have color. We focus on property precedence since it has special importance in the context of classification. A class in a data model is represented in terms of set of generic properties while instances of that class possess specific properties implying those generic properties (Parsons & Wand, 2003). In other words, possessing a specific property manifests possessing a generic property. For example, 'gender=male' and 'gender=female' are two specific properties of a generic property 'having gender'.

According to Bunge's ontology [13], things and their properties exist prior to and independent of any classification. However, in the relational model (and consequently the row-store model based on it), an instance should be member of a class before insertion to the database. In this thesis, we distinguish between the classification based on generic properties (that constitutes the definition of classes in a data model) and classification based on specific properties (different manifestations of a generic property). Classification of instances based on generic properties prior to existence of things and their properties results in the problem of inherent classification. However, classification

of instances based on specific properties after insertion to the database is not in contradiction with ontological foundations. In addition, this type of classification is more compatible with the nature of read-oriented queries. Query answering in read-oriented queries is nothing but selecting instances from a database based on specific properties (manifestations). As a result, if these different manifestations are classified and stored contiguously on the memory, a query processor needs to read only those manifestations that are indicated in query predicates. The notion of classification based on manifestations constitutes the main idea behind the sliced column-store model (SCS). Although the column-store approach has addressed the problem of inherent classification by reading only the relevant properties, the query processor still needs to read all values of these relevant properties even if only particular values (that are indicated in query predicates) are required to answer a query.

In particular, the row-store model is the consequence of the classification of instances based on a set of generic properties where each set constitutes a class. This model prescribes instances that belong to the same class should be stored contiguously in memory. In the CS model, all values of a specific property are stored contiguously for the instances that possess that property. In SCS, classification in the manifestation layer propagates to the storage where identifications of instances possessing a specific manifestation are stored contiguously on the memory.

Practical Implementation. Sliced Column Store (SCS) model is an extended version of the column-store model that exploits the advantages of this model while providing better query performance by narrowing the search space in query processing. Column-store

techniques reduce the amount of data read by considering only those attributes that are necessary to answer a query. SCS goes beyond this idea, and not only ignores irrelevant attributes, but also reads only those particular values that are specified in query predicates (selection criteria). This is achieved through slicing property tables based on different values. For example, a gender column that stores gender of people in a column in the column store model is sliced and stored in two columns *gender_male* and *gender_female* such that each slice stores the ID of instances that possess each particular property value. In SCS, the value of properties is implicit in the name of properties. As shown in Figure 6-1, if the CS model is an effort to reduce I/O by vertically narrowing search space, SCS narrows this space by further partitioning of the search space. Note that unlike the common horizontal data partitioning in which the main table is horizontally partitioned (all partitions have the same data schema), in SCS partitioning is performed on binary tables that are already partitioned vertically according to the column-store model.

In spite of the advantages discussed earlier, the column store model has some limitations. The main problem is the cost of materialization (Abadi et al., 2007). Column store databases store data in a set of binary tables, while users request data in the form of row-style tuples that requires merging existing column (i.e., called materialization). Materialization is an important issue in column-store databases since it directly affects query processing. Two common techniques in materialization are Early Materialization (EM), that involves forming intermediate tuples from set of columns as they are accessed, and Late Materialization, in which intermediate tuples are not formed until after some part of the query is performed based on query predicates. For example, for a query over

three columns *A*, *B* and *C* with selection operations $\sigma 1$, $\sigma 2$ and $\sigma 3$, EM technique reads a block of *A*, *B* and *C*, and creates a row-style tuple (*A*, *B*, *C*). Then, it applies selection operations on these tuples. However, in LM, first the query processor reads tuples satisfying $\sigma 1$ from *A*, then reads *B* and *C* based on $\sigma 2$ and $\sigma 3$ respectively. Finally, these items are stitched together. According to (Agrawal et al., 2004), the total cost of joins among many small partitions with few properties is much less than a join between two tables with many properties. Although simple joins are required because joins are between two-column tables, the cost of these join operations are not negligible. As a result, any step towards reducing the cost of materialization in column store databases can have an important effect on the overall query answering performance. The column slicing technique diminishes the cost of materialization by reducing the cost of joins. This is achieved by eliminating the cost of selection operations in joins since IDs of instances possessing the same value of a particular attribute are stored in the same slice.

6.3 Column Slicing Techniques

The main contribution of this chapter is the basic idea of *column slicing* that can be implemented with any appropriate column slicing techniques. To implement this idea, in this thesis we propose two different slicing techniques for nominal and string data types, and we defer slicing techniques for numerical and other data types for future work. In the following, the details of these partitioning techniques are elaborated. An example of column slicing on a sample column store database is shown in Figure 6-2.

In the case of categorical properties, in accordance with the concept of classification based on manifestations, each binary table in the column-store model for a

nominal property (with domain of *n* possible values) is split to *n* single-column slices where each slice stores the unique identifiers of instances possessing a particular value of that property. The name of each slice implicitly shows the value of that property for those instances that are stored in that slice (i.e., property_value). Unlike the column store approach in which the values of properties are stored as data, in the column slicing approach, values of properties are considered as a metadata (i.e., the name of slice) representing the class of objects that possess a particular value of a property. For example, as shown in Figure 6-2 the nominal property *A* with the domain of values *a1*, *a2*, *a3*, *a4* and *a5* is split to five single-column slices *A_a1*, *A_a2*, *A_a3*, *A_a4*, *A_a5* that stores ID of instances possessing each particular value of *A*.

Compression, which is used widely in column store databases to reduce the size of databases and speeding up the query answering (Lemke et al., 2010; Stonebraker et al., 2005), is implicit inside the column slicing approach for nominal properties since each particular value of a property is stored one time as a metadata in the name of the single-column slice, and only ID of instances possessing this particular value are stored in that slice.

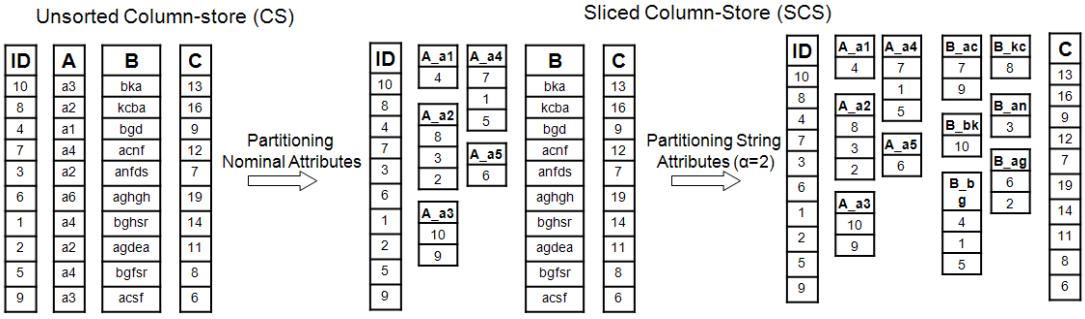


Figure 6-2: An example of partitioning technique for nominal (A) and string (B) attributes

In the case of string properties, we propose an α-level slicing technique based on the concept of *trie* data structure (Heinz et al., 2002). A trie is a tree-based data structure that is useful for handling strings over alphabet. In the trie data structure, information about the content of each node is stored in the path from the root to the node, rather than the node itself. We use this notion to classify the values of string properties based on their characters starting from the first character. More specifically, the first level includes 26 classes (a-z) where each class represents a class of strings starting with a particular English letter. In the case of lower- and upper-case letters, the number of classes will be 52 (a..z, A…Z) and in the case of all ASCII characters, the total number of classes in the first level is 128). At each level, nodes are further classified based on the next character. An example of this classification for data of Figure 6-2 is shown in Figure 6-3.

A path from the root to each leaf node (i.e., an order of characters) represents the name of the slice that stores the IDs of instances possessing string values starting with that path. Note that the concept of the trie data structure is used only for the purpose of explaining the slicing and classifying string properties, and there is no need to store this structure in the database since the name of each slice implicitly indicates the value of string properties starting with that name. For example, *B_be* represents a slice of property *B* that stores instances that their value for *B* starts with *be*. Since not all slices are created in the beginning, and slices are created in the record insertion time, there is no slice without a value. That way, we control the number of slices by avoiding empty slices. However, the number of slices can growth exponentially with increasing α. The effect of

applying different values for α on query answering and database size is discussed in the evaluation section.

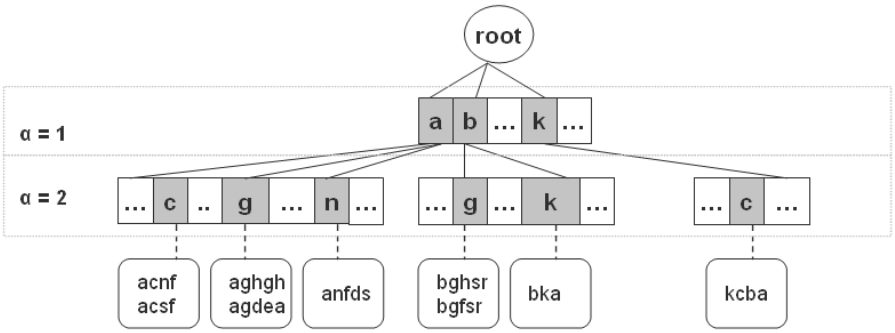


Figure 6-3: An example of slicing a string property (property B in Figure 6-2)

6.4 Column Slicing vs. Database Cracking, Sorting and Bitmap Index

Column-stores are heavily optimized to perform materialization as tuple reconstruction is the main cost of column-store query plans. Having unaligned columns is always something to avoid in order to reduce random accesses during tuple reconstruction. This is why C-store (Stonebraker et al., 2005) uses column-store projections where data is replicated, sorted and then compressed. Similarly, database cracking (Idreos et al., 2007) replicates columns in multiple orders but performs the sorting partially (i.e., only for the range of values referenced in query predicates). In this section we discuss how these techniques are different from column slicing.

Database cracking (Idreos et al., 2007; Idreos et al., 2011) is a dynamic partitioning technique for numerical properties such that the physical organization of data is continuously updated based on input queries. More specifically, based on the query predicate of input queries, those values of a numerical column that satisfy the query predicate are partially sorted. The main idea behind dynamic reorganization of data based

on input queries is that the way that users request data in the future is similar to the way they requested data in the past. We argue that database cracking has three main drawbacks. First, Business Intelligence applications usually deal with many new ad-hoc queries in which the behavior of users in query building is not necessarily the function of their behavior in the past. As a result, dynamic reorganization can worsen query performance of queries in which there is no overlap between the query predicates of past and future queries. However, the proposed column slicing technique in this thesis is a query independent technique in which the same manifestations of a property are classified and stored contiguously on the memory. Such a query independent technique is more appropriate to answer new ad-hoc queries. Second, the cost of reorganization of data in database cracking during query processing is an additional overhead that is added to the cost of query answering time. However, in the column slicing technique, physical reorganization and query processing are two independent processes performed in two different phases, and physical reorganization of data does not negatively affect query processing time. Third, in database cracking, inserting new data to the database may result in full reorganization of data since it will require shifting and physical reorganization of data. This would not be a problem for our column slicing technique as new records are stored in predefined slices that there is no need for reorganization of data.

Column sorting (in which binary tables are sorted) is a more general solution compared to database cracking (i.e., partially sorting column based on query predicates). Although column sorting can significantly reduce the cost of selection operations and consequently the cost of record materialization by reducing the cost of join operations, we

argue that column slicing works better than column sorting. Assume a binary table including n rows of (ID, value) for a property. To answer a query including a selection operation on a specific property, a binary search with the cost of O(log n) is required to select data from a sorted column. However, in the case of full data partitioning, the list of instances satisfying the selection operation already exist in a single-column slice where the name of each slice implicitly indicates its contents.

The Sliced Column Store (SCS) technique proposed in thesis is also different from the bitmap index technique (Chan et al., 1998). In the bitmap index, the unique identifier of an instance possessing a specific value is an index key while a boolean value (0 or 1) for each distinct value of that property is stored. For example, the gender property is stored as:

Instance id	male	Female
I1	0	1
I2	0	0
I3	1	0
I4	1	0

This structure is implemented using bit arrays (commonly called bitmaps) that supports performing query answering using fast bitwise logical operations on these bitmaps. However, in SCS, we do not store boolean property values for each instance. Instead, we only store unique identifiers of instances possessing a specific value of a property to reduce time required to select instances based on specific values. While SCS has some similar benefits to bitmap indexes, such as reducing the time it takes to filter on a column with a small number of distinct values, the underlying structure is different.

6.5 Experiments

In this Section, we evaluate the effect of applying the column slicing technique on query answering; as well, we explore the side effects of this technique. SQL Server 2012 (Larson et al., 2011) is used as a database engine. The new added feature in this database engine (called column store index) is a pure column-store system in which data for different columns are stored on separate pages. As a result, performing the column index on a column ensures physically creating a column store database. The hardware setting of the employed database server is AMD Athlon, 64 X2 Dual Core 2.71 GH, 1GB RAM, and 280 GB HD. To gain an understanding of column slicing techniques, we conducted experiments in two phases with two different datasets. In the first phase, a dataset generated by our simple data generator (MUN-DGen) is used to explore the pure effect of slicing on query operations. In the second phase, TPC-H benchmark (www.tcp.org) with data that typically found in data warehousing and business intelligence applications is used.

6.5.1 Phase1: The Pure Effect of Slicing on Query Operations

In the first phase, we have conducted a set of experiments on a simple database. The purpose of using this simple database for exploring the pure effect of slicing is controlling over the number of tables, number of properties, types of the properties and number of categorical values. We implemented a data generator (MUN-Dgen) that automatically generates column store and sliced column store datasets for a big table. This big table *A* includes *m* sting properties *A_SP* , *n* integer properties *A_IP*, and *k* categorical

(represented by a integer data type.) properties *A_CP*. In the case of categorical properties in SCS, $A\_CP_{i\_j}$ is the value *j* of the categorical property A_CP_i.

Simple Selections. In the case of column slicing, we expect better performance compared to the pure column store technique for simple selection queries with high selectivity query predicates because they access only few records compared to the whole column scanning in column store. To explore this effect, we take into account a query $Q_S = \pi_{InId}(\sigma_{A\_CP1='CP1\text{-}1'}(A))$ that selects instances from a single table based the value of a categorical property (A_CP_1 is the referenced categorical property, and $CP_{1\text{-}1}$ is a specific value of this property). Let *r* the number of records in table *A* and *s* the average size of allocated space by a nominal property (that can be shown by an integer or a string data type). The amount of data scanned in CS is *rs* this amount is *rs/t* for SCS where *t* is the average number of categorical values. In SCS, the selection criterion is implicit in the name of table, and there is no need for explicit filtering of data. The experiment supporting this idea is shown in Figure 6-4. With increasing the number of categorical values, the execution time of Q_S on SCS is reduced because of increasing the selectivity of the query predicate.

Aggregation Functions. To explore the effect of slicing on queries including aggregate functions, we consider the query $Q_A = \pi_{A\_CP1,Count(A\text{-}ID)}(A)$ that adds an aggregation operator on top of the selection query. Since the list of instances possessing each specific value of a categorical property is stored in different slices, aggregate functions access those data directly. However, since all records must be scanned in both CS and SCS, there

is no considerable improvement on aggregate functions running on SCS. The experiments regarding Q_A are shown in Figure 6-4.

Join Operations. We expect a considerable improvement in performance of queries including join operations because of reducing the cost of materialization (reconstructing row-style tuples from a set of columns) in SCS. Reduction in the cost of the materialization is the consequence of reducing the cost of selections operations on columns since the list of instances are pre-categorized in a set of column slices. In the case of joining multiple columns, all relevant columns (including columns that will be materialized and columns that are referenced in the query predicates) participate in the join operation. Since only pre-selected values of columns are entered into join operations, we expect considerable improvement for SCS. Recall that according to (Agrawal et al., 2004), the join between two tables with *m>2* properties is more expensive than joins between *m* tables with two properties. To study the effect of slicing technique on the join operations, we used the query Q_J: $\pi_{InId,\ A\_CP1,A\_SP1,A\_1P1}\ [\sigma_{A\_CP2=\ 'CP2\text{-}1'}\ (A)]$. As shown in Figure 6-4, as the number of categorical values increases, the execution time of Q_J decreases**.**

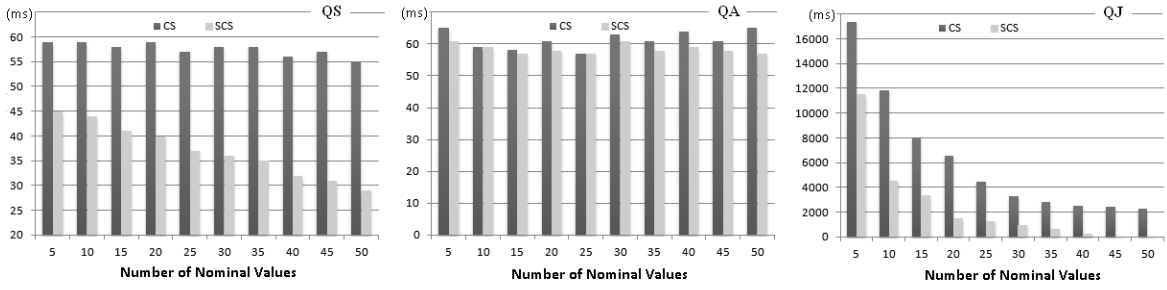


Figure 6-4: Execution time of QS, QA and QJ for different number of nominal values

6.5.2 Phase2: TPC-H Benchmark

In the second phase of experiments, in order to evaluate the effect of column slicing on real business intelligence and data warehousing dataset, an instance of TPC-H at scale 1 is generated using the TPC-H data generator where the total database size is around 1GB. The database generator program in TPC-H was modified such that creates equivalent databases in three different states: 1) slicing only nominal attributes, 2) slicing only string attributes, and 3) slicing both nominal and string attributes. This categorization allows gaining an understanding about both pure and mutual effects of different column slicing techniques on the performance of query processing.

Among 22 Queries in TPC-H benchmark (Http://www.tcp.org), queries Q3, Q10, Q12 were selected to explore the pure effect of partitioning nominal attributes because at least one nominal attribute exists in their predicates while there are no string attributes. These queries were performed on an instance of the TPC-H dataset in which only columns regarding nominal attributes were sliced. The query answering time is compared with running these queries on original instances of the generated database (Table 6-1).

To explore the effect of slicing string properties, Queries Q2, Q5, Q7, Q11, Q17 and Q20 were selected from TPC-H benchmark queries as they include at least one string property in their query predicates with no nominal property. Three different instances of TPC-H dataset (regarding $\alpha = 1, 2, 3$) in which all string properties are sliced were created. The results of these experiments regarding these queries are shown in Table 6-1. We do not suggest more than three levels of slicing for TPC-H dataset as exponentially increasing the number of slices negatively affects the advantages of column slicing. The

effect of increasing slices and consequently increasing the database size is discussed in Section 5.3.

Among TPC-H queries, Q8, Q19 and Q21 are queries that both nominal and string attributes are addressed in their query predicates. To explore the effect of slicing nominal properties in conjunction with string properties, instances of TPC-H dataset are generated in which all nominal and string properties were sliced. As shown in Table 6-1, our experiments demonstrate that the proposed techniques for slicing nominal and string properties can increase the performance of read-oriented queries.

Table 6-1: Execution time of TPC-H queries on CS and SCS (ms)

Category	**Nominal -No string**			**String-No Nominal**						**Nominal and string**		
Query	**Q3**	**Q10**	**Q12**	**Q2**	**Q5**	**Q7**	**Q11**	**Q17**	**Q20**	**Q8**	**Q19**	**Q21**
CS	1086	6103	1573	196	2376	1116	246	4090	1186	876	1913	2716
SCS (α=1)	724	2878	953	109	1033	791	161	2763	878	515	865	1364
SCS (α=2)	-	-	-	91	865	721	124	2235	687	406	723	1148
SCS (α=3)	-	-	-	85	791	702	113	2103	654	389	631	1007

6.5.3 The Side Effects of Column Slicing Approach

Increasing the number of slices by increasing the diversity of property values is the main side effect of column slicing approach. In the case of string attributes, at most *128^n* slices in level *n* can be created. One obvious consequence of applying full column slicing on a property with a large domain of values would be increasing a large number of slices while each slice stores only a few items. The result of increasing the number of slices will be increasing the metadata required to store these slices. With increasing the number of slices, we may reach a point where the advantage of slicing is offset by the extra cost to store and manage metadata regarding large number of slices. In other words, column slicing in SCS is acceptable only to the extent that the amount of growth for database is

not a big issue in an application. As shown in Table 6-1, there is only slight improvement in performance of SCS(α=2) to SCS(α=3) that shows the negative effect of increasing the number of slices. Unlike column sorting and column cracking technique in which data insertion is expensive and requires reorganization of physical storage, insertion is not problematic in SCS as identification of new instances possessing a particular value of a property must be saved in one of the fixed and predefined slices.

6.6 Conclusion and Future Work

In this chapter, we discussed the ontological foundations behind the column-store data model and we showed that how this model can be a solution for the problem of inherent classification in the row-store model. Unlike the row-store model, which prescribes contiguously storing properties of instances that are in the same class, in the column store model the problem of inherent classification is alleviated by allocating a separate property table for each property and contiguously storing all values of each property. In this thesis, we go beyond this approach and contended that from ontological point of view, there is no rationale behind storing all values of a property in the same column where extra selection operations are required to access a particular value of that property. We proposed the Sliced Column Store model in which property columns are horizontally partitioned to some slices, where each slice stores the identification of instances that possess a particular value of that property. We suggested a full slicing technique for nominal properties, and the α-level slicing technique for string properties. We conducted a set of experiments to study the effect of column slicing on the performance of query processing. The pure effect of each partitioning technique and the mutual effect of them

were explored using both a simple dataset (generated by MUN-DGEN data generator) and the TPC-H benchmark. Unlike the column-store approach in which the whole property table is scanned to find instances possessing a specific value, the major consequence of column slicing in SCS is eliminating this effort by directly reading only those instances that possess this property. We argued that this virtue can result in significant improvement in query processing from two points of view. First, improving query performance by reducing the cost of selection operations through reducing I/O amount; this is achieved by reading only those property values that are indicated in query predicates since instances possessing particular values are stored in separate slices. Second, by reducing the cost of join operations through decreasing the cost of selection operations, resulting in simple joins (i.e., joins between columns having smaller records). These improvements are achievable with a slight cost of increasing the size of database.

In this chapter, we sketched a research landscape with a large number of opportunities that slicing can contribute to database design, and we suggested two slicing techniques for string and nominal properties. For future work, we focus on efficient slicing techniques for other data types. We also aim to implement an SQL-independent prototype of SCS to study the scalability this method to handle big data.

6.7 References

Abadi, D., Myers, D. S., DeWitt, D. J., & Madden, S. R. (2007). Materialization strategies in a column-oriented DBMS. *Proceedings of the IEEE 23rd International Conference on Data Engineering,* Istanbul, Turkey. 466-475. doi: 10.1109/ICDE.2007.367892

Abadi, D. J., Madden, S. R., & Hachem, N. (2008). Column-stores vs. row-stores: how different are they really? *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Vancouver, Canada. 967-980. doi: 10.1145/1376616.1376712

Agrawal, S., Narasayya, V., & Yang, B. (2004). Integrating vertical and horizontal partitioning into automated physical database design. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Paris, France. 359-370. doi: 10.1145/1007568.1007609

Boncz, P. A., Zukowski, M., & Nes, N. (2005). MonetDB/X100: hyper-pipelining query execution. *Proceedings of the Biennial Conference on Innovative Data Systems Research,* Asilomar, CA, USA. 225-237.

Bunge, M. (1977). *Treatise on Basic Philosophy*: *the Furniture of the World*. Boston, MA: Reidel.

Chan, C., Ioannidis, Y. E. (1998). Bitmap index design and evaluation. Proceedings of the 1998 ACM SIGMOD international conference on Management of data. Seattle, Washington, USA, 355-366. doi: 10.1145/276304.276336

Gemino, A., & Wand, Y. (2004). A framework for empirical evaluation of conceptual modeling techniques. *Requirements Engineering, 9*(4), 248-260. doi: 10.1007/s00766-004-0204-6

Harizopoulos, S., Liang, V., Abadi, D. J., & Madden, S. (2006). Performance tradeoffs in read-optimized databases. *Proceedings of the 32nd International Conference on Very Large Data Bases,* Seoul, Korea. 487-498.

Heinz, S., Zobel, J., & Williams, H. E. (2002). Burst tries: a fast, efficient data structure for string keys. *ACM Transactions on Information Systems, 20*(2), 192-223. doi: 10.1145/506309.506312

Herodotou, H., Borisov, N., & Babu, S. (2011). Query optimization techniques for partitioned tables. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Athens, Greece. 49-60. doi: 10.1145/1989323.1989330

Idreos, S., Kersten, M. L., & Manegold, S. (2007). Database cracking. *Proceedings of the Biennial Conference on Innovative Data Systems Research,* Asilomar, CA, USA. 342-350.

Idreos, S., Manegold, S., Kuno, H., & Graefe, G. (2011). Merging what's cracked, cracking what's merged: adaptive indexing in main-memory column-stores. *Proceedings of the VLDB Endowment, 4*(9), 586-597.

Larson, P., Clinciu, C., Hanson, E. N., Oks, A., Price, S. L., Rangarajan, S., Surna, A., & Zhou, Q. (2011). SQL server column store indexes. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Athens, Greece. 1177-1184. doi: 10.1145/1989323.1989448

Lemke, C., Sattler, K., Faerber, F., & Zeier, A. (2010). Speeding up queries in column stores: a case for compression. *Proceedings of the 12th International Conference on Data Warehousing and Knowledge Discovery,* Bilbao, Spain. 117-129.

MacNicol, R., & French, B. (2004). Sybase IQ multiplex - designed for analytics. *Proceedings of the VLDB Endowment*, *30*(1), 1227-1230.

Parsons, J., & Wand, Y. (2000). Emancipating instances from the tyranny of classes in information modeling. *ACM Transactions on Database Systems, 25*(2), 228-268. doi: 10.1145/357775.357778

Parsons, J., & Wand, Y. (2003). Attribute-based semantic reconciliation of multiple data sources. *Journal on Data Semantics*, *2800*(1), 21-47. doi: 10.1007/978-3-540-39733-5_2

Stonebraker, M., Abadi, D. J., Batkin, A., Chen, X., Cherniack, M., Ferreira, M., Lau, E., Lin, A., Madden, S., O'Neil, E., O'Neil, P., Rasin, A., Tran, N., Zdonik, S. (2005). C-store: a column-oriented DBMS. *Proceedings of the 31st International Conference on Very Large Data Bases,* Trondheim, Norway. 553-564.

TPC-H benchmark specification. Retrieved from Http://www.tcp.org

Wand, Y., Storey, V. C., & Weber, R. (1999). An ontological analysis of the relationship construct in conceptual modeling. *ACM Transactions on Database Systems*, 24(4), 494-528. doi: 10.1145/331983.331989

Chapter 7 Summary

7.1 Overview of the problems and contributions

In this thesis, I exploited two fundamental notions of Bunge's ontology (*property precedence* and *composites*) as well as the concept of *assumption of inherent classification* developed in (Parsons & Wand, 2000) to provide theoretical foundations for information integration. All components of the information integration framework proposed in this thesis are developed based on these foundations.

Information integration has always been one of the important problems in effective management of data, providing building blocks for business intelligence, data analysis, decision making and many other applications. Generally, information integration is performed based on schema mappings, which are high level expressions that show relations between different data sources independent of implementation details. Such mappings are used either in data integration or data exchange. One of the important challenges in information integration is understanding the meaning of data in different data sources – known as semantic heterogeneity reconciliation. The need for semantic heterogeneity reconciliation emerges from the fact that same concept can be shown using different representations. In either data integration or data exchange approaches for information integration, appropriately handling semantic heterogeneities is a key success factor.

In spite of a large amount of research in information integration, semantic issues are not well studied. First, many existing techniques for schema mapping (Alexe et al.,

2008; Bohannon et al., 2006; Bonifati et al., 2005; Bonifati et al., 2010; Fagin et al., 2009; Fuxman et al., 2006; Marnette et al., 2011; Miller et al., 2000; Popa et al., 2002; Raffio et al., 2008; An et al., 2007) are not able to handle ambiguous mappings. Second, some intuitive mappings are neglected by these systems. To address these problems, I proposed a technique to semantically enrich schema mapping using conceptual models called SESM (Chapter 3). This technique recovers the semantics of generalization relations based on the semantics of conceptual models. The result of a case study performed on the SESM shows that a considerable number of ambiguous mappings can be addressed using this technique. Moreover many new mappings based on the concept of composites in Bunge's ontology (Bunge, 1977) can be generated that are largely ignored in many existing works. I showed the success of using SESM varies based on the characterisations of source and target schemas (e.g., the number of generalization relations that are implemented differently in various data sources).

In this thesis, I showed that relying solely on direct property correspondences is not capable of handling all semantic heterogeneities (Chapter 4). As a result, data integration or data exchange systems that employ these mappings fail to capture a rich set of relations between data sources. I contended that gaining the full semantics in schema mapping requires establishing a comprehensive set of relations. For this purpose, I used local and global property precedence schemas (representing fundamental relations between properties). The experiments performed on various datasets in different domains confirmed that this technique can result in a more complete binding between source and target that consequently generates a more complete set of schema mappings. Using a

global property precedence schema that represents relations between two similar properties at different levels of abstraction, I proposed a new approach in data integration in which a user can compromise accuracy to achieve a more complete result (Chapter 4). Experiments performed on various data sources showed how different query rewriting algorithms proposed in this thesis can provide such flexibly in data integration.

In the materialization technique for information integration (data exchange), the prevailing approach has been based on schema mapping, in which schema mapping expressions are used to generate the target instance. Clio (Fagin et al., 2009; Miller et al., 2000; Popa et al., 2002) has pioneered this approach, and many subsequent research prototypes such as (Bonifati et al., 2010; Hernández et al., 2008) are proposed based on this approach. I showed that data exchange based on schema mapping expressions cannot handle all semantic heterogeneities because the semantics of the data transformation are confined in the semantics of classes (tables). I showed how the assumption of inherent classification (Parsons & Wand, 2000) in mapping expression has resulted in this problem. To address this issue, I proposed an Entity Preserving Data Exchange (EDEX), in which transformation is performed independent of classification of instances in the source and the target (Chapter 5). The experiments performed on various data sources shows the effectiveness of using this approach compared to ++Spicy (Marnette et al., 2011), one of the leading existing projects and open source implementation of Clio.

7.2 Significance of the Thesis

The significance of this thesis is in providing ontological foundations for semantic information integration and exploiting these foundations to improve quality and

efficiency of information integration. Unlike existing information integration techniques that fail to capture the details and richness of relationships between concepts, I have proposed a set of semantic heterogeneity reconciliation techniques to address this problem by semantic enrichment of information integration process.

One important aspect of this thesis is introducing a new approach to capture similarities in information integration. Unlike many existing works that look for similarities in similar structures, I have proposed a semantic heterogeneity reconciliation approach in which implicit similarities are inferred from existing data and metadata. The importance of this approach is that allows establishing rich semantic associations between structures that seem different, but they are semantically similar. The key consequence of the semantic enrichment of association between data sources is sketching a new insight into data integration which provides flexibility and tradeoffs between complete and sound data integration. Moreover, human attention and semantic clues about the data, metadata and relationships can be reused to speed up information integration.

Another significance of this thesis is in identifying and addressing ambiguous data exchange scenarios in existing data exchange. In particular, I claim that confining the relations between data sources in terms of associations between a set of classes is the root problem of ambiguity in data exchange. To address this problem, I proposed a new approach for data exchange in which data transformation is performed independent of classification of entities. This approach not only alleviates the problems of ambiguous data exchange scenarios, but also outperforms many existing approach in terms of the quality of data exchange and execution time.

Although the techniques proposed in thesis have made many improvements in information interoperability, there are some drawbacks that must be addressed. First, although users' knowledge is reused by employing property precedence schemas, the quality of semantic enhancement technique proposed in Chapter 4 depends on the quality of property precedence schemas. Second, increasing the quality of data exchange technique proposed inChapter 5 is achieved at the cost of extra storage space required to exchange data. Although this problem is partially addressed using a "process-as-generate" technique, there is room to completely address this problem.

7.3 Realization of Research Goals

This section briefly summarizes how the how research questions in Chapter 1 are addressed in this thesis.

a) What are the root problems of existing information integration techniques?

In this thesis, I argued that relying solely on syntax and structure of data and neglecting rich semantics between data items hinders effective information integration. I showed that simple correspondences between properties that do not contain semantic knowledge beyond the simple rule of "this property in source matches that property in that target," are not able to fully reconcile semantic heterogeneities in information integration. I showed such simple correspondences could fail to capture the detail and richness of relationships that might exist between concepts and may miss valid semantic connections between two data sources.

I contend that existing information integration technique suffer from the lack of theoretical foundations to show how some portion of the data is equivalent to some other portion of the data. I showed how ontological principles can provide theoretical foundations to reconcile semantic heterogeneities in information integration.

Another important root problem in existing information integration techniques is that they only deal with schema level relations between source and target schemas. I argued that semantic heterogeneities in many data integration and data exchange scenarios are not completely resolved because of the gap between data level and schema level approaches. I showed how this gap can result in ambiguity in interpreting schema mappings and consequently improper information integration.

b) How can ontological foundations be used to enrich semantic heterogeneity reconciliation techniques?

I proposed three different techniques to enhance semantic heterogeneity reconciliation techniques. In the first technique, I used implicit semantics of conceptual models to recover the semantics of associations in the relational database. In the second technique, I used property precedence relations to bind similar properties that are presented at different levels of abstracts. In the third technique, I used the concept of composite in Bunge's ontology to explore some new semantic mappings that are neglected in existing schema mapping techniques.

In the first technique, I proposed an algorithm in Section 3.4.1 to recover the semantics of generalization relations from a conceptual model (Algorithm I). Then, I used these enhanced schemas to generate new type of mappings (called manifestation-based

mappings) by employing Algorithms II and Algorithms III. Unlike many existing post processing techniques that attempt to resolve ambiguities and verify mappings after mapping generation, these algorithms addresses this problem by preprocessing relational schemas and recovering the semantics of generalization relations based on conceptual models. Table 3-1 in Section 3.5 shows that for error prone mapping scenarios, a significant number of ambiguous mappings is generated by ++Spicy (representative of existing Clio-based algorithms), while the technique I have proposed in this thesis is able to resolve such ambiguities without negatively affecting unambiguous scenarios.

In the second technique, in Chapter 4, I showed how the concept of property precedence can be used to enhance schema mappings. Using simple property precedence relations, an implicit property of an instance can be inferred from its other properties (Algorithm I in Section 4.4.1). On the other hand, using compound property precedence, a property of an instance can be inferred from properties of another instance (under the condition of possessing some specific properties by those instances) (Algorithm II in Section 4.4.1). Using property precedence relations, I introduced and generated an extended version of the logical association (called manifestation based association), which is a logical association enhanced by semantics of relations. I proposed a mapping generator algorithm that generates a set of manifestation based mappings, in which mappings are accompanied with some specific properties to resolve ambiguous cases (Algorithm III in Section 4.4.2). As shown in Figure 4-5, using property precedence relations can result in increasing the completeness of query answering compared to Clio-based mapping generator. The experiments in Section 4.5 show that these algorithms find

some new mappings regarding the semantics of implicit relations between properties that are generated in traditional Clio-based mapping algorithms.

In the third technique, I used the concepts of *composites* and *emergent properties* as an ontological foundation for the problem of *sequence of relations* (Section 3.4.1.3). Using these concepts and Algorithm IV in Section 3.4.1.3, I showed that indirect relations between entities in a schema can be employed to create some new plausible semantic mappings that are not identified in many existing techniques (Marnette et al., 2011; Miller et al., 2000; Popa et al., 2002). I performed a case study in the healthcare domain to show the effectiveness of this approach in heterogeneity avoidance without need for human intervention after generating mappings. As shown in Table 3-2 in Section 3.5, a considerable number of mappings are generated based on the concept of composites in many mapping scenarios, which shows the importance of identifying composites and relations between them for capturing semantic relations between entities.

c) What does it mean to accurately or completely answer a query given semantically enriched relations between data items?

As an important consequence of binding similar properties that are represented at different levels of abstraction, I proposed a new data integration approach that allows exploring potential answers in data integration with the cost of losing accuracy. This new approach provides tradeoff between accuracy and completeness.

In Chapter 4, I proposed the Configurable Data Integration (CDI) approach that provides a flexible query rewriting. This configurable and flexible query rewriting is realized through implementing two different query expansion algorithms presented in

Algorithm IV in Section 4.4.3. CDI rewrites a query posed to the target into a set of source queries where evaluating the union of these queries on the sources results in the same outcome as running this query on the instance that is materialized in the target. CDI employs global property precedence schemas to bind similar properties that are represented at different levels of abstraction. Using this auxiliary information, CDI allows establishing the tradeoff between accuracy and completeness in query answering. This is achieved through using different approaches for query expansion in replacing general terms with more specific terms or vice versa. Different query rewriting scenarios using these query expansion algorithms show what it means to accurately or completely answer a query given global property precedence relations between properties of source and target.

d) How is it possible to address the problem of ambiguous scenarios in data exchange and data integration? Can ontological foundations be used to address these problems?

In Chapter 5, I investigated many existing data exchange systems which are based on schema mappings. I showed that class based mapping expressions are not capable of handling many ambiguous cases in data exchange, that finally result in generating incorrect target instances. I attributed this problem to the *assumption of inherent classification* in schema mapping. In Section 5.2.1, I discussed why confining the semantics of data transformation in the semantics of classes is in contradiction with a fundamental concept in Bunge's ontology (which adheres to the fact that things exist in the real-world regardless of the classes they belong). This provided a basis for a new

approach which is independent of classification and provides unambiguous data exchange.

To address the problem of the assumption of inherent classification in existing schema mapping based data exchange techniques, I proposed an entity preserving approach (EDEX) for data exchange, in which the focus is on preserving source entities in the target regardless of the classes they belong to in the source. I introduced the concept of super entity to capture indirect properties of an entity in Section 5.4. Several experiments were performed on many data exchange scenarios to show the effectiveness of using EDEX to resolve ambiguous scenarios. Table 5-2 shows that using EDEX can results in resolving many ambiguous cases that cannot be handled properly by many existing data exchange systems. The result of experiments in Section 5.7 shows that the ability of EDEX in reconciling semantic heterogeneities varies depending on different characteristics of data.

e) How can we improve the performance of data exchange process using ontological foundations?

In Chapter 6, to improve the performance of EDEX, I proposed a Sliced Column Store (SCS) model to improve read-oriented queries. This storage model speeds up the overall process of data transformation through improving read-oriented queries. Similar to other components of the information integration framework proposed in this thesis, SCS is also grounded in ontological foundations providing a theoretical basis for column-store databases based on representational adequacy. Two different sets of experiments performed on this storage model showed the performance of SCS compared to row store

and pure column store techniques. The experiments performed in Section 5.7.4 showed that this storage model can improve the efficiency of data exchange in terms of execution time with a slight increase in the storage space required to perform data exchange.

7.4 Future Work

This thesis opens a new research area for future work towards semantic data integration and data exchange. Development of a mapping language to express relations between source and target schemas independent of classification can be useful to formalize the approach proposed in this thesis. Another direction for future work would be exploring implicit semantics in conceptual models to generate new plausible mappings. Using this technique, it can be expected to generate some new complex mapping expressions that cannot be generated by current mapping techniques.

Although this thesis showed the usefulness of using property precedence relations to semantically enrich schema mappings and query answering, further research is required to automatically generate property precedence relations and validate them. In particular, I believe machine learning techniques such as associations rule mining technique can be used for this purpose. In addition, as many large knowledge bases such as YAGO (Suchanek et al., 2007) and Freebase (Suchanek et al., 2008) are publicly available, they can be used to discover semantic relations between properties to enhance the schema mapping process.

Since the experiments in this thesis are performed on enterprise level data, further comprehensive experiments are required to find if property precedence relations can be extracted and used on big data. In addition, finding semantic mappings between big

knowledge bases and enterprise level data in terms of property precedence relations would be an interesting problem for future work.

The configurable data integration approach proposed in this thesis can also be extended in many different directions. First, we need to know how certain query expansion algorithm can be combined to achieve a certain level of completeness or soundness. Improving the efficiency of data exchange system proposed in this thesis can be another direction for future research. In particular, generating intermediate super entities during data exchange process is one of the bottlenecks of EDEX where further research is required to address this problem.

7.5 References

Alexe, B., Chiticariu, L., Miller, R. J., & Wang-Chiew Tan. (2008). Muse: mapping understanding and design by example. *Proceedings of the IEEE 24rd International Conference on Data Engineering,* Cancún, México. 10-19. doi: 10.1109/ICDE.2008.4497409

An, Y., Borgida, A., Miller, R. J., & Mylopoulos, J. (2007). A semantic approach to discovering schema mapping expressions. *Proceedings of the IEEE 23rd International Conference on Data Engineering,* Istanbul, Turkey. 206-215. doi: 10.1109/ICDE.2007.367866

Bohannon, P., Elnahrawy, E., Fan, W., & Flaster, M. (2006). Putting context into schema matching. *Proceedings of the 32nd International Conference on Very Large Data Bases,* Seoul, Korea. 307-318.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. Vancouver, Canada, 1247-1250.

Bonifati, A., Chang, E. Q., Lakshmanan, A. V. S., Ho, T., & Pottinger, R. (2005). HePToX: marrying XML and heterogeneity in your P2P databases. *Proceedings of*

the 31st International Conference on Very Large Data Bases, Trondheim, Norway. 1267-1270.

Bonifati, A., Chang, E., Ho, T., Lakshmanan, L. V., Pottinger, R., & Chung, Y. (2010). Schema mapping and query translation in heterogeneous P2P XML databases. *The VLDB Journal, 19*(2), 231-256. doi: 10.1007/s00778-009-0159-9

Bunge, M. (1977). *Treatise on Basic Philosophy*: *the Furniture of the World*. Boston, MA: Reidel.

Fagin, R., Haas, L. M., Hernández, M., Miller, R. J., Popa, L., & Velegrakis, Y. (2009). Conceptual modeling: foundations and applications. In A. Borgida, er T., V. K. Chaudhri, P. Giorgini & E. S. Yu (Eds.), *Essays in Honor of John Mylopoulos* (pp. 198-236). Berlin, Heidelberg: Springer-Verlag. doi: 10.1007/978-3-642-02463-4_12

Fuxman, A., Hernandez, M. A., Ho, H., Miller, R. J., Papotti, P., & Popa, L. (2006). Nested mappings: schema mapping reloaded. *Proceedings of the 32nd International Conference on Very Large Data Bases*, Seoul, Korea. 67-78.

Hernández, M. A., Papotti, P., & Tan, W. (2008). Data exchange with data-metadata translations. *Proceedings of the VLDB Endowment, 1*(1), 260-273.

Marnette, B., Mecca, G., Papotti, P., Raunich, S., & Santoro, D. (2011). ++Spicy: an open-source tool for second-generation schema mapping and data exchange. *Proceedings of the VLDB Endowment, 4*(12), 1438-1441.

Miller, R. J., Haas, L. M., & Hernández, M. A. (2000). Schema mapping as query discovery. *Proceedings of the 26th International Conference on Very Large Data Bases*, Cairo, Egypt. 77-88.

Parsons, J., & Wand, Y. (2000). Emancipating instances from the tyranny of classes in information modeling. *ACM Transactions on Database Systems, 25*(2), 228-268. doi: 10.1145/357775.357778

Popa, L., Velegrakis, Y., Hernández, M. A., Miller, R. J., & Fagin, R. (2002). Translating Web data. *Proceedings of the 28th International Conference on Very Large Data Bases*, Hong Kong, China. 598-609.

Raffio, A., Braga, D., Ceri, S., Papotti, P., & Hernandez, M. A. (2008). Clip: a visual language for explicit schema mappings. *Proceedings of the IEEE 24th International Conference on Data Engineering*, Cancún, México. 30-39. doi: 10.1109/ICDE.2008.4497411

Suchanek, F. M., Kasneci, G., Weikum G. (2007). YAGO - A Core of Semantic Knowledge. *Proceedings of the 16th international conference on World Wide Web*. Banff, Alberta, Canada, 697-706

Bibliography

Abadi, D., Myers, D. S., DeWitt, D. J., & Madden, S. R. (2007). Materialization strategies in a column-oriented DBMS. *Proceedings of the IEEE 23rd International Conference on Data Engineering,* Istanbul, Turkey. 466-475. doi: 10.1109/ICDE.2007.367892

Abadi, D. J., Madden, S. R., & Hachem, N. (2008). Column-stores vs. row-stores: how different are they really? *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Vancouver, Canada. 967-980. doi: 10.1145/1376616.1376712

Agrawal, S., Narasayya, V., & Yang, B. (2004). Integrating vertical and horizontal partitioning into automated physical database design. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Paris, France. 359-370. doi: 10.1145/1007568.1007609

Alexe, B., Chiticariu, L., & Tan, W. (2006). Spider: A schema mapping debugger. *Proceedings of the 32nd International Conference on Very Large Data Bases,* Seoul, Korea. 1179-1182.

Alexe, B., Chiticariu, L., Miller, R. J., & Wang-Chiew Tan. (2008). Muse: mapping understanding and design by example. *Proceedings of the IEEE 24rd International Conference on Data Engineering,* Cancún, México. 10-19. doi: 10.1109/ICDE.2008.4497409

Alexe, B., Tan, W., & Velegrakis, Y. (2008). Stbenchmark: towards a benchmark for mapping systems. *Proceedings of the VLDB Endowment, 1*(1), 230-244.

Alexe, B., Hernández, M., Popa, L., & Tan, W. (2010). Mapmerge: correlating independent schema mappings. *Proceedings of the VLDB Endowment, 3*(1-2), 81-92.

Alexe, B., ten Cate, B., Kolaitis, P. G., & Tan, W. (2011). EIRENE: Interactive design and refinement of schema mappings via data examples. *Proceedings of the VLDB Endowment, 4*(12), 1414-1417.

Alexe, B., ten Cate, B., Kolaitis, P. G., & Tan, W. (2011). Designing and refining schema mappings via data examples. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Athens, Greece. 133-144. doi: 10.1145/1989323.1989338

An, Y., Borgida, A., Miller, R. J., & Mylopoulos, J. (2007). A semantic approach to discovering schema mapping expressions. *Proceedings of the IEEE 23rd*

International Conference on Data Engineering, Istanbul, Turkey. 206-215. doi: 10.1109/ICDE.2007.367866

An, Y., & Song, I. (2008). Discovering semantically similar associations (SeSA) for complex mappings between conceptual models. *Proceedings of the 27th International Conference on Conceptual Modeling,* Barcelona, Spain. 369-382. doi: 10.1007/978-3-540-87877-3_27

An, Y., Hu, X., & Song, I. (2010). Maintaining mappings between conceptual models and relational schemas. *Journal of Database Management, 21*(3), 36-68.

Arch-Int, N., Li, Y., Roe, P., & Sophatsathit, P. (2003). Query processing the heterogeneous information sources using ontology-based approach. *Proceedings of the 18th International Conference on Computers and their Applications,* Honolulu, HI, USA. 438-441.

Arenas, M., Barceló Pablo, Fagin, R., & Libkin, L. (2004). Locally consistent transformations and query answering in data exchange. *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems,* Paris, France. 229-240. doi: 10.1145/1055558.1055592

Beath, C., Becerra-Fernandez, I., Ross, J., & Short, J. (2012). Finding value in the information explosion. *MIT Sloan Management Review, 53*(4), 18.

Belhajjame, K., Paton, N. W., Embury, S. M., Fernandes, A. A. A., & Hedeler, C. (2010). Feedback-based annotation, selection and refinement of schema mappings for dataspaces. *Proceedings of the 13th International Conference on Extending Database Technology,* Lausanne, Switzerland. 573-584. doi: 10.1145/1739041.1739110

Bellahsene, Z., Bonifati, A. & Rahm, E. (2011). *Schema Matching and Mapping*. Berlin, Heidelberg: Springer-Verlag.

Bohannon, P., Elnahrawy, E., Fan, W., & Flaster, M. (2006). Putting context into schema matching. *Proceedings of the 32nd International Conference on Very Large Data Bases,* Seoul, Korea. 307-318.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. Vancouver, Canada, 1247-1250.

Boncz, P. A., Zukowski, M., & Nes, N. (2005). MonetDB/X100: hyper-pipelining query execution. *Proceedings of the Biennial Conference on Innovative Data Systems Research,* Asilomar, CA, USA. 225-237.

Bonifati, A., Chang, E. Q., Lakshmanan, A. V. S., Ho, T., & Pottinger, R. (2005). HePToX: marrying XML and heterogeneity in your P2P databases. *Proceedings of the 31st International Conference on Very Large Data Bases*, Trondheim, Norway. 1267-1270.

Bonifati, A., Mecca, G., Pappalardo, A., Raunich, S., & Summa, G. (2008). Schema mapping verification: the spicy way. *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, Nantes, France. 85-96. doi: 10.1145/1353343.1353358

Bonifati, A., Mecca, G., Pappalardo, A., Raunich, S., & Summa, G. (2008). The spicy system: towards a notion of mapping quality. *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, Vancouver, Canada. 1289-1294. doi: 10.1145/1376616.1376757

Bonifati, A., Chang, E., Ho, T., Lakshmanan, L. V., Pottinger, R., & Chung, Y. (2010). Schema mapping and query translation in heterogeneous P2P XML databases. *The VLDB Journal, 19*(2), 231-256. doi: 10.1007/s00778-009-0159-9

Bonifati, A., Mecca, G., Papotti, P., & Velegrakis, Y. (2011). Discovery and correctness of schema mapping transformations. In Bellahsene, Z., Bonifati, A. & Rahm, E. (Ed.), *Schema Matching and Mapping* (pp. 111-147). Berlin, Heidelberg: Springer-Verlag.

Bunge, M. (1977). *Treatise on Basic Philosophy*: *the Furniture of the World*. *Boston, MA: Reidel.*

Calì, A., Gottlob, G., & Lukasiewicz, T. (2009). Datalog±: a unified approach to ontologies and integrity constraints. *Proceedings of the 12th International Conference on Database Theory*, St. Petersburg, Russia. 14-30. doi: 10.1145/1514894.1514897

Calì, A., Gottlob, G., & Pieris, A. (2012). Ontological query answering under expressive Entity–Relationship schemata. *Information Systems, 37*(4), 320-335.

Calvanese, D., De Giacomo, G., Lenzerini, M., & Vardi, M. Y. (2013). Query processing under GLAV mappings for relational and graph databases. *Proceedings of the 39th International Conference on Very Large Data Bases*, Trento, Italy. 61-72.

Cappellari, P., Barbosa, D., & Atzeni, P. (2010). A framework for automatic schema mapping verification through reasoning. *Proceedings of the IEEE Data Engineering Workshops,* Long Beach, CA, USA, 245-250. doi: 10.1109/ICDEW.2010.5452703

Carnap, R., & George, R. A. (1969). *The logical Structure of the World: and, Pseudoproblems in Philosophy*. Berkley, CA: Open Court Publishing.

Casanova, M. A., Fagin, R., & Papadimitriou, C. H. (1982). Inclusion dependencies and their interaction with functional dependencies. *Proceedings of the 1st ACM SIGACT-SIGMOD Symposium on Principles of Database Systems,* Los Angeles, California. 171-176. doi: 10.1145/588111.588141

Ceusters, W., Smith, B., & Fielding, J. M. (2004). LinkSuiteTM: formally robust ontology-based data and information integration. *Data Integration in the Life Sciences, 2994*(1), 124-139.

Chan, C., Ioannidis, Y. E. (1998). Bitmap index design and evaluation. Proceedings of the 1998 ACM SIGMOD international conference on Management of data. Seattle, Washington, USA, 355-366. doi: 10.1145/276304.276336

Chen, P. P. (1976). The entity-relationship model-Toward a unified view of data. *ACM Transactions on Database Systems, 1*(1), 9-36. doi: 10.1145/320434.320440

Chiticariu, L., Kolaitis, P. G., & Popa, L. (2008). Interactive generation of integrated schemas. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Vancouver, Canada. 833-846. doi: 10.1145/1376616.1376700

Correndo, G., Salvadores, M., Millard, I., Glaser, H., & Shadbolt, N. (2010). SPARQL query rewriting for implementing data integration over linked data. *Proceedings of the EDBT/ICDT Workshops,* Lausanne, Switzerland. 4:1-4:11. doi: 10.1145/1754239.1754244

Das Sarma, A., Dong, X. L., & Halevy, A. Y. (2008). Bootstrapping pay-as-you-go data integration systems. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Vancouver, Canada. 861-874. doi: 10.1145/1376616.1376702

Das Sarma, A., Dong, X. L., & Halevy, A. Y. (2011). Uncertainty in data integration and dataspace support platforms. In Bellahsene, Z., Bonifati, A., & Rahm, E. (Ed.), *Schema Matching and Mapping* (pp. 75-108). Berlin, Heidelberg: Springer-Verlag.

Doan, A., & Halevy, A. Y. (2005). Semantic integration research in the database community: a brief survey. *AI Magazine, 26*(1), 83-94.

Doan, A., Halevy, A. Y., & Ives, Z. (2012). *Principles of Data Integration,* Waltham, MA: Morgan Kaufmann.

Dong, X. L., Halevy, A. Y., & Yu, C. (2009). Data integration with uncertainty. *The VLDB Journal, 18*(2), 469-500. doi: 10.1007/s00778-008-0119-9

Elmeleegy, H., Elmagarmid, A., & Lee, J. (2011). Leveraging query logs for schema mapping generation in U-MAP. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Athens, Greece. 121-132. doi: 10.1145/1989323.1989337

Embley, D. W., Xu, L., & Ding, Y. (2004). Automatic direct and indirect schema mapping: experiences and lessons learned. *SIGMOD Record, 33*(4), 14-19. doi: 10.1145/1041410.1041413

Evermann, J., & Wand, Y. (2005). Ontology based object-oriented domain modelling: fundamental concepts. *Requirements Engineering, 10*(2), 146-160. doi: 10.1007/s00766-004-0208-2

Evermann, J. (2009). A UML and OWL description of Bunge's upper-level ontology model. *Software & Systems Modeling, 8*(2), 235-249. doi: 10.1007/s10270-008-0082-3

Fagin, R., Kolaitis, P. G., Miller, R. J., & Popa, L. (2005). Data exchange: semantics and query answering. *Theoretical Computer Science, 336*(1), 89-124. doi: 10.1016/j.tcs.2004.10.033

Fagin, R., Kolaitis, P. G., Popa, L., & Tan, W. (2005). Composing schema mappings: second-order dependencies to the rescue. *ACM Transactions on Database Systems, 30*(4), 994-1055. doi: 10.1145/1114244.1114249

Fagin, R., Kolaitis, P. G., & Popa, L. (2005). Data exchange: getting to the core. *ACM Transactions on Database Systems, 30*(1), 174-210. doi:10.1145/1061318.1061323

Fagin, R. (2007). Inverting schema mappings. *ACM Transactions on Database Systems, 32*(4), 25-53. doi: 10.1145/1292609.1292615

Fagin, R., Haas, L. M., Hernández, M., Miller, R. J., Popa, L., & Velegrakis, Y. (2009). Conceptual modeling: foundations and applications. In A. Borgida, er T., V. K. Chaudhri, P. Giorgini & E. S. Yu (Eds.), *Essays in Honor of John Mylopoulos* (pp. 198-236). Berlin, Heidelberg: Springer-Verlag. doi: 10.1007/978-3-642-02463-4_12

Fagin, R., Kimelfeld, B., Li, Y., Raghavan, S., & Vaithyanathan, S. (2011). Rewrite rules for search database systems. *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems,* Athens, Greece. 271-282. doi: 10.1145/1989284.1989322

Fletcher, G. H., & Wyss, C. M. (2006). Data mapping as search. *Advances in Database Technology, 3896*(1), 95-111. doi:10.1007/11687238_9

Fonseca, F. (2007). The double role of ontologies in information science research: research articles. *Journal of the American Society for Information Science and Technology, 58*(6), 786-793. doi: 10.1002/asi.v58:6

Fuxman, A., Hernandez, M. A., Ho, H., Miller, R. J., Papotti, P., & Popa, L. (2006). Nested mappings: schema mapping reloaded. *Proceedings of the 32nd International Conference on Very Large Data Bases,* Seoul, Korea. 67-78.

Fuxman, A., Kolaitis, P. G., Miller, R. J., & Tan, W. (2006). Peer data exchange. *ACM Transactions on Database Systems, 31*(4), 1454-1498. doi: 10.1145/1189769.1189778

Gal, A. (2006). Managing uncertainty in schema matching with top-K schema mappings. *Journal on Data Semantics*, *4090*(1), 90-114. doi: 10.1007/11803034_5

Gemino, A., & Wand, Y. (2004). A framework for empirical evaluation of conceptual modeling techniques. *Requirements Engineering, 9*(4), 248-260. doi: 10.1007/s00766-004-0204-6

George, D., & Preston, U. (2005). Understanding structural and semantic heterogeneity in the context of database schema integration. *Journal of the Department of Computing, UCLAN 4*(1), 29-44.

Glavic, B., Alonso, G., Miller, R. J., & Haas, L. M. (2010). TRAMP: understanding the behavior of schema mappings through provenance. *Proceedings of the VLDB Endowment, 3*(1-2), 1314-1325.

Gottlob, G., & Nash, A. (2008). Efficient core computation in data exchange. *Journal of the ACM, 55*(2), 9:1-9:49. doi: 10.1145/1346330.1346334

Haas, L. M. (2006). Beauty and the beast: the theory and practice of information integration. *Proceedings of the 11th International Conference on Database Theory,* Barcelona, Spain. 28-43. doi: 10.1007/11965893_3

Haas, L. M., Hentschel, M., Kossmann, D., & Miller, R. J. (2009). Schema AND data: a holistic approach to mapping, resolution and fusion in information integration. *Proceedings of the 28th International Conference on Conceptual Modeling,* Gramado, Brazil. 27-40. doi: 10.1007/978-3-642-04840-1_5

Hakimpour, F., & Geppert, A. (2001). Resolving semantic heterogeneity in schema integration. *Proceedings of the International Conference on Formal Ontology in Information Systems*, Ogunquit, ME, USA. 297-308. doi: 10.1145/505168.505196

Halevy, A. Y., Rajaraman, A., & Ordille, J. (2006). Data integration: the teenage years. *Proceedings of the 32nd International Conference on Very Large Data Bases,* Seoul, Korea. 9-16.

Halevy, A. Y. (2010). Technical perspective schema mappings: rules for mixing data. *Communications of the ACM, 53*(1), 100-101.

Harizopoulos, S., Liang, V., Abadi, D. J., & Madden, S. (2006). Performance tradeoffs in read-optimized databases. *Proceedings of the 32nd International Conference on Very Large Data Bases,* Seoul, Korea. 487-498.

Hassanzadeh, O., Kementsietsidis, A., Lim, L., Miller, R. J., & Wang, M. (2009). A framework for semantic link discovery over relational data. *Proceedings of the 18th ACM Conference on Information and Knowledge Management,* Hong Kong, China. 1027-1036. doi: 10.1145/1645953.1646084

Heinz, S., Zobel, J., & Williams, H. E. (2002). Burst tries: a fast, efficient data structure for string keys. *ACM Transactions on Information Systems, 20*(2), 192-223. doi: 10.1145/506309.506312

Hentschel, M., Kossmann, D., Florescu, D., Haas, L. M., Kraska, T., & Miller, R. J. (2009). Scalable data integration by mapping data to queries. *ETH Zurich, Computer Science, Technical Report, 633.*

Hernández, M. A., Papotti, P., & Tan, W. (2008). Data exchange with data-metadata translations. *Proceedings of the VLDB Endowment, 1*(1), 260-273.

Herodotou, H., Borisov, N., & Babu, S. (2011). Query optimization techniques for partitioned tables. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Athens, Greece. 49-60. doi: 10.1145/1989323.1989330

Idreos, S., Kersten, M. L., & Manegold, S. (2007). Database cracking. *Proceedings of the Biennial Conference on Innovative Data Systems Research,* Asilomar, CA, USA. 342-350.

Idreos, S., Manegold, S., Kuno, H., & Graefe, G. (2011). Merging what's cracked, cracking what's merged: adaptive indexing in main-memory column-stores. *Proceedings of the VLDB Endowment, 4*(9), 586-597.

Ives, Z. G., Halevy, A. Y., Mork, P., & Tatarinov, I. (2004). Piazza: mediation and integration infrastructure for semantic Web data. *Web Semantics: Science, Services and Agents on the World Wide Web, 1*(2), 155-175. doi: 10.1016/j.websem.2003.11.003

Ives, Z. G., Green, T. J., Karvounarakis, G., Taylor, N. E., Tannen, V., Talukdar, P. P., Pereira, F. (2008). The ORCHESTRA collaborative data sharing system. *SIGMOD Record, 37*(3), 26-32. doi: 10.1145/1462571.1462577

Jiang, H., Ho, H., Popa, L., & Han, W. (2007). Mapping-driven XML transformation. *Proceedings of the 16th International Conference on World Wide Web,* Banff, Alberta, Canada. 1063-1072. doi: 10.1145/1242572.1242715

Jurisica, I., Mylopoulos, J., & Yu, E. (2004). Ontologies for knowledge management: an information systems perspective. *Knowledge and Information Systems, 6*(4), 380-401. doi: 10.1007/s10115-003-0135-4

Kementsietsidis, A., Arenas, M., & Miller, R. J. (2003). Mapping data in peer-to-peer systems: semantics and algorithmic issues. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* San Diego, CA, USA. 325-336. doi: 10.1145/872757.872798

Köpcke, H., & Rahm, E. (2010). Frameworks for entity matching: a comparison. *Data & Knowledge Engineering, 69*(2), 197-210. doi: 10.1016/j.datak.2009.10.003

Larson, P., Clinciu, C., Hanson, E. N., Oks, A., Price, S. L., Rangarajan, S., Surna, A., & Zhou, Q. (2011). SQL server column store indexes. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Athens, Greece. 1177-1184. doi: 10.1145/1989323.1989448

Lemke, C., Sattler, K., Faerber, F., & Zeier, A. (2010). Speeding up queries in column stores: a case for compression. *Proceedings of the 12th International Conference on Data Warehousing and Knowledge Discovery,* Bilbao, Spain. 117-129.

Lu, M., Agrawal, D., Dai, B. T., & Tung, A. K. (2011). Schema-as-you-go: on probabilistic tagging and querying of wide tables. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Athens, Greece. 181-192.

MacNicol, R., & French, B. (2004). Sybase IQ multiplex - designed for analytics. *Proceedings of the VLDB Endowment, 30*(1), 1227-1230.

Madhavan, J., Jeffery, S., Cohen, S., Dong, X. L., Ko, D., Yu, C., & Halevy, A. Y. (2007). Web-scale data integration: you can only afford to pay as you go. *Proceedings of the Biennial Conference on Innovative Data Systems Research,* Asilomar, CA, USA. 342-350.

Magnani, M., Rizopoulos, N., McBrien, P., & Montesi, D. (2005). Schema integration based on uncertain semantic mappings. *Proceedings of the 24th International*

Conference on Conceptual Modeling, Klagenfurt, Austria. 31-46. doi: 10.1007/11568322_3

Magnani, M., & Montesi, D. (2007). Uncertainty in data integration: current approaches and open problems. *Proceedings of the VLDB Workshop on Management of Uncertain Data,* Vienna, Austria. 18-32.

Mandreoli, F., & Martoglia, R. (2011). Knowledge-based sense disambiguation (almost) for all structures. *Information Systems, 36*(2), 406-430. doi: 10.1016/j.is.2010.08.004

Marnette, B., Mecca, G., & Papotti, P. (2010). Scalable data exchange with functional dependencies. *Proceedings of the VLDB Endowment, 3*(1-2), 105-116.

Marnette, B., Mecca, G., Papotti, P., Raunich, S., & Santoro, D. (2011). ++Spicy: an open-source tool for second-generation schema mapping and data exchange. *Proceedings of the VLDB Endowment, 4*(12), 1438-1441.

Mecca, G., Papotti, P., & Raunich, S. (2009). Core schema mappings. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Providence, RI, USA. 655-668. doi: 10.1145/1559845.1559914

Mecca, G., Papotti, P., & Raunich, S. (2012). Core schema mappings: scalable core computations in data exchange. *Information Systems, 37*(7), 677-711. doi: 10.1016/j.is.2012.03.004

Mena, E., Illarramendi, A., Kashyap, V., & Sheth, A. P. (2000). OBSERVER: an approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases, 8*(2), 223-271. doi: 10.1023/A:1008741824956

Miller, R. J., Haas, L. M., & Hernández, M. A. (2000). Schema mapping as query discovery. *Proceedings of the 26th International Conference on Very Large Data Bases,* Cairo, Egypt. 77-88.

Milton, S. K. (2004). Top-level ontology: the problem with naturalism. *Proceedings of the International Conference on Formal Ontology in Information Systems,* Torino, Italy. 85-94.

Mutis, I., & Issa, R. R. (2012). Framework for semantic reconciliation of construction project information, *Journal of Information Technology in Construction*, *17*(1), 1-24

Orsi, G., & Pieris, A. (2011). Optimizing query answering under ontological constraints. *Proceedings of the VLDB Endowment, 4*(11), 1004-1015.

Pablo, B. (2009). Logical foundations of relational data exchange. *SIGMOD Record., 38*(1), 49-58. doi: 10.1145/1558334.1558341

Pankowski, T. (2013). Semantics preservation in schema mappings within data exchange systems. *Proceedings of the 16th International Conference on Knowledge Engineering, Machine Learning and Lattice Computing with Applications,* San Sebastian, Spain. 88-97. doi: 10.1007/978-3-642-37343-5_10

Parsons, J., & Wand, Y. (2000). Emancipating instances from the tyranny of classes in information modeling. *ACM Transactions on Database Systems, 25*(2), 228-268. doi: 10.1145/357775.357778

Parsons, J., & Wand, Y. (2003). Attribute-based semantic reconciliation of multiple data sources. *Journal on Data Semantics*, *2800*(1), 21-47. doi: 10.1007/978-3-540-39733-5_2

Parsons, J., & Cole, L. (2004). An experimental examination of property precedence in conceptual modelling. *Proceedings of the 1st Asian-Pacific Conference on Conceptual Modelling,* Dunedin, New Zealand. 101-110.

Parsons, J., & Chen, T. (2008). Using property precedence to enhance the effectiveness of queries on unstructured data. *Proceedings of 18th Workshop on Information Technology Systems,* Paris, France, 73-78.

Parsons, J., & Wand, Y. (2008). Using cognitive principles to guide classification in information systems modeling. *MIS Quarterly, 32*(4), 839-868.

Parsons, J. (2011). An experimental study of the effects of representing property precedence on the comprehension of conceptual schemas. *Journal of the Association for Information Systems, 12*(6), 1.

Pichler, R., & Savenkov, V. (2010). Towards practical feasibility of core computation in data exchange. *Theoretical Computer Science, 411*(7-9), 935-957. doi: 10.1016/j.tcs.2009.09.035

Popa, L., & Tannen, V. (1999). An equational chase for path-conjunctive queries, constraints, and views. *Proceedings of the 7th International Conference on Database Theory,* Jerusalem, Israel. 39-57.

Popa, L., Velegrakis, Y., Hernández, M. A., Miller, R. J., & Fagin, R. (2002). Translating Web data. *Proceedings of the 28th International Conference on Very Large Data Bases,* Hong Kong, China. 598-609.

Pottinger, R., & Bernstein, P. A. (2008). Schema merging and mapping creation for relational sources. *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology,* Nantes, France. 73-84. doi: 10.1145/1353343.1353357

Qian, L., Cafarella, M. J., & Jagadish, H. V. (2012). Sample-driven schema mapping. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Scottsdale, AZ, USA. 73-84. doi: 10.1145/2213836.2213846

Raffio, A., Braga, D., Ceri, S., Papotti, P., & Hernandez, M. A. (2008). Clip: a visual language for explicit schema mappings. *Proceedings of the IEEE 24th International Conference on Data Engineering,* Cancún, México. 30-39. doi: 10.1109/ICDE.2008.4497411

Roth, M., & Tan, W. (2013). Data integration and data exchange: it's really about time. *Proceedings of the Biennial Conference on Innovative Data Systems Research,* Asilomar, CA, USA. 342-350.

Rull, G., Farré C., Teniente, E., & Urpí Toni. (2009). MVT: a schema mapping validation tool. *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology,* Saint Petersburg, Russia. 1120-1123. doi: 10.1145/1516360.1516492

Rull, G., Farré, C., Teniente, E., & Urpí, T. (2013). Validation of schema mappings with nested queries. *Computer Science and Information Systems, 10*(1), 79-104.

San, R. (2012). *Ventana research 2012 value index for data integration.* (Research). CA, USA: Ventana Research.

Sekhavat, Y. A. (2012). Semantic heterogeneity reconciliation in data integration. *Proceedings of the PhD Workshop of 38th International Conference on Very Large Data Bases,* Istanbul, Turkey. 19-24

Sekhavat, Y. A., & Parsons, J. (2012). Semantic schema mapping using property precedence relations. *Proceedings of the IEEE 6th International Conference on Semantic Computing,* Palermo, Italy. 210-217. doi: 10.1109/ICSC.2012.24

Sekhavat, Y. A., & Parsons, J. (2012). Sliced column-store (SCS): ontological foundations and practical implications. *Proceedings of the 31st International Conference on Conceptual Modeling,* Florence, Italy. 102-115. doi: 10.1007/978-3-642-34002-4_8

Sekhavat, Y. A., & Parsons, J. (2013). SESM: semantic enrichment of schema mappings. *Proceedings of 4th ICDE International Workshop on Data Engineering Meets Semantic Web,* Brisbane, Australia (to appear).

Sekhavat, Y. A., & Parsons, J. (2013). EDEX: Entity Preserving Data Exchange. *Proceedings of DATA'13 International Conference on Data Management Technologies,* Reykjavík, Iceland (to appear)

Shanks, G., Tansley, E., Nuredini, J., Tobin, D., & Weber, R. (2002). Representing part-whole relationships in conceptual modeling: an empirical evaluation. *Proceedings of the 23rd International Conference on Information Systems,* Barcelona, Spain. 89-100.

Shanks, G., Tansley, E., & Weber, R. (2004). Representing composites in conceptual modeling. *Communications of the ACM, 47*(7), 77-80. doi: 10.1145/1005817.1005826

Suchanek, F. M., Kasneci, G., Weikum G. (2007). YAGO - A Core of Semantic Knowledge. *Proceedings of the 16th international conference on World Wide Web*. Banff, Alberta, Canada, 697-706

Stonebraker, M., Abadi, D. J., Batkin, A., Chen, X., Cherniack, M., Ferreira, M., Lau, E., Lin, A., Madden, S., O'Neil, E., O'Neil, P., Rasin, A., Tran, N., Zdonik, S. (2005). C-store: a column-oriented DBMS. *Proceedings of the 31st International Conference on Very Large Data Bases,* Trondheim, Norway. 553-564.

Talburt, J. R. (2011). *Entity Resolution and Information Quality,* Burlington, MA: Morgan Kaufmann.

Tatarinov, I., & Halevy, A. Y. (2004). Efficient query reformulation in peer data management systems. *Proceedings of the ACM SIGMOD International Conference on Management of Data,* Paris, France. 539-550. doi: 10.1145/1007568.1007629

ten Cate, B., Chiticariu, L., Kolaitis, P., & Tan, W. (2009). Laconic schema mappings: computing the core with SQL queries. *Proceedings of the VLDB Endowment, 2*(1), 1006-1017.

ten Cate, B., Kolaitis, P. G., & Tan, W. (2013). Schema mappings and data examples. *Proceedings of the 16th International Conference on Extending Database Technology,* Genoa, Italy. 777-780. doi: 10.1145/2452376.2452479

TPC-H benchmark specification. Retrieved from *Http://www.tcp.org*

Wand, Y., & Weber, R. (1990). Mario Bunge's ontology as a formal foundation for information systems concepts. In Weingartner, P., Dorn, & G. J. W. (Ed.), *Studies on Mario Bunge's Treatise* (pp. 123-149). Atlanta: Rodopi.

Wand, Y. (1996). Ontology as a foundation for meta-modelling and method engineering. *Information and Software Technology, 38*(4), 281-287. doi: 10.1016/0950-5849(95)01052-1

Wand, Y., Storey, V. C., & Weber, R. (1999). An ontological analysis of the relationship construct in conceptual modeling. *ACM Transactions on Database Systems, 24*(4), 494-528. doi: 10.1145/331983.331989

Wang, T., & Pottinger, R. (2008). SeMap: a generic mapping construction system. *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, Nantes, France. 97-108. doi: 10.1145/1353343.1353359

Yu, C., & Popa, L. (2004). Constraint-based XML query rewriting for data integration. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Paris, France. 371-382. doi: 10.1145/1007568.1007611