Tabular Representation of Schema Mappings: Semantics and Algorithms

by

Md. Anisur Rahman

uOttawa

A thesis submitted in conformity with the requirements for
the degree of Doctor of Philosophy in Computer Science,
School of Information Technology and Engineering,
at the University of Ottawa, Canada.

Abstract

Our thesis investigates a mechanism for representing schema mapping by tabular forms and checking utility of the new representation. *Schema mapping* is a high-level specification that describes the relationship between two database schemas. Schema mappings constitute essential building blocks of data integration, data exchange and peer-to-peer data sharing systems. *Global-and-local-as-view* (GLAV) is one of the approaches for specifying the schema mappings. *Tableaux* are used for expressing queries and functional dependencies on a single database in a tabular form. In our thesis, we first introduce a tabular representation of GLAV mappings. We find that this tabular representation helps to solve many mapping-related algorithmic and semantic problems. For example, a well-known problem is to find the minimal instance of the target schema for a given instance of the source schema and a set of mappings between the source and the target schema. Second, we show that our proposed tabular mapping can be used as an operator on an instance of the source schema to produce an instance of the target schema which is 'minimal' and 'most general' in nature. There exists a tableaux-based mechanism for finding equivalence of two queries. Third, we extend that mechanism for deducing equivalence between two schema mappings using their corresponding tabular representations. Sometimes, there exist redundant conjuncts in a schema mapping which causes data exchange, data integration and data sharing operations more time consuming. Fourth, we present an algorithm that utilizes the tabular representations for reducing number of constraints in the schema mappings. At present, either schema-level mappings or data-level mappings are used for data sharing purposes. Fifth, we introduce and give the semantics of *bi-level mapping* that combines the schema-level and data-level mappings. We also show that bi-level mappings are more effective for data sharing systems. Finally, we implemented our algorithms and developed a software prototype to evaluate our proposed strategies.

Acknowledgements

First of all, I glorify the greatness and bounty of almighty Allah who has bestowed on me the strength and ability without which it would not have been possible to carry out the thesis.

I am grateful to my supervisor Dr. Iluju Kiringa who has given me the opportunity to do research to the exiting and evolving field and guided me to the way of innovation and novelty. I am also grateful to my co-supervisor Prof. Abdulmotaleb El Saddik who has inspired and motivated me time to time by providing valuable guidelines and suggestions. Their invaluable suggestions and profound research experience kept me enthusiastic and optimistic all the way to the completion of this thesis. I thank them for their many hours of patience for listening to my problems. Their assistance, comments, constructive criticism, and positive attitude helped me to proceed towards completing each step of the thesis.

I would also like to thank Prof. Amy Felty and Dr. Mengchi Liu for their valuable comments in the thesis proposal defence that have further enriched qualities of the content of this thesis.

I must acknowledge the support and facilities I received from the staff of School of Information Technology and Engineering.

I am also grateful to my friends Anwar, Mehedi, Shamim, Amir, Masud, Mamun and other colleagues in my work place for their constant encouragement, inspiration, and suggestions.

Lastly, I thank my parents, my brother Amanur, my wife Maushumi and my daughters Rameen and Ariya for their constant prayer, inspiration, and support which provided me with optimism in all situations.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Designing a system for integrated access to distributed and heterogeneous information sources, e.g. *data integration*, *data exchange* and *P2P data sharing*, is an important research area. Data integration is the problem of combining data residing at different sources to provide the user with a unified view [42]. Data exchange is the problem of taking data structured under a source schema and creating an instance of a target schema that reflects the source data as accurately as possible [25]. A P2P data sharing system consists of an open-ended network of distributed computational peers, where each peer can exchange data with a set of other peers [7]. Schema Mappings constitute the building blocks of data integration system, data exchange system and P2P data sharing system. A schema mapping describes the relationship between two database schemas at high level. There are three approaches for specifying the schema mappings: *local-as-view* (LAV), *global-as view* (GAV), and *global-and-local-as-view* (GLAV).

GAV approach is global-schema-centric (or target-schema-centric in case of data exchange) and specifies the mappings by a set of *assertions* of the form $\forall\vec{x}(\phi_{\mathcal{S}}(\vec{x}) \rightsquigarrow g(\vec{x}))$ where g is an element of global schema (target schema) $\mathcal{G}$ and $\phi_{\mathcal{S}}$ is a query over source schema $\mathcal{S}$. Relations in $\mathcal{G}$ are views and queries are expressed over the views. Queries can

simply be evaluated over the data satisfying the global relations. LAV approach is source-centric and specifies the mappings by a set of assertions of the form $\forall\vec{x}(s(\vec{x}) \rightsquigarrow \phi_{\mathcal{G}}(\vec{x}))$ where s is an element of $\mathcal{S}$ and $\phi_{\mathcal{G}}$ is a query over $\mathcal{G}$. GLAV is a mixed approach and specifies the mappings by a set of assertions of the form $\forall\vec{x}(\exists\vec{y}\phi_{\mathcal{S}}(\vec{x},\vec{y}) \rightsquigarrow \exists\vec{z}\phi_{\mathcal{G}}(\vec{x},\vec{z}))$ where $\phi_{\mathcal{G}}$ is a query over $\mathcal{G}$ and $\phi_{\mathcal{S}}$ is a query over $\mathcal{S}$. Note that all of GAV, LAV and GLAV mappings are represented by *First Order Logic* (FOL) statements and queries are important part of the mappings. Although, there exist an alternate tabular representation of queries, called *tableaux*, no tabular representation of the schema mapping has been introduced yet.

Tableaux as defined by Aho et al. [2, 4, 5] are tabular notations for a subset of relational calculus queries, characterized by containing only AND-connected terms and no universal quantifiers. Thus tableau queries are a particular kind of conjunctive queries [20, 57]. Tableaux are specialized matrices, the columns of which correspond to the attributes of the underlying database schema. The first row of the matrix, the summary, serves the same purpose as the target list of a relational calculus expression. The other rows describe the predicate. The symbols appearing in a tableau are distinguished variables (corresponding to free variables), nondistinguished variables (corresponding to existentially quantified variables), constants, blanks, and tags (indicating the range relation). Expressions containing disjunction (set union) and negation (set difference) can be represented by sets of tableaux [59]. Klug et al. [37] use sets of tableaux for representing general conjunctive queries.

There might be several equivalent expressions representing the same query. One source of differences between any two equivalent expressions is their degree of redundancy [63]. A straightforward evaluation of a redundant expression would lead to the execution of a set of operations, some of which are superfluous. Therefore query optimization aims at the elimination of redundancy by means of transforming a redundant expression into an equivalent nonredundant one. Algorithms that minimize the number of rows in tableaux systematically exploit such simplification rules for conjunctive queries [4, 5, 58]. Since the

number of rows in a tableau is one more than the number of joins in the expression, the minimization of the number of rows corresponds to the elimination of redundant joins. Sagiv et al. [59] extend the tableau techniques to cover the simplification of expressions containing disjunctions. We extend the idea of representing queries in tabular form to the mechanism of representing GLAV mappings in a tabular form and then use that tabular representation to optimize the schema mappings mechanically.

A GLAV mapping can be viewed as a correspondence between two queries (one query on the source schema and the other on the target schema). So, we express a GLAV mapping by a tabular structure called *mapping assertion tableaux* (MAT). MAT is partitioned into two tables: the left hand side table corresponds to the source query (call it *source tableau*) and the right hand side table corresponds to the target query (call it *target tableau*). The source table acts as the *body* and the target table acts as the *summary* of the MAT. Similarly, we express *tuple generating dependency* (tgd) constraints and the *equality generating dependency* (egd) constraints of a data integration system by tabular structures, called *tabular tgd* (TTGD) and *tabular egd* (TEGD), respectively. A set consisting of MATs, TTGDs and TEGDs is called *schema mapping tableaux* (SMT). We find that SMTs can be optimized mechanically by adapting existing tableau-based query optimization techniques. So, to optimize a FOL GLAV mapping, we represent it by SMT, then optimize the SMT and finally convert the SMT back to the FOL representation. Since our proposed mapping optimization technique is based on the optimization of the tableaux, we call it a *tableaux-based optimization* technique [56].

Even though several different aspects of data integration, data exchange, P2P data sharing system and schema mappings have been explored extensively, the study of schema mapping optimization has not been addressed much. In [26] Fagin et al. introduced the notions of *Data-Exchange Equivalence* and *Conjunctive-Query Equivalence* between the schema mappings of two data exchange systems. Two schema mappings are data-exchange equivalent if they are indistinguishable for data-exchange purposes, and two schema mappings are Conjunctive-Query equivalent if they are indistinguishable for the

purpose of answering conjunctive queries. The authors in [26] presented the necessary conditions for Conjunctive-Query equivalence and left data-exchange equivalence as an open problem. In this thesis, we show that in a data integration system equivalence between two schema mappings can be characterized using our proposed tabular representation of the mappings.

'Generality' and 'Minimality' are two desired properties of an instance of the target schema [26]. We observe that if we use SMT as an operator on source instances to produce a target instance, it produces the 'most general' and 'minimal' target instance. We give detailed syntax and semantics of SMT in the context of data integration, data exchange and data sharing.

Peer data sharing systems use either *schema-level* or *data-level* mappings to resolve schema as well as data heterogeneity among data sources (peers). Schema-level mappings create structural relationships among different schemas. On the other hand, data-level mappings associate data values in two different sources. These two kinds of mappings are complementary to each other. However, existing peer database systems have been based solely on either one of these mappings. We believe that if both mappings are addressed simultaneously in a single framework, the resulting approach will enhance data sharing in a way such that we can overcome the limitations of the non-combined approaches. Besides the schema mapping representation and optimization issues, in our thesis, we also address this issue. We introduce a model of a peer database management system (PDMS) which uses a mapping that combines schema-level and data-level mappings. We call this new kind of mapping *bi-level mapping* [54, 55]. We present the syntax and semantics of bi-level mappings. We also express bi-level mappings by MATs and show that tabular representation of bi-level mappings helps the query translation process.

1.2 Problem Statement

Consider two GLAV mappings m and m' between a source schema $\mathcal{S}$ and a target schema $\mathcal{T}$, where $\mathcal{S}$ has a single relation $P(A, B, C)$ and $\mathcal{T}$ has a single relation $Q(A, B, C)$. m and m' are defined as follows:

$m : \forall x_1 x_2 x_3 (\exists y_1 y_2 y_3 y_4 (P(x_1, x_2, y_1) \wedge P(y_2, x_2, y_3) \wedge P(y_4, x_2, x_3)) \rightsquigarrow Q(x_1, x_2, x_3))$

$m' : \forall x_4 x_5 x_6 (\exists y_1 y_2 (P(x_4, x_5, y_1) \wedge P(y_2, x_5, x_6)) \rightsquigarrow Q(x_4, x_5, x_6))$

Now the question is: are m and m' equivalent regarding answering queries, exchanging data or other purposes? If so, how can we deduce that m and m' are equivalent? Note that m' is simpler than m and is more efficient in query answering and data exchange purposes. Now, if m and m' are equivalent, it is preferable to have m' instead of m. So, a different algorithmic problem is: how can we reduce m to m'? A representational problem is: if FOL representation of the GLAV mappings make the reduction process difficult, can some other form of representation (say, tabular representation) of those mappings make the reduction process easier?

In our thesis we aim to find solutions to these problems in the context of data integration, data exchange and P2P data sharing systems.

1.3 Objectives

Objectives of our thesis are:

- Find an alternative tabular representation of schema mappings, say *schema mapping tableaux* (SMT), that provides some additional reasoning on the mappings.
- Investigate whether SMT has some utility such as using it as an operator on a source instance to produce a target instance with some 'good' properties like 'minimal', 'most general' etc.
- Give semantics of SMT as an operator on a source instance.

- Check how SMT can help query answering.
- Define and characterize equivalence of schema mappings using their corresponding SMTs.
- Present algorithms for optimizing a schema mapping by utilizing it's tabular representation.
- Introduce a mapping that simultaneously resolve schema-level and data-level heterogeneity in a peer-to-peer data sharing system.
- Evaluate the proposed algorithms.

1.4 Thesis Statement and Contribution

The central idea of this thesis is:

> *Schema mappings of data integration, data exchange and P2P data sharing systems can be expressed by tabular forms and this tabular representation helps optimization of the schema mappings.*

To this end, this thesis makes the following contribution:

- Introduce a tabular representation of schema mappings and give the syntax and semantics of the new representation.
- Utilize the tabular representation for checking equivalence of two schema mappings.
- Propose algorithms for optimizing schema mappings.
- Introduce a mapping for peer-to-peer data sharing system that resolves schema-level and data-level heterogeneity simultaneously.
- Design and develop a software tool for evaluating the proposed algorithms.

1.5 Organization of the thesis

The remainder of this thesis is organized as follows. Chapter 2 reviews related literature and presents technical preliminaries. Chapter 3 and 4 provide the syntax and semantics of tabular representation of schema mapping in the area of Data Integration and Data Exchange. These chapters also present some findings related to the usefulness of the tabular representation of schema mappings. Chapter 5 defines the notions of equivalence of schema mappings and characterize them in terms of tabular representation of the schema mappings. Chapter 5 also presents algorithms for optimizing schema mappings by manipulating their tabular forms. Chapter 6 introduces a new type of mapping for peer-to-peer data sharing system that combines data-level and schema-level mappings. Chapter 7 presents implementation and evaluation of the algorithms proposed in this thesis. Finally, Chapter 8 presents a summary of the research and states the future research direction.

Chapter 2

Literature Review and Preliminaries

In this chapter, we present previous works that are broadly related to this thesis. As the goal of this thesis is to derive a tabular representation of schema mappings and show how the new representation can be useful in the area of data integration system, data exchange system and data sharing system, there are four main areas pertinent to this topic. The primary area is the existing representation techniques of schema mappings. The second area of relevance is data integration, data exchange and data sharing systems where the schema mappings are used. The third area of relevance is tabular representation of queries (tableaux queries) from which we were inspired to derive a tabular representation of schema mappings. We use the tabular representation of schema mappings for the optimization of schema mappings. So, the fourth area of relevance is schema mapping optimization. We first review existing literature related to these areas. We also describe the notations related to basic relational model, nulls, data integration, data exchange, data sharing systems, and tableaux queries.

Dept

	D	M	MN
t_1	CS	$\perp_{1_1}$	$Mary$
t_2	CS	$E005$	$Mary$
t_3	CS	$\perp_{2_3}$	$Mary$

Figure 2.1: Example of tuples with null

2.1 Basic Database Notations

In this section, we review some basic database terms and notations like, schema, null, homomorphism etc [44].

Schema. A schema or signature $\mathbf{R}$ is a finite sequence $(R_1, \ldots, R_k)$ of relation symbols, each of a fixed arity. An instance $\mathcal{C}$ over $\mathbf{R}$ is a sequence $(R_1^{\mathcal{C}}, \ldots, R_k^{\mathcal{C}})$, where each $R_i^{\mathcal{C}}$ is a relation of the same arity as R_i. R_i is used to denote both the relation symbol and the relation $R_i^{\mathcal{C}}$ that interprets it. An *atom* (over $\mathbf{R}$) is a formula $P(x_1, \ldots, x_m)$, where P is a relation symbol in $\mathbf{R}$ and $x_1, \ldots, x_m$ are variables, not necessarily distinct. A fact of an instance $\mathcal{C}$ (over $\mathbf{R}$) is an expression $P^{\mathcal{C}}(v_1, \ldots, v_m)$, where P is a relation symbol in $\mathbf{R}$ and $v_1, \ldots, v_m$ are values such that $(v_1, \ldots, v_m) \in P^{\mathcal{C}}$. All instances $\mathcal{C}$ are considered to be finite, which means that every relation $R_i^{\mathcal{C}}$ is finite, for $1 \leq i \leq k$.

Sometimes rules are used as a part of schema to ensure accuracy and consistency of data in a relational database. These rules are called *integrity constraints*. An instance $\mathcal{C}$ of a schema $\mathbf{R}$ is said to be *legal* wrt $\mathbf{R}$ if $\mathcal{C}$ satisfies the integrity constraints of $\mathbf{R}$.
The basic relational building block is the *domain* or *data type*. A *tuple* is an ordered set of *attribute values*. An *attribute* is an ordered pair of *attribute name* and *data type name*. An *attribute value* is a specific valid value for the data type of the attribute. $t(A)$ denotes the attribute value of the attribute A in the tuple t.

Nulls.The term *null* is used to mean 'value exists but unknown'. Usually the symbol $\perp$ is used to denote a null. Since the target instance may contain null variables, we review

some notations regarding tuples, relations and database instances with null variables in the following.

A tuple containing 0 or more nulls is *partial*. A tuple with no nulls is *total*. Thus, every total tuple is a partial tuple. A tuple t whose schema includes attribute A is definite on A, written $t(A)\downarrow$, if $t(A)$ is not null. This notation extends to sets of attributes: $t(X)\downarrow$ means $t(A)\downarrow$ for every attribute $A \in X$. $t\downarrow$ means that t is total. If t is a tuple over schema R, then $DEF(t) = \{A \in R | t(A)\downarrow\}$ and $NDEF(t) = \{B \in R | t(B) \in \textbf{Null}\ \}$. For tuples t and u on the same scheme, t *subsumes* u, written $t \geq u$, if $u(A)\downarrow$ implies $u(A) = t(A)$ for every attribute A in R. We write $t \simeq u$ when $t \geq u$ and $u \geq t$. Note that, $t \simeq u$ expresses the fact that t and u are equivalent up to the renaming of the null variables in them. If $t \geq u$ and $t\downarrow$, then t is called an extension of u, written $t\downarrow\geq u$.

A relation r is total, written $r\downarrow$, if all of its tuples are total. Relations containing 0 or more nulls are partial. For relation scheme R, we let $Rel\uparrow(R)$ be the set of all partial relations over R and let $Rel(R)$ be the set of all total relations over R. For relations r and s over R, r *subsumes* s, written $r \geq s$, if for every tuple $t_s \in s$ there is a tuple $t_r \in r$ such that $t_r \geq t_s$. If $r \geq s$ and $s \geq r$, we write $r \simeq s$. If r is total, then r is an extension of s, written $r\downarrow\geq s$. If r can be obtained from s by changing some nulls in s to values, then r *augments* s, written $r \succeq s$. Clearly, $r \succeq s$ implies $r \geq s$. If r is total, then r *completes* s, or r is a completion of s, written $r\downarrow\succeq s$. If $r \succeq s$ and $s \succeq r$ then $r = s$. Evidently, r augments s if there is a mapping α of the tuples of s onto the tuples of r such that $\alpha(t) \geq t$ for every tuple $t \in s$.

An *instance* $\mathcal{B}$ of a schema $\mathcal{G}$ is called *total* if all relation $r \in \mathcal{B}$, $r\downarrow$. Otherwise the instance is called *partial*. For two instances $\mathcal{B}$ and $\mathcal{B}'$, $\mathcal{B}' \geq \mathcal{B}$ expresses the fact that for all relations $r \in \mathcal{B}$ there is a relation $r' \in \mathcal{B}'$ such that $r' \geq r$.

Example 1 *Let us consider the tuples t_1, t_2 and t_3 of Figure 2.1. Now, all the tuples are partial and t_2 is total. We have, $t_1(D)\downarrow$, $DEF(t_1) = \{D, MN\}$ and $NDEF(t_1) = \{M\}$. We also have $t_2 \geq t_1$, $t_2\downarrow\geq t_1$ and $t_1 \simeq t_3$.* □

Homomorphism. Let **Const** be a fixed infinite set of constants and **Null** be a fixed

infinite set of nulls that is disjoint from **Const**t. Also, let $\mathcal{C}$ and $\mathcal{C}'$ be two instances over a schema $\mathcal{S}$. A function h from **Const** $\cup$ **Null** to **Const** $\cup$ **Null** is a homomorphism from $\mathcal{C}$ to $\mathcal{C}'$ if for every $c \in$ **Const**, we have that $h(c) = c$, and for every relation symbol R in $\mathcal{S}$ and every tuple $(a_1, \ldots, a_n) \in R^{\mathcal{C}}$, we have that $(h(a_1), \ldots, h(a_n)) \in R^{\mathcal{C}'}$. $\mathcal{C} \to \mathcal{C}'$ denotes that there is a homomorphism from $\mathcal{C}$ to $\mathcal{C}'$. The instances $\mathcal{C}$ and $\mathcal{C}'$ are said to be *homomorphically equivalent*, denoted by $\mathcal{C} \leftrightarrow \mathcal{C}'$, if $\mathcal{C} \to \mathcal{C}'$ and $\mathcal{C}' \to \mathcal{C}$.

2.2 Schema Mapping

Schema mappings are specifications that describe the relationships between schemas at a high level. These specifications are typically given in a logical formalism that captures the interaction between schemas at a logical level without spelling out implementation details relevant to the physical level. Schema mappings are widely used in all data management applications that involve data sharing or data transformation. In particular, schema mappings are essential building blocks in information integration, data exchange, and peer-to-peer data management systems. These three systems share several ideas, concepts, and techniques, but the corresponding communities have stayed relatively distant from one another.

Let $\mathcal{S}$ and $\mathcal{G}$ be two schemas with no relation symbols in common. A schema mapping is a triple $\mathcal{M} = (\mathcal{S}, \mathcal{G}, \Sigma)$ such that Σ is a set of formulas of some logic $\mathcal{L}$ over $\langle \mathcal{S}, \mathcal{G} \rangle$. In such a schema mapping, $\mathcal{S}$ is called the *source schema*, and $\mathcal{G}$ is called the *global schema* (or *target schema*).

Global-as-view (GAV), *local-as-view* (LAV) and *global-and-local-as-view* (GLAV) are the three approaches for specifying schema mappings.

GAV approach is global-schema-centric and specifies the mappings by a set of assertions of the form $\forall \vec{x}(\phi_{\mathcal{S}}(\vec{x}) \rightsquigarrow g(\vec{x}))$ where g is an element of global schema $\mathcal{G}$ and $\phi_{\mathcal{S}}$ is

a query over source schema $\mathcal{S}$. Relations in $\mathcal{G}$ are views and queries are expressed over the views. Queries can simply be evaluated over the data satisfying the global relations.

LAV approach is source-schema-centric and specifies the mappings by a set of assertions of the form $\forall\vec{x}(\phi_{\mathcal{G}}(\vec{x}) \rightsquigarrow s(\vec{x}))$ where s is an element of $\mathcal{S}$ and $\phi_{\mathcal{G}}$ is a query over $\mathcal{G}$. Sources are views and the queries are answered on the basis of the available data in the views.

GLAV is a mixed approach and specifies the mappings by a set of assertions of the form $\forall\vec{x}(\exists\vec{y}\phi_{\mathcal{S}}(\vec{x},\vec{y}) \rightsquigarrow \exists\vec{z}\phi_{\mathcal{G}}(\vec{x},\vec{z}))$ where $\phi_{\mathcal{G}}$ is a query over $\mathcal{G}$ and $\phi_{\mathcal{S}}$ is a query over $\mathcal{S}$. To answer a query over $\mathcal{G}$, one has to infer how to use the mappings in order to access the source instance.

Metadata is utilized to improve communication between heterogeneous information systems [10, 11, 53]. Schema mappings are metadata. Bernstein [10] treated schema mappings as objects and defined operators on schema mappings such as *composition*, *merge*, *match* and *change*. Progress has been made in the study of sevaral operators, including composition [24, 49, 43] and merge [50]. Development of techniques for transforming schema mappings to 'equivalent' ones that are more manageable from the standpoint of data integration, data exchange and P2P data sharing bear great importance. Fagin et al. presented the formalism for defining 'equivalences' between schema mappings in the context of data exchange in [26]. This work is regarded as the pioneer work in the direction of schema mapping optimization. In [16] a technique is introduced for testing whether a schema mapping is contained by another schema mapping. This technique is based on the existence of a homomorphism between special 'dummy' instances, which are built from schema mappings.

In this thesis, we characterize the notion of equivalence between schema mappings of two data integration systems and also propose an algorithm to optimize a schema

mapping to an equivalent simpler one.

2.3 Data Integration

Data integration is the problem of combining data residing at different sources, and providing the user with a unified view of these data [31, 64, 36]. The data integration systems are characterized by an architecture based on a global schema and a set of sources. The sources contain the real data, while the global schema provides a reconciled, integrated, and virtual view of the underlying sources. Modeling the relation between the sources and the global schema is therefore a crucial aspect. Three basic approaches have been proposed to this purpose. The first approach, called global-as-view, requires that the global schema is expressed in terms of the data sources. Most of current data integration systems follow this approach. Notable examples are [29], Garlic [19], COIN [30], MOMIS [9], Squirrel [65], and IBIS [15]. The second approach, called local-as-view [46, 40], requires the global schema to be specified independently from the sources, and the relationships between the global schema and the sources are established by defining every source as a view over the global schema. In the third approach, called global-and-local-as-view [28] the relationships between the global schema and the sources are established by making use of both local-as-view and global-as-view assertions.

Irrespective of the method used for the specification of the mapping between the global schema and the sources, one basic service provided by the data integration system is to answer queries posed in terms of the global schema. Given the architecture of the system, query processing in data integration requires a reformulation step: the query over the global schema has to be reformulated in terms of a set of queries over the sources.

Since sources are in general autonomous, in many real-world applications the problem arises of mutually inconsistent data sources. In practice, this problem is generally

dealt with by means of suitable transformation and cleaning procedures applied to data retrieved from the sources.

The goal of data integration is to provide a uniform interface for querying a collection of disparate and heterogeneous data sources [42]. Several aspects of data integration like, background theory [1, 32, 42], query answering [31, 32, 51], query rewriting [18], query plans [22, 23], query containment [48] and consistency [13] have thoroughly been investigated. However, the optimization of the schema mappings of data integration systems has not been addressed much which we do in this thesis.

2.4 Data Exchange

Data exchange, also known as data translation, is the problem of taking data structured under a schema, called the *source schema*, and transforming it into data structured under another schema, called the *target schema*. In [10] data exchange has been described as the oldest database problem. EXPRESS system is one of the earliest low-level tool for data exchange between hierarchical databases, developed in the 1970s [61]. Data exchange has been a recurrent problem that has taken a new significance with the advent of semi-structured data and the resulting need to exchange data between heterogeneous schemas.

There are obvious similarities, but also clear differences, between data integration and data exchange. In both frameworks, schema mappings are used to specify the relationships between the schemas involved. In data integration, the goal is to synthesize data from different sources into a unified view under a global schema; this view is virtual, in that the data remain in the sources and are accessed by users symbolically via the global schema. In data exchange, the goal is to take a given source instance and transform it to a target instance such that it satisfies the specifications of the schema mapping and also reflects the given source data as accurately as possible; unlike data integration, this target instance is a materialized instance, not a virtual view.

Consider a schema mapping between a source schema and a target schema. It is often the case that, given a source instance, there may be multiple target instances, called *solutions*, that satisfy the specifications of the schema mapping under consideration. The challenge in data exchange arises because typically there exists more than one solution that satisfy the specifications of the schema mapping [25].
Both data integration and data exchange use the concept of the certain answers as the standard semantics of query answering, a concept that originated in the study of incomplete databases [47]. The two frameworks, however, adopt different approaches to obtain the certain answers of queries. In data integration, queries posed against the global schema are usually processed via rewriting to queries posed against the source schemas. In data exchange, however, it is natural to try to process target queries by making use of the materialized target instance; furthermore, this may be the only reasonable approach in cases in which the source instance becomes inaccessible after the exchange has taken place. In turn, this raises the question: for which target queries can the certain answers be obtained by evaluating them on a good solution [41]?

Some of the semantic and algorithmic issues related to data exchange problem are (a) given a source instance, which solutions are better than others? (b) which solution should one choose to materialize? (c) how difficult is to compute such a good solution? (d) what are the semantics of target queries and how difficult is it to evaluate such queries? These issues were addressed in [6, 21, 25, 27].

In this thesis, we express the mappings and constraints of data exchange system by tabular representation and show how this helps to find a “most general” and “minimal” solution.

2.5 Peer-to-Peer Data Sharing

Peer-to-peer data sharing systems (PDMS) are natural extension of integrated information system. In a conventional data integration system which manages an integrated schema, distributes queries to appropriate sources, and integrates incoming data to a common result. In contrast to that, a PDMS consists of a set of peers, each of which can play the role of an integrating component. A peer knows about its neighboring peers by pairwise schema mappings, which help to translate queries and transform data. Queries submitted to one peer are answered by data residing at that peer and by data that is reached along paths of mappings through the network of peers [35].

In the last few years, steady progress has been made in research on various issues related to peer data management systems. The vision of PDMS was first introduced in [12]. Theoretical foundation of PDMS was given in [17]. Piazza [33], Hyperion [7] and PeerDB [62] are some of the proposed framework for PDMS. These systems combine both P2P and database management system functionalities. The local databases on peers are called peer databases. Each peer chooses its own database schema and maintains data independently. The data is accessed globally from any peer by traversing the network of peers.

Contrary to the traditional data integration settings where a global mediated schema is required for data exchange, in a data sharing setting, semantic relationships that exist between peers are exploited for sharing data. When a request (query or update) is posed to a peer, it is executed in the system by running into semantically related peers. Moreover, in a typical data integration and a data exchange setting a single domain is dealt with, for example, a set of sources all containing information about videos or music files. Hence, data integration or data exchange between data sources is provided mainly through the use of views or schema mappings, i.e., queries that map

and restructure data between schemas. However, in a data sharing setting, sources may represent different worlds with different schemas and the real world entities denoted by a symbol in different sources may be related [12]. Moreover, the symbols representing the entities in two sources can be represented by two different vocabularies since sources are designed independently. In order to represent this situation, one may use a global schema with appropriate mappings to/from each local source schema. However, building a global schema is not feasible in a P2P setting since (i) P2P networks are open-ended and continuously evolve, (ii) it requires huge effort and time to build a global schema for large number of peers, and (iii) it is not practical to build a global schema for every peer and her acquaintances as acquaintances keep changing [60]. Instead, a solution is adopted to this problem by creating a domain relation between sources through pairwise schema-level and data-level mappings. The mappings map the data elements of a source domain to the data elements of another source domain. In order to create a domain relation, value correspondences between sources are created through the use of mapping tables [39]. The schema mappings between sources are established through the use of GLAV mappings.

Some of the issues related to peer data management systems that attained research interest are data integration models [34], mediation methods [33], coordination mechanisms [60], and data-level mappings [39, 52] among the peer databases. In this thesis we introduce a mapping that simultaneously resolve schema-level and data-level heterogeneity in a peer-to-peer data sharing system.

2.6 Tableaux

A *tableau* is a two-dimensional representation for Selection-Projection-Join (SPJ) expressions [5, 4]. Tableaux may be considered a stylized notation for a subset of Zloofs Query-by-Example language [66] and Chandra and Merlins conjunctive queries [20]. Aho et al. [2] reduced the equivalence and optimization problems for queries to the analogous

problems for tableaux. The tableau approach allows to deal with functional dependencies mechanically, an advantage not possessed by more direct techniques. They present an algorithm for minimizing the number of rows in a tableau, an operation that corresponds to minimizing the number of joins needed to evaluate an SPJ-expression. Since join is typically a very expensive operator to implement, this optimization is quite desirable in query evaluation. Row minimization also serves to eliminate common subexpressions from a query. Some results and applications of tableaux are contained in [2, 3, 8, 14, 45, 59].

The mechanism for expressing conjunctive queries by tableaux is described below.

Let $\{\vec{x} \mid \forall\vec{x}\exists\vec{y}\,\varphi(\vec{x},\vec{y})\}$ be a conjunctive query , where $\vec{x} = (x_1, \ldots, x_n)$, $\vec{y} = (y_1, \ldots, y_m)$, and $\varphi(\vec{x},\vec{y})$ is a conjunction $C_1 \wedge \ldots \wedge C_k$. Each C_i can be of the form

- $R(c_1, \ldots, c_r)$ with the c_i being x_j and y_k to indicate that the tuple $\langle c_1, \ldots, c_r\rangle$ shall be in a relation R, or
- $c = d$ with c, d are x_i, y_j or constants as selection/join predicates

A *tableau* $\mathcal{T}$ for the query $\{\vec{x} \mid \forall\vec{x}\exists\vec{y}\varphi(\vec{x},\vec{y})\}$ is a tabular representation having the following components

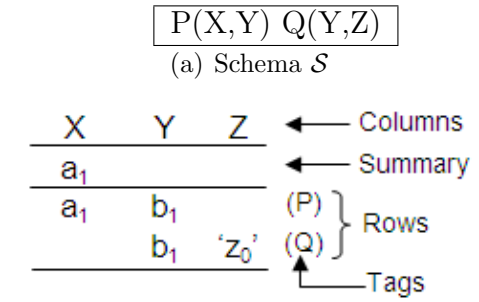
Summary- The first row representing the schema of the resulting tuple corresponding to $\vec{x}$.

Rows- Other rows representing the conjuncts C_i.

Columns- The collection of all attributes of all relations.

Tags- Relation names appended to rows.

Columns not corresponding to attributes of the tag-relations are always blank. Symbols a_k are used to represent each x_i and b_l are used represent each y_j. Constants and a-symbols are called *distinguished* symbols. Blanks and b-symbols are called *non-distinguished* symbols.



(a) Schema $\mathcal{S}$

(b) Tableau T_q for query $q \equiv \{x \mid \forall x \exists y(P(x,y) \wedge Q(y,`z_0'))\}$

P

X	Y
x_1	y_1
x_2	y_1
x_3	y_2

Q

Y	Z
y_1	z_0
y_1	z_1
y_2	z_2

(c) Source instance $\mathcal{C}$

X
x_1
x_2

(d) $\mathcal{T}_q(\mathcal{C})$

Figure 2.2: Example of Tableau

Let $\mathcal{T}$ be a tableau with a summary w_0 and rows $w_1, \ldots, w_n$, and let S be the set of all the nonblank symbols in $\mathcal{T}$. A valuation ρ for $\mathcal{T}$ maps each symbol in S to a constant, such that if c is a constant in S, then $\rho(c) = c$. The valuation ρ is extended to the rows and summary of $\mathcal{T}$ by defining $\rho(w_i)$ to be the result of substituting $\rho(v)$ for every variable v that appears in w_i. The tableau $\mathcal{T}$ defines a mapping from instances to relations on its *target relation scheme*, which is the set of all the attributes corresponding to columns that have a nonblank symbol in the summary. Given an instance $\mathcal{C}$, the value of $\mathcal{T}$, written $\mathcal{T}(\mathcal{C})$, is

$$\mathcal{T}(\mathcal{C}) = \{\rho(w_o) \mid \text{ for some valuation } \rho, \text{ we have } \rho(w_i) \text{ in } \mathcal{C} \text{ for } 1 \leq i \leq n\}$$

A set of tableaux $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ is expressed by the expression $\cup_{i=1}^{n} \mathcal{T}_i$. Result of applying $\cup_{i=1}^{n} \mathcal{T}_i$ to an instance $\mathcal{C}$ is given below:

$$(\cup_{i=1}^{n} \mathcal{T}_i(\mathcal{C})) = \cup_{i=1}^{n} (\mathcal{T}_i(\mathcal{C}))$$

Proposition 1 *For every SPJ query q over a schema S there is an equivalent tableau query T_q and vice versa.*

Theorem 1 *Let T and T' be two tableau queries representing the queries q and q' over the same schema S. Then $q \subseteq q'$ iff there exists a homomorphism from T' to T.*

Example 2 *Consider a schema $\mathcal{S}$ having two relations $P(X, Y)$ and $Q(X, Y)$ as shown in Figure 2.2(a). The tableaux $\mathcal{T}_q$ for the query $q \equiv \{x \mid \forall x \exists y(P(x, y) \wedge Q(y, `z_0'))\}$ on schema S is shown in Figure 2.2(b). For the instance I in Figure 2.2(c) no valuation for $\mathcal{T}_q$ (wrt I) produces a tuple other than $\{x_1\}$ or $\{x_2\}$. So, we have $\mathcal{T}_q(I)$ as shown in Figure 2.2(d).* □

Our motivation for expressing schema mappings by tabular representation came from the idea of expressing queries by tableaux.

2.7 Summary

In this chapter, we first briefly discussed some state of the art research works in the area of schema mapping, data integration, data exchange, data sharing, mapping optimization and tableaux . Next, we presented basic notations and technical preliminaries related to those fields. Notations and terms defined in this chapter will be used in the following chapters of this thesis for presenting our proposed techniques and algorithms.

Chapter 3

Tableaux-Based Schema Mappings in a Data Integration Systems

In this chapter, we present some algorithms to convert GLAV mappings of a Data Integration System into tabular forms called *Mapping Assertion Tableaux* (MAT). We also express TGD and EGD constraints of the global schema of a data integration system by tabular forms called *Tabular TGD* (TTGD) and *Tabular EGD* (TEGD), respectively. We show how these MATs, TTGDs and TEGDs can be used as operators on a source instance to produce a minimal universal model in a data integration system. We also introduce some theories describing the utility of using tabular forms of mappings and constraints.

3.1 Data Integration System

In this section we define some terms that are related to *data integration systems* (DIS).

Data integration systems are based on the following general architecture. The user interacts with a uniform interface in the form of a set of *global relation names* that are used in formulating queries. These relations are called the *global schema*. The actual

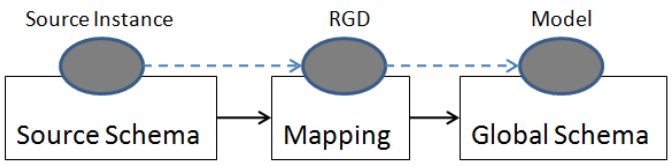


Figure 3.1: Data integration problem

data is stored in external sources, called the source relations. The schema of the source relations is called *source schema*. In order for the system to be able to answer queries, mappings must be specified between the relations in the global schema and the source schema. A set of GLAV mapping assertions are commonly used to specify these mappings.

Formally, a data integration system is a triple $\mathcal{I} = (\mathcal{G}, \mathcal{S}, \mathcal{M})$ where $\mathcal{G}$ is the global schema, $\mathcal{S}$ is the source schema and $\mathcal{M}$ is the mapping between $\mathcal{G}$ and $\mathcal{S}$. A global instance $\mathcal{B}$ is called *retrieved global database* (RGD) for $\mathcal{I}$ wrt a source database $\mathcal{C}$ of $\mathcal{S}$ if $\mathcal{B}$ and $\mathcal{C}$ together satisfies $\mathcal{M}$, denoted by $(\mathcal{B}, \mathcal{C}) \models \mathcal{M}$. an RGD may be partial (i.e. it may have relations containing nulls). Semantics of a data integration system $\mathcal{I} = (\mathcal{G}, \mathcal{S}, \mathcal{M})$ wrt a source database $\mathcal{C}$, as depicted in Figure 3.1, is given in terms of a set of models that satisfy $\mathcal{M}$ together with $\mathcal{C}$ and also satisfies the integrity constraints of $\mathcal{G}$. Formally,

$$sem^{\mathcal{C}}(\mathcal{I}) = \{\mathcal{A} | \mathcal{A} \text{ is an RGD of } \mathcal{I} \text{ wrt } \mathcal{C} \text{ and also legal wrt } \mathcal{G}\}.$$

Each database $\mathcal{A} \in sem^{\mathcal{C}}(\mathcal{I})$ is called a *model* of $\mathcal{I}$ wrt $\mathcal{C}$.

In the following we define some terms related to DIS which we will use in the following chapters for expressing the semantics of tabular schema mappings.

Global Constraints. We assume that there are two types of constraints in the global schema: *tuple generating dependency* (TGD) and *equality generating dependency* (EGD). TGD is a constraint of the form

$$\forall \vec{x}(\exists \vec{y}\varphi(\vec{x},\vec{y}) \rightarrow \exists \vec{z}\psi(\vec{x},\vec{z}))$$

where φ and ψ are conjunctions of atoms (in other words join of relations of the global schema) and every variable in $\vec{x}$ occurs in both φ and ψ. These dependencies are also known as *global-and-local-as-view* (GLAV) constraints [42].
An EGD has the form

$$\forall \vec{x}(\varphi(\vec{x}) \rightarrow (x_1 = x_2)),$$

where $\varphi(\vec{x})$ is a conjunction of atomic formulas over the global schema and $x_1 \in \vec{x},\ x_2 \in \vec{x}$.

Universal RGD. For a data integration system $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$ and a source instance $\mathcal{C}$, an RGD $\mathcal{B}$ is said to be a *universal retrieved global database* (URGD) of $\mathcal{I}$ wrt $\mathcal{C}$ if for every RGD $\mathcal{B}'$ of $\mathcal{I}$ wrt $\mathcal{C}$, we have a homomorphism $\mathcal{B} \rightarrow \mathcal{B}'$.

Minimal Universal RGD. For a data integration system $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$ and a source instance $\mathcal{C}$, a URGD $\mathcal{B}$ is said to be a *minimal universal retrieved global database* (MURGD) of $\mathcal{I}$ wrt $\mathcal{C}$ if $\nexists \mathcal{B}''(\mathcal{B}'' \subset \mathcal{B} \wedge \mathcal{B} \rightarrow \mathcal{B}'')$ holds.

Universal Model. For a data integration system $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$ and a source instance $\mathcal{C}$, a model $\mathcal{A}$ is said to be a *universal model* of $\mathcal{I}$ wrt $\mathcal{C}$ if for every model $\mathcal{A}'$ of $\mathcal{I}$ wrt $\mathcal{C}$, we have that $\mathcal{A} \rightarrow \mathcal{A}'$.

Minimal Universal Model. For a data integration system $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$ and a source instance $\mathcal{C}$, a universal model $\mathcal{A}$ is said to be a *minimal universal model* (MUM) of $\mathcal{I}$ wrt $\mathcal{C}$ if $\nexists \mathcal{A}''(\mathcal{A}'' \subset \mathcal{A} \wedge \mathcal{A} \rightarrow \mathcal{A}'')$ holds.

In this chapter, we show that when our proposed tabular representation of schema mapping is used as an operator on a source instance of a data integration system, it outputs a minimal universal model.

Procedure MA2MAT(m)
Input : A mapping assertion $m : \forall\vec{x}(\exists\vec{y}\phi_{\mathcal{S}}(\vec{x},\vec{y}) \rightsquigarrow \exists\vec{z}\phi_{\mathcal{G}}(\vec{x},\vec{z}))$
Output: A MAT $T_{\phi_{\mathcal{S}}}^{\phi_{\mathcal{G}}}$ representing m
begin
Create a tableau $\mathcal{T}_{\phi_{\mathcal{S}}}$ for the expression $\forall\vec{x}\exists\vec{y}\phi_{\mathcal{S}}(\vec{x},\vec{y})$
Delete the summary of $\mathcal{T}_{\phi_{\mathcal{S}}}$
Create a tableau $\mathcal{T}_{\phi_{\mathcal{G}}}$ for the expression $\forall\vec{x}\exists\vec{z}\phi_{\mathcal{G}}(\vec{x},\vec{z})$
Delete the summary of $\mathcal{T}_{\phi_{\mathcal{G}}}$
Modify $\mathcal{T}_{\phi_{\mathcal{G}}}$ as follows:
for each $z_i \in \vec{z}$ **do**
if (z_i is represented by b_j in $\mathcal{T}_{\phi_{\mathcal{G}}}$) **then**
replace b_j by $\perp_j$
endif
endfor
for each $x_i \in \vec{x}$ **do**
if x_i is represented by a_j in $\mathcal{T}_{\phi_{\mathcal{S}}}$ and by a_k in $\mathcal{T}_{\phi_{\mathcal{G}}}$
replace a_k by a_j in $\mathcal{T}_{\phi_{\mathcal{G}}}$
endfor
$\mathcal{T}_{\phi_{\mathcal{S}}}^{\phi_{\mathcal{G}}} \leftarrow \langle \mathcal{T}_{\phi_{\mathcal{S}}}, \mathcal{T}_{\phi_{\mathcal{G}}} \rangle$
return $\mathcal{T}_{\phi_{\mathcal{S}}}^{\phi_{\mathcal{G}}}$
end

Figure 3.2: Procedure MA2MAT()

3.2 Mapping Assertion Tableaux

In this section, we show how to convert a GLAV mapping assertion to a tabular representation called MAT. We also present the semantics of MAT as an operator on a source instance.

Consider a data integration system $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$, where $\mathcal{M} = \{m_1, \ldots, m_r\}$ is a set of GLAV mapping assertions. Let $m : \forall\vec{x}(\exists\vec{y}\phi_{\mathcal{S}}(\vec{x},\vec{y}) \rightsquigarrow \exists\vec{z}\phi_{\mathcal{G}}(\vec{x},\vec{z}))$ be an arbitrary mapping assertion in $\mathcal{M}$ where $\vec{x} = (x_1, \ldots, x_p)$, $\vec{y} = (y_1, \ldots, y_q)$, and $\vec{z} = (z_1, \ldots, z_r)$. Remember that $\phi_{\mathcal{S}}(\vec{x},\vec{y})$ must have relations only from the source schema $\mathcal{S}$ and $\phi_{\mathcal{G}}(\vec{x},\vec{z})$ must have relations only from the global schema $\mathcal{G}$. A MAT $\mathcal{T}_{\phi_{\mathcal{S}}}^{\phi_{\mathcal{G}}}$ for m is a partitioned table whose left hand side partition is a summary-free source tableau $\mathcal{T}_{\phi_{\mathcal{S}}}$ (representing

	D	M	E
	a_1	a_2	a_3
DeptEmp	a_1	a_2	a_3

(a) Initial $\mathcal{T}_{\phi_\mathcal{S}}$

D	M	MN	E	
a_4		a_5	a_6	
a_4	b_1	a_5		Dept
a_4			a_6	Emp

(b) Initial $\mathcal{T}_{\phi_\mathcal{G}}$

	D	M	E
DeptEmp	a_1	a_2	a_3

(c) Modified $\mathcal{T}_{\phi_\mathcal{S}}$

D	M	MN	E	
a_1	$\perp_1$	a_2		Dept
a_1			a_3	Emp

(d) Modified $\mathcal{T}_{\phi_\mathcal{G}}$

	D	M	E	D	M	MN	E	
DeptEmp	a_1	a_2	a_3	a_1	$\perp_1$	a_2		Dept
				a_1			a_3	Emp

(e) MAT $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}} = \langle \mathcal{T}_{\phi_\mathcal{S}}, \mathcal{T}_{\phi_\mathcal{G}} \rangle$

Figure 3.3: Steps for converting the mapping assertion m of Example 3 to a MAT

the query $\{\vec{x} \mid \forall\vec{x}\exists\vec{y}\phi_\mathcal{S}(\vec{x},\vec{y})\}$) and the right hand side partition is a summary-free target tableau $\mathcal{T}_{\phi_\mathcal{G}}$ (representing the query $\{\vec{x} \mid \forall\vec{x}\exists\vec{z}\phi_\mathcal{G}(\vec{x},\vec{z})\}$). Sometimes we express the MAT by a pair $\langle \mathcal{T}_\mathcal{S}, \mathcal{T}_\mathcal{G} \rangle$. By 'summary-free tableau' we mean a structure which is exactly like a tableau except it has no summary row. In the depictive representation of a MAT, the constituent tableaux are separated by a vertical bar. Procedure $MA2MAT()$, as shown in Figure 3.2, converts a given mapping assertion $m : \forall\vec{x}(\exists\vec{y}\phi_\mathcal{S}(\vec{x},\vec{y}) \rightsquigarrow \exists\vec{z}\phi_\mathcal{G}(\vec{x},\vec{z}))$ into a MAT $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}$. In this procedure, two separate tableaux $\mathcal{T}_{\phi_\mathcal{S}}$ and $\mathcal{T}_{\phi_\mathcal{G}}$ are created for the queries $\{\vec{x} \mid \forall\vec{x}\exists\vec{y}\phi_\mathcal{S}(\vec{x},\vec{y})\}$ and $\{\vec{x} \mid \forall\vec{x}\exists\vec{z}\phi_\mathcal{G}(\vec{x},\vec{z})\}$, respectively. Then the summaries of the tableaux $\mathcal{T}_{\phi_\mathcal{S}}$ and $\mathcal{T}_{\phi_\mathcal{G}}$ are discarded and distinguished variables of $\mathcal{T}_{\phi_\mathcal{S}}$ and $\mathcal{T}_{\phi_\mathcal{G}}$ are unified by using the same distinguished variable in both the tableaux for each universally quantified variable in m. Non-distinguished variables of $\mathcal{T}_{\phi_\mathcal{G}}$ are then replaced by labeled nulls. Finally, modified $\mathcal{T}_{\phi_\mathcal{S}}$ and modified $\mathcal{T}_{\phi_\mathcal{G}}$ are merged together into a single structure to form the MAT $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}$.

Note, the algorithm $MA2MAT()$ deletes summaries from both $\mathcal{T}_{\phi_\mathcal{S}}$ and $T_{\phi_\mathcal{G}}$ before combining them to form $T_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}$. $T_{\phi_\mathcal{S}}$ acts as the *body* and $T_{\phi_\mathcal{G}}$ acts as the *summary* for the MAT $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}$. Also note that, the rows of $\mathcal{T}_{\phi_\mathcal{G}}$ contain distinguished variables and null

variables. Each variable in $\vec{x}$ (which are common in both $\phi_{\mathcal{S}}$ and $\phi_{\mathcal{G}}$) are represented by the same distinguished variable in $\mathcal{T}_{\phi_{\mathcal{S}}}$ and $\mathcal{T}_{\phi_{\mathcal{G}}}$.

Example 3 *Consider the data integration system $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$ where the source schema $\mathcal{S}$ has one relation $DeptEmp(D, M, E)$ listing departments with their manager names and their employee ids. The global schema $\mathcal{G}$ has two relations $Dept(D, M, MN)$ and $Emp(E, D)$. $Dept(D, M, MN)$ lists departments, their manager ids and manager names, and $Emp(E, D)$ relates employee ids with their departments. $\mathcal{M}$ contains a single mapping assertion m defined as follows:*

$$m : \forall d \forall n \forall e(DeptEmp(d, n, e) \rightarrow \exists m(Dept(d, m, n) \ \wedge Emp(e, d)))$$

When the algorithm $MA2MAT()$ of Figure 3.2 is applied on m, initially it creates two tableaux $\mathcal{T}_{\phi_{\mathcal{S}}}$ and $\mathcal{T}_{\phi_{\mathcal{G}}}$ as shown in Figure 3.3(a) and Figure 3.3(b), respectively.

After deleting the summary of $\mathcal{T}_{\phi_{\mathcal{S}}}$ and $\mathcal{T}_{\phi_{\mathcal{G}}}$, replacing nondistinguished variables of $\mathcal{T}_{\phi_{\mathcal{G}}}$ with null variables, and unifying the distinguished variables of $\mathcal{T}_{\phi_{\mathcal{S}}}$ and $\mathcal{T}_{\phi_{\mathcal{G}}}$, we get modified $\mathcal{T}_{\phi_{\mathcal{S}}}$ and $\mathcal{T}_{\phi_{\mathcal{G}}}$ as shown in Figure 3.3(c) and Figure 3.3(d), respectively. Finally merging modified $\mathcal{T}_{\phi_{\mathcal{S}}}$ and $\mathcal{T}_{\phi_{\mathcal{G}}}$, we get the MAT $\mathcal{T}_{\phi_{\mathcal{S}}}^{\phi_{\mathcal{G}}}$ as shown in Figure 3.3(e). □

In Definition 1 we define the valuation function ρ.

Definition 1 *A valuation ρ is a function that maps each constant value to itself and each variable v of a tableau to the value in the intersection of the domains of the attributes where v appear and each null variable $\perp_i$ to a fresh labeled null. When more than one row in $\mathcal{T}_{\phi_{\mathcal{G}}}$ share the same $\perp_i$, ρ maps $\perp_i$ to the same null variable $\perp_j$ for each row sharing $\perp_i$. Let $\mathcal{T}_{\phi_{\mathcal{S}}}^{\phi_{\mathcal{G}}} = \langle \mathcal{T}_{\phi_{\mathcal{S}}}, \mathcal{T}_{\phi_{\mathcal{G}}} \rangle$ be a MAT. $\mathcal{T}_{\phi_{\mathcal{S}}}$ has the rows $w_1^{\phi_{\mathcal{S}}}, \ldots, w_m^{\phi_{\mathcal{S}}}$ and $\mathcal{T}_{\phi_{\mathcal{G}}}$ has the rows $w_1^{\phi_{\mathcal{G}}}, \ldots, w_n^{\phi_{\mathcal{G}}}$. For a valuation ρ, $\rho(\mathcal{T}_{\phi_{\mathcal{S}}})$ returns a tuple for each relation represented by a row in $\mathcal{T}_{\phi_{\mathcal{S}}}$ by applying ρ to the row. $\rho(\mathcal{T}_{\phi_{\mathcal{G}}})$ also has the same interpretation. Formally,*

$$\rho(\mathcal{T}_{\phi_{\mathcal{S}}}) = \{\rho(w_1^{\phi_{\mathcal{S}}}), \ldots, \rho(w_m^{\phi_{\mathcal{S}}})\}$$

$$\rho(\mathcal{T}_{\phi_{\mathcal{G}}}) = \{\rho(w_1^{\phi_{\mathcal{G}}}), \ldots, \rho(w_n^{\phi_{\mathcal{G}}})\}$$

DeptEmp

D	M	E
CS	$Mary$	$E003$

(a) Source instance $\mathcal{C}$

Dept

D	M	MN
CS	$\perp_{1_1}$	$Mary$

Emp

E	D
$E003$	CS

(b) RGD $\mathcal{B} = \mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}(\mathcal{C})$

Figure 3.4: Example of a Source Instance and its RGD

In Definition 2 we define the semantics of a MAT $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}$ as an operator on an instance $\mathcal{C}$ of a source schema $\mathcal{S}$.

Definition 2 *Given a source instance $\mathcal{C}$, the value of the MAT $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}} = \langle \mathcal{T}_{\phi_\mathcal{S}}, \mathcal{T}_{\phi_\mathcal{G}} \rangle$ wrt $\mathcal{C}$, written as $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}(\mathcal{C})$, is a set of tuples obtained by applying some valuation function ρ on the target tableau $\mathcal{T}_{\phi_\mathcal{G}}$ such that ρ produces some tuples of the source instance $\mathcal{C}$ while applied on the source tableau $\mathcal{T}_{\phi_\mathcal{S}}$. Formally,*

$\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}(\mathcal{C}) = \{\rho(\mathcal{T}_{\phi_\mathcal{G}}) |$ *for some valuation* ρ, $\rho(\mathcal{T}_{\phi_\mathcal{S}}) \in \mathcal{C}\}$

Example 4 *Consider a source instance $\mathcal{C}$ of $\mathcal{S}$ as shown in 3.4(a). When we apply the MAT $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}$ of Figure 3.3(e) to $\mathcal{C}$, we get the RGD $\mathcal{B}$ as shown in Figure 3.4(b) by a valuation ρ where, $\rho(a_1) = CS$, $\rho(a_2) = Mary$, $\rho(a_3) = E003$, and $\rho(\perp_1) = \perp_{1_1}$.* □

Proposition 2 *Consider a data integration system $\mathcal{I} = (\mathcal{G}, \mathcal{S}, \mathcal{M})$. Let $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}} = \langle \mathcal{T}_{\phi_\mathcal{S}}, \mathcal{T}_{\phi_\mathcal{G}} \rangle = MA2MAT(m)$ where $m \in \mathcal{M}$. For an instance $\mathcal{C}$ of $\mathcal{S}$ and a valuation ρ, whenever $\rho(\mathcal{T}_{\phi_\mathcal{S}}) \subseteq \mathcal{C}$ holds, then we have $\rho(\mathcal{T}_{\phi_\mathcal{G}}) \equiv \mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}(\mathcal{C})$.*

Proof: This result can directly be derived from the definition of $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}(\mathcal{C})$. □

In Theorem 2 we show that a MAT $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}$ produces an RGD when it is applied on a source instance $\mathcal{C}$.

Theorem 2 *Let $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$ be a data integration system and $m \in \mathcal{M}$ be a mapping assertion of the form $\forall \vec{x}(\exists \vec{y} \phi_\mathcal{S}(\vec{x}, \vec{y}) \rightsquigarrow \exists \vec{z} \phi_\mathcal{G}(\vec{x}, \vec{z}))$. For any source instance $\mathcal{C}$ of $\mathcal{S}$, we*

have $(\mathcal{C}, \mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}(\mathcal{C})) \models m$*, where* $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}$ *is obtained by applying the procedure* $MA2MAT()$ *on* m*, i.e.* $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}} = MA2MAT(m)$.

Proof: According to the MAT construction rules of the procedure $MA2MAT$, in $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}} = \langle \mathcal{T}_{\phi_\mathcal{S}}, \mathcal{T}_{\phi_\mathcal{G}} \rangle$, both $\mathcal{T}_{\phi_\mathcal{S}}$ and $\mathcal{T}_{\phi_\mathcal{G}}$ have the same distinguished variable for representing $x_i \in \vec{x}$. So, for any valuation ρ, $\rho(\mathcal{T}_{\phi_\mathcal{S}})$ and $\rho(\mathcal{T}_{\phi_\mathcal{G}})$ will have the same value for the $\vec{x}$ components. Now, the $\vec{z}$ components are represented by null variables and $\vec{y}$ components are represented by nondistinguished variables. So, they have no effect on each other. To prove Theorem 2, it is enough to prove that whenever $\phi_\mathcal{S}$ is satisfied by some tuples in $\mathcal{C}$, $\phi_\mathcal{G}$ is also satisfied by some tuples in $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}(\mathcal{C})$. Now, let $\mathcal{C}$ satisfies $\phi_\mathcal{S}$. In this case for some valuation ρ, $\rho(\mathcal{T}_{\phi_\mathcal{S}}) \subseteq \mathcal{C}$ must hold. From Proposition 2, we get $\rho(\mathcal{T}_{\phi_\mathcal{G}}) \equiv \mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}(\mathcal{C})$. Since, $\rho(\mathcal{T}_{\phi_\mathcal{G}})$ satisfies $\phi_\mathcal{G}$, $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}(\mathcal{C})$ must satisfy $\phi_\mathcal{G}$. Thus we prove the theorem. □

Definition 3 gives the semantics of a set of MAT as an operator on a source instance.

Definition 3 *Consider a data integration system* $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$*, where* $\mathcal{M} = \{m_1 \ldots m_r\}$*. Let* $\mathcal{T}_\mathcal{M} = \{\mathcal{T}_1 \ldots \mathcal{T}_r\}$ *be a set of MATs, where* $\mathcal{T}_i = MA2MAT(m_i)$*. Now, if* $\mathcal{C}$ *is an instance of* $\mathcal{S}$*,* $\mathcal{T}_\mathcal{M}(\mathcal{C})$ *denotes the union of all the tuples obtained by applying each* $\mathcal{T}_i \in \mathcal{T}_\mathcal{M}$ *on* $\mathcal{C}$ *for* $1 \leq i \leq r$*. Formally,*

$$\mathcal{T}_\mathcal{M}(\mathcal{C}) = \cup_{i=1}^{r} \mathcal{T}_i(\mathcal{C})$$

In Theorem 3 we claim that the set of MAT $\mathcal{T}_\mathcal{M}$ corresponding to a set of mapping assertions $\mathcal{M}$ can be used as an operator on a source instance $\mathcal{C}$ to generate a universal retrieved global database (URGD).

Theorem 3 *Let* $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$ *be a data integration system and* $\mathcal{T}_\mathcal{M}$ *be the set of MATs for* $\mathcal{M}$*. If* $\mathcal{C}$ *is an instance of* $\mathcal{S}$ *and* $\mathcal{B} = \mathcal{T}_\mathcal{M}(\mathcal{C})$*, then*
(1) $\mathcal{B}$ *is an RGD for* $\mathcal{I}$ *wrt* $\mathcal{C}$*.*
(2) For any other RGD $\mathcal{B}'$ *for* $\mathcal{I}$ *wrt* $\mathcal{C}$*, there is a homomorphism* $\mathcal{B} \rightarrow \mathcal{B}'$*.*

Proof.(1) For $\mathcal{B}$ to be an RGD of $\mathcal{I}$ wrt $\mathcal{C}$, $\mathcal{C}$ and $\mathcal{B}$ together have to satisfy all the mappings of $\mathcal{M}$, i.e. $(\mathcal{C}, \mathcal{B}) \models \mathcal{M}$ should hold. Now, let $\mathcal{M} = \{m_1, \ldots, m_r\}$ and

$\mathcal{T}_{\mathcal{M}} = \{\mathcal{T}_1, \ldots, \mathcal{T}_r\}$, where $\mathcal{T}_i = MA2MAT(m_i)$. According to Theorem 2 we have that $(\mathcal{C}, \mathcal{T}_i(\mathcal{C})) \models m_i$ for $1 \leq i \leq r$. Combining the statement for all m_i together, we have

$$\begin{aligned} & (\mathcal{C}, \cup_{i=1}^{r} \mathcal{T}_i(\mathcal{C})) \models \{m_1, \ldots, m_r\} \\ \Rightarrow\ & (\mathcal{C}, \mathcal{T}_{\mathcal{M}}(\mathcal{C})) \models \mathcal{M} \text{ (from definition 3)} \\ \Rightarrow\ & (\mathcal{C}, \mathcal{B}) \models \mathcal{M} \end{aligned}$$

(2) Consider an arbitrary $m_i \in \mathcal{M}$ where $m_i \equiv \forall \vec{x}(\exists \vec{y} \phi_{\mathcal{S}_i}(\vec{x}, \vec{y}) \rightsquigarrow \exists \vec{z} \phi_{\mathcal{G}_i}(\vec{x}, \vec{z}))$. If $\mathcal{B}'$ is an RGD of $\mathcal{I}$ wrt $\mathcal{C}$, we have that $(\mathcal{C}, \mathcal{B}') \models \mathcal{M}$. Since the source instance $\mathcal{C}$ consists of constants only, $\vec{x}$ components must be constants in $\mathcal{B}'$. This is also the case for $\mathcal{B}$ according to the construction rules of the procedure $MA2MAT()$. So, we have a homomorphism $h_i : \mathcal{B} \rightarrow \mathcal{B}'$, where $h_i(c) = c$ for all $\vec{x}$ components. All $\vec{z}$ components are labeled nulls in $\mathcal{B}$. On the other hand, the $\vec{z}$ components of $\mathcal{B}'$ would either be a constant or a labeled null. Let $z_l \in \vec{z}$ component is represented by $\perp_q$ in $\mathcal{B}$. For $\mathcal{B}'$ we have the following two cases.

Case 1: z_l is represented by a constant c_n. In this case, extend h_i by adding $h_i(\perp_q) = c_n$.

Case 2: z_l is represented by a labeled null $\perp_o$. In this case, extend h_i by adding $h_i(\perp_q) = \perp_o$. This completes the homomorphism $h_i : \mathcal{B} \rightarrow \mathcal{B}'$ for the tuples of $\mathcal{B}$ that come from the valuation of $\mathcal{T}_{\phi_{\mathcal{G}_i}}$. In this way, we can get homomorphisms $\{h_j : 1 \leq j \leq r\}$ for the mapping assertions $\{m_j : 1 \leq j \leq r\}$. Since, fresh null variables are used whenever a valuation is applied to a target tableau, there should not be any conflict among the homomorphisms $\{h_j : 1 \leq j \leq r\}$ and we can take union of them to build a new homomorphism. Let $h = \cup_{i=1}^{r} h_i$. Now, the homomorphism $h : \mathcal{B} \rightarrow \mathcal{B}'$ maps each element of $\mathcal{B}$ to some elements of $\mathcal{B}'$. □

In Theorem 3 we have shown that a set of MAT generated by the procedure $MA2MAT()$ produces URGD for a given source instance. But there is no guarantee that the URGD produced by the set of MAT is a minimal one (i.e. MURGD). To convert the URGD into a MURGD we introduce an algorithm $ReduceBy^{\simeq}$ in Figure 3.5 that takes a URGD as input and converts it into a MURGD. The algorithm $ReduceBy^{\simeq}$ uses a procedure $Update(\mathcal{B}, V_{old}, V_{new})$ that updates $\mathcal{B}$ by replacing V_{old} with V_{new} in all the tuples of all

```
Procedure ReduceBy≃(B)
Input : A URGD B.
Output:An MURGD B' ⊆ B obtained by deleting
        some tuple t ∈ B, such that ∃t'(t' ∈ B ∧ t ≃ t')
begin
    for each r ∈ B do
      for each t ∈ r do
        for each t' ∈ r do
          if (t ≠ t' ∧ t ≃ t') then
            X ← NDEF(t')
            //NDEF returns the set of attributes containing nulls
            for each A ∈ X do
              B ← Update(B, t'(A), t(A))
            endfor
            delete t' from r
          endif
        endfor
      endfor
    endfor
    return B
end
```

Figure 3.5: Procedure $ReduceBy^{\simeq}$

the relations of $\mathcal{B}$. Procedure *Update* is shown in Figure 3.6. Note, V_{old} must be a null variable and V_{new} may either be a constant or a null. If V_{old} is a constant then $Update(\mathcal{B}, V_{old}, V_{new})$ returns an empty instance.

Now, in theorem 4 we claim that the algorithm $ReduceBy^{\simeq}$ can be used along with the set of MAT to produce an MURGD for a given source instance.

Theorem 4 *Let $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$ be a data integration system and $\mathcal{C}$ is an instance of $\mathcal{S}$. If $\mathcal{B} = \mathcal{T}_{\mathcal{M}}(\mathcal{C})$ and $\mathcal{B}' = ReduceBy^{\simeq}(\mathcal{B})$ then $\mathcal{B}'$ is a MURGD for $\mathcal{I}$ wrt $\mathcal{C}$, where the procedure $ReduceBy^{\simeq}()$ is shown in Figure 3.5.*

Proof: In theorem 3 we proved that $\mathcal{B}$ is an RGD having homomorphism to all other RGD of $\mathcal{I}$ wrt $\mathcal{C}$. But $\mathcal{B}$ may contain a group of tuples which are equivalent up to the renaming of the nulls. $ReduceBy^{\simeq}$ eliminates such redundancy by keeping a single

Procedure $Update(\mathcal{B}, V_{old}, V_{new})$
Input : A database instance $\mathcal{B}$, two values V_{old} and V_{new}.
Output: An updated $\mathcal{B}$ such that the value V_{old}
in $\mathcal{B}$ is replaced by V_{new} in all the relation.
begin
if $IsNull(V_{old})$ **then**
for each $r \in \mathcal{B}$ **do**
replace V_{old} by V_{new} in r
endfor
else
$\mathcal{B} \leftarrow EmptyDatabase()$
endif
return $\mathcal{B}$
end

Figure 3.6: Procedure *Update*

tuple representing the group and deleting the rest. Let t and t' be two tuples which are equivalent up to the renaming of the nulls. Before deleting t' from $\mathcal{B}$, if any other tuple t'' of $\mathcal{B}$ has some common null variable as t' in the attribute A then $t''[A]$ is replaced by the value $t[A]$. This ensures that $\mathcal{B}'$ does not violate the two properties in theorem 3 possessed by $\mathcal{B}$. So, $\mathcal{B}'$ is the smallest RGD of $\mathcal{I}$ wrt $\mathcal{C}$ having homomorphism to other RGD. i.e. $\mathcal{B}'$ is a MURGD of $\mathcal{I}$ wrt $\mathcal{C}$. □

Theorem 5 *Computational complexity of* $ReduceBy^{\geq}(\mathcal{B})$ *is polynomial in the size of the input instance* $\mathcal{B}$.

Proof: Let l be the number of relations in $\mathcal{B}$, m be the maximum number of tuples in a relation, and n be maximum number of columns in a relation. Then the complexity of $ReduceBy^{\geq}()$ is $O(lm^2n)$. □

3.3 Tableaux for TGD

Tuple generating dependency (TGD) is a global constraint of the form

$$\forall \vec{x}(\exists \vec{y}\varphi(\vec{x},\vec{y}) \rightarrow \exists \vec{z}\psi(\vec{x},\vec{z}))$$

where φ and ψ are conjunctions of atoms (in other words, join of relations) of the global schema and every variable in $\vec{x}$ occurs in both φ and ψ. TGDs of the global schema $\mathcal{G}$ for a data integration system $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$ can also be expressed by tableaux in the same way as the mapping assertions are expressed by the MATs. The main difference between mapping assertions and global TGDs is that, for a global TGD σ : $\forall \vec{x}(\exists \vec{y}\varphi_{\mathcal{G}}(\vec{x},\vec{y}) \rightarrow \exists \vec{z}\psi_{\mathcal{G}}(\vec{x},\vec{z}))$, all the relations in both $\varphi_{\mathcal{G}}$ and $\psi_{\mathcal{G}}$ are from the global schema. Since the mapping assertions and the TGDs have the same form, a similar procedure $TGD2TTGD()$ like $MA2MAT()$ can be applied to σ to get a tabular structure $\mathcal{T}^{\psi_{\mathcal{G}}}_{\varphi_{\mathcal{G}}} = \langle \mathcal{T}_{\varphi_{\mathcal{G}}}, \mathcal{T}_{\psi_{\mathcal{G}}} \rangle$ for σ. We call such a structure a *tabular TGD* (TTGD). Procedure $TGD2TTGD()$ is shown in Figure 3.7. A TTGD $\mathcal{T}^{\psi_{\mathcal{G}}}_{\varphi_{\mathcal{G}}}$ can be applied to an RGD $\mathcal{B}$ to get an instance $\mathcal{B}' \supseteq \mathcal{B}$ such that $\mathcal{B}' \models \sigma$.

The interpretation of application of TTGD $\mathcal{T}^{\psi_{\mathcal{G}}}_{\varphi_{\mathcal{G}}}$ as an operator to an RGD $\mathcal{B}$ (i.e. $\mathcal{T}^{\psi_{\mathcal{G}}}_{\varphi_{\mathcal{G}}}(\mathcal{B})$) is different than the application of a MAT $\mathcal{T}^{\phi_{\mathcal{G}}}_{\phi_{\mathcal{S}}}$ to a source instance $\mathcal{C}$ (i.e. $\mathcal{T}^{\phi_{\mathcal{G}}}_{\phi_{\mathcal{S}}}(\mathcal{C})$). The difference arises because of two reasons, (1) RGD may contain null variables, and (2) application of a TTGD $\mathcal{T}^{\psi_{\mathcal{G}}}_{\varphi_{\mathcal{G}}}$ to an RGD may generate some tuples which again may lead to the violation of σ (i.e. may need repeated application of $\mathcal{T}^{\psi_{\mathcal{G}}}_{\varphi_{\mathcal{G}}}$ until a stable solution is reached).

When a TTGD is applied on an RGD as an operator, it adds zero or more tuples to the RGD so that it now satisfies the corresponding constraints. The following definition gives the semantics of a TTGD as an operator on a global instance.

Definition 4 *Given an RGD $\mathcal{B}$ and a TTGD $\mathcal{T}^{\psi_{\mathcal{G}}}_{\varphi_{\mathcal{G}}} = \langle \mathcal{T}_{\varphi_{\mathcal{G}}}, \mathcal{T}_{\psi_{\mathcal{G}}} \rangle$, $\mathcal{T}^{\psi_{\mathcal{G}}}_{\varphi_{\mathcal{G}}}(\mathcal{B})$ is formally defined as follows*

$$\mathcal{T}^{\psi_{\mathcal{G}}}_{\varphi_{\mathcal{G}}}(\mathcal{B}) = \mathcal{B} \cup \{\rho(\mathcal{T}_{\psi_{\mathcal{G}}}) | \textit{ for some valuation } \rho \textit{ we have } \rho(\mathcal{T}_{\varphi_{\mathcal{G}}}) \textit{ in } \mathcal{B}\}$$

In Definition 5 we give the semantics of a set of TTGDs as an operator on a global instance.

Procedure TGD2TTGD(m)
Input : A TGD assertion $\sigma : \forall\vec{x}(\exists\vec{y}\varphi_{\mathcal{G}}(\vec{x},\vec{y}) \rightarrow \exists\vec{z}\psi_{\mathcal{G}}(\vec{x},\vec{z}))$
Output: A TTGD $\mathcal{T}^{\psi_{\mathcal{G}}}_{\varphi_{\mathcal{G}}}$ representing σ
begin
Create a tableau $\mathcal{T}_{\varphi_{\mathcal{G}}}$ for the expression $\forall\vec{x}\exists\vec{y}\varphi_{\mathcal{G}}(\vec{x},\vec{y})$
Delete the summary of $\mathcal{T}_{\varphi_{\mathcal{G}}}$
Create a tableau $\mathcal{T}_{\psi_{\mathcal{G}}}$ for the expression $\forall\vec{x}\exists\vec{z}\psi_{\mathcal{G}}(\vec{x},\vec{z})$
Delete the summary of $\mathcal{T}_{\psi_{\mathcal{G}}}$
Modify $\mathcal{T}_{\psi_{\mathcal{G}}}$ as follows:
for each $z_i \in \vec{z}$ **do**
if (z_i is represented by b_j in $\mathcal{T}_{\psi_{\mathcal{G}}}$) **then**
replace b_j by $\perp_j$
endif
endfor
for each $x_i \in \vec{x}$ **do**
if x_i is represented by a_j in $\mathcal{T}_{\varphi_{\mathcal{G}}}$ and by a_k in $\mathcal{T}_{\psi_{\mathcal{G}}}$
replace a_k by a_j in $\mathcal{T}_{\psi_{\mathcal{G}}}$
endfor
$\mathcal{T}^{\psi_{\mathcal{G}}}_{\varphi_{\mathcal{G}}} \leftarrow \langle \mathcal{T}_{\varphi_{\mathcal{G}}}, \mathcal{T}_{\psi_{\mathcal{G}}} \rangle$
return $\mathcal{T}^{\psi_{\mathcal{G}}}_{\varphi_{\mathcal{G}}}$
end

Figure 3.7: Procedure TGD2TTGD()

Procedure FixedPoint$(\mathcal{B}, \mathcal{T}_{\Sigma})$
Input : an RGD $\mathcal{B}$
A set of TTGD $\mathcal{T}_{\Sigma}$ for a set of global TGD Σ
Output: an RGD $\mathcal{B}'$ such that $\mathcal{B}' \models \Sigma$
begin
$\mathcal{B}' = ReduceBy^{\simeq}(\mathcal{T}_{\Sigma}(\mathcal{B}))$
if $\mathcal{B} \simeq \mathcal{B}'$ **then**
return $\mathcal{B}'$
else
return FixedPoint$(\mathcal{B}', \mathcal{T}_{\Sigma})$
endif
end

Figure 3.8: Procedure FixedPoint()

Definition 5 *Let $\Sigma = \{\sigma_1 \dots \sigma_r\}$ be a set of global TGD constraint and $\mathcal{T}_\Sigma = \{\mathcal{T}_1 \dots \mathcal{T}_r\}$ be the set of TTGD for Σ, where $\mathcal{T}_i = TGD2TTGD(\sigma_i)$. Now, if $\mathcal{B}$ is an RGD, $\mathcal{T}_\Sigma(\mathcal{B})$ is defined as follows*

$$\mathcal{T}_\Sigma(\mathcal{B}) = \mathcal{T}_r(\mathcal{T}_{r-1}(\dots(\mathcal{T}_1(\mathcal{B}))\dots)$$

A TTGD $\mathcal{T}_\Sigma$ may have to be repeatedly applied on an RGD $\mathcal{B}$ until no more tuple is added to $\mathcal{B}$ on the application of $\mathcal{T}_\Sigma$. To determine up to what point $\mathcal{T}_\Sigma$ has to be applied on $\mathcal{B}$, we introduce the notion of $FixedPoint$ in the following definition.

Definition 6 *An RGD $\mathcal{B}$ is said to be a fixed point under the set of TTGD $\mathcal{T}_\Sigma$, if $\mathcal{B} \simeq FixedPoint(\mathcal{B}, \mathcal{T}_\Sigma)$ where $FixedPoint(\mathcal{B}, \mathcal{T}_\Sigma)$ is defined as follows:*

$$FixedPoint(\mathcal{B}, \mathcal{T}_\Sigma) = \begin{cases} \mathcal{B} & \text{If } \mathcal{B} \simeq ReduceBy^{\simeq}(\mathcal{T}_\Sigma(\mathcal{B})) \\ FixedPoint(ReduceBy^{\simeq}(\mathcal{T}_\Sigma(\mathcal{B})), \mathcal{T}_\Sigma) & \text{Otherwise} \end{cases}$$

A procedure for computing *Fixedpoint* is shown in Figure 3.8.

Theorem 6 *Computational complexity of $FixedPoint(\mathcal{B}, \mathcal{T}_\Sigma)$ is polynomial in the size of the input instance $\mathcal{B}$ and the size of $\mathcal{T}_\Sigma$.*

Proof: Complexity of $FixedPoint(\mathcal{B}, \mathcal{T}_\Sigma)$ is determined by the complexity of $ReduceBy^{\geq}()$ and $\mathcal{T}_\Sigma$ as an operator which are polynomial. So, $FixedPoint(\mathcal{B}, \mathcal{T}_\Sigma)$ is also polynomial.□

The following proposition states that when a set of TTGD is repeatedly applied on an RGD to reach a fixed point, then the final RGD satisfies the corresponding set of TGDs.

Proposition 3 *Let $\mathcal{B}' = FixedPoint(\mathcal{B}, \mathcal{T}_\Sigma)$ where $\mathcal{B}$ is an RGD and $\mathcal{T}_\Sigma$ is a set of TTGD for the set of target TGD Σ. We have that, $\mathcal{B}' \models \Sigma$.*

Theorem 7 states an important finding that a set of MATs and TTGDs can together be applied on a source instance of a data integration system to produce a minimal universal model for that system.

Dept

D	M	MN
CS	$\perp_{1_1}$	$Mary$

Emp

E	D
$E003$	CS

(a) RGD $\mathcal{B} = \mathcal{T}^{\phi_\mathcal{G}}_{\phi_\mathcal{S}}(\mathcal{C})$

	D	M	MN	E	D	
Dept	a_1	a_2	b_1	a_2	a_1	Emp

(b) TTGD $\mathcal{T}_{\sigma_1}$

	E	D	D	M	MN	
Emp	b_1	a_3	a_3	$\perp_2$	$\perp_3$	Dept

(c) TTGD $\mathcal{T}_{\sigma_2}$

Dept

D	M	MN
CS	$\perp_{1_1}$	$Mary$
CS	$\perp_{2_1}$	$\perp_{3_1}$

Emp

E	D
$E003$	CS
$\perp_{1_1}$	CS

(d) Model $\mathcal{A} = FixedPoint(\mathcal{B}, \mathcal{T}_\Sigma)$

Figure 3.9: Example of a set of TTGD as an operator on an RGD

Theorem 7 *Let $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$ be a data integration system where the global schema $\mathcal{G}$ has the set of TGD constraints Σ. For a source instance $\mathcal{C}$ of $\mathcal{S}$, if $\mathcal{A} = FixedPoint(\mathcal{T}_\mathcal{M}(\mathcal{C}), \mathcal{T}_\Sigma)$ then, $\mathcal{A}$ is a minimal universal model of $\mathcal{I}$ wrt $\mathcal{C}$.*

Proof: This proof can be divided into four subproofs as follows

1. $(\mathcal{C}, \mathcal{A}) \models \mathcal{M}$
2. $\mathcal{A} \models \Sigma$
3. For any other model $\mathcal{A}'$ of $\mathcal{I}$ wrt $\mathcal{C}$, there exist a homomorphism $\mathcal{A} \to \mathcal{A}'$
4. For $\mathcal{A}'' \subset \mathcal{A}$ there exist no homomorphism $\mathcal{A} \to \mathcal{A}''$

In the following we prove them one by one.

1. Let $\mathcal{B} = \mathcal{T}_\mathcal{M}(\mathcal{C})$. As $\mathcal{A} = FixedPoint(\mathcal{B}, \mathcal{T}_\Sigma)$, we have $\forall t(t \in \mathcal{B} \to \exists t'(t' \in \mathcal{A} \wedge t \simeq t'))$. So if, $(\mathcal{C}, \mathcal{B}) \models \mathcal{M}$ then we will have $(\mathcal{C}, \mathcal{A}) \models \mathcal{M}$. From theory 3 we know

that $(\mathcal{C}, \mathcal{B}) \models \mathcal{M}$. So, we have $(\mathcal{C}, \mathcal{A}) \models \mathcal{M}$

2. The algorithm $FixedPoint(\mathcal{B}, \mathcal{T}_\Sigma)$ halts when $\mathcal{A} \simeq ReduceBy^{\simeq}(\mathcal{T}_\Sigma(\mathcal{A}))$. i.e. when $\mathcal{T}_\Sigma(\mathcal{A})$ adds no tuples to $\mathcal{A}$ except copying some existing tuples with fresh nulls with new labels. So, if $\mathcal{A} = FixedPoint(\mathcal{B}, \mathcal{T}_\Sigma)$ we have $\mathcal{A} \models \Sigma$.

3. Since the structure of MAT for mapping assertions and TTGD for global TGDs are the same, we can use the same arguments as given for the proof of theorem 3 to deduce that for any other model $\mathcal{A}'$ of $\mathcal{I}$ wrt $\mathcal{C}$, there exist a homomorphism $\mathcal{A} \rightarrow \mathcal{A}'$

4. Minimization of $\mathcal{A}$ by using $ReduceBy^{\simeq}$ in the $FixedPoint()$ procedure gives the guarantee that $\mathcal{A}$ is the smallest model. So, $\mathcal{A}$ is a minimal universal model of $\mathcal{I}$ wrt $\mathcal{C}$. □

Example 5 *Consider the data integration system of Example 3. We add some global TGD constraints Σ to $\mathcal{G}$ where*

$\Sigma = \{\sigma_1, \sigma_2\}$

$\sigma_1 \equiv \forall d \forall m (\exists N.Dept(d, m, N) \rightarrow Emp(m, d))$

$\sigma_2 \equiv \forall d (\exists E.Emp(E, d) \rightarrow \exists M \exists N.Dept(d, M, N))$

We keep the same mapping assertion $\mathcal{M}$ as in example 3. So, MAT $\mathcal{T}_\mathcal{M}$ would be the same as in figure 3.3. Let the source instance $\mathcal{C}$ also be the same as in Example 3. So, we have $\mathcal{B} = \mathcal{T}_\mathcal{M}(\mathcal{C})$ as shown in Figure 3.9(a). TTGD for σ_1 and σ_2 are shown in figure 3.9(b) and figure 3.9(c), respectively. Let $\mathcal{T}_\Sigma = \{\mathcal{T}_{\sigma_1}, \mathcal{T}_{\sigma_2}\}$. $\mathcal{A} = FixedPoint(\mathcal{B}, \mathcal{T}_\Sigma)$, as shown in Figure 3.9(d), is a model of $\mathcal{I}$ wrt $\mathcal{C}$. Furthermore, $\mathcal{A}$ is a minimal universal model of $\mathcal{I}$ wrt $\mathcal{C}$. □

3.4 Tabuleaux for EGD

An *equality generating dependency* (EGD) is a global constraint having the form

$$\forall\vec{x}(\varphi(\vec{x}) \rightarrow (x_1 = x_2)),$$

where $\varphi(\vec{x})$ is a conjunction of atomic formulas over the global schema and $x_1 \in \vec{x},\ x_2 \in \vec{x}$. In the previous sections we showed how mapping assertions and global TGD constraints can be represented by MATs and TTGDs and how they can be used to produce a minimal universal model for a data integration system. In this section we show how to express an EGD constraint by a tabular form TEGD and how MATs, TTGDs and TEGDs all be used together to to produce a minimal universal model for a data integration system.

An EGD $\forall\vec{x}(\varphi(\vec{x}) \rightarrow (x_1 = x_2))$ can be expressed as a pair $\mathcal{E}_{\mathcal{T}_\varphi}^{b_1=b_2} = \langle\mathcal{T}_\varphi, (b_1, b_2)\rangle$, called *tabuleaux EGD* (TEGD), where $\mathcal{T}_\varphi$ is the summary-free tableau for $\varphi(\vec{x})$ and $b_1,\ b_2$ are non-distinguished variables that represent $x_1,\ x_2$, respectively, in $\mathcal{T}_\varphi$. Any $x_i \in \vec{x}$, shared by two or more atoms of φ, is represented by distinguished variables a_k in $\mathcal{T}_\varphi$ and all other $x_j \in \vec{x}$ are represented by non-distinguished variables. A TEGD $\mathcal{E}_{\mathcal{T}_\varphi}^{b_1=b_2} = \langle\mathcal{T}_\varphi, (b_1, b_2)\rangle$ can be depicted by a partitioned table whose left hand side contains $\mathcal{T}_\varphi$ and the right hand side contains the equality $b_1 = b_2$. An algorithm for converting an EGD to a TEGD is shown in Figure 3.10.

Example 6 *Consider a global schema $\mathcal{G}$ which has a single relation $R(A1, A2, A3)$ with attributes $A1$, $A2$ and $A3$. Now for an EGD $e : \forall x_1x_2x_3x_4x_5(R(x_1, x_2, x_3)\wedge R(x_1, x_4, x_5) \rightarrow (x_3 = x_5))$, we can have the TEGD $\mathcal{E}_{\mathcal{T}_\varphi}^{b_2=b_4}$, as shown in Figure 3.11(a), by applying the procedure $EGD2TEGD()$ on e. In other words, $\mathcal{E}_{\mathcal{T}_\varphi}^{b_2=b_4} = EGD2TEGD(e)$. Here φ stands for $\forall x_1x_2x_3x_4x_5(R(x_1, x_2, x_3) \wedge R(x_1, x_4, x_5))$.* □

A TEGD $\mathcal{E}_{\mathcal{T}_\varphi}^{b_i=b_j}$ can be used as an operator so that when applied to an instance $\mathcal{A}$, the resultant instance $\mathcal{A}' = \mathcal{E}_{\mathcal{T}_\varphi}^{b_i=b_j}(\mathcal{A})$ satisfies the EGD $e : \forall\vec{x}(\varphi(\vec{x}) \rightarrow (x_i = x_j))$. The following definitions give semantics to a TEGD as an operator.

Procedure EGD2TEGD(e)
Input : An EGD $e : \forall\vec{x}(\varphi(\vec{x}) \rightarrow (x_i = x_j))$
Output: A TEGD $\mathcal{E}_{\mathcal{T}_\varphi}^{b_k=b_l}$ representing e
begin
Create a tableau $\mathcal{T}_\varphi$ for the expression $\forall\vec{x}\varphi(\vec{x})$
Let x_i is represented by b_k and x_j is represented by b_l in $\mathcal{T}_\varphi$
Create an equality expression $(b_k = b_l)$
$\mathcal{E}_{\mathcal{T}_\varphi}^{b_k=b_l} \leftarrow \langle \mathcal{T}_\varphi, (b_k, b_l)\rangle$
return $\mathcal{E}_{\mathcal{T}_\varphi}^{b_k=b_l}$
end

Figure 3.10: Procedure EGD2TEGD()

The following definition provides the semantics of a TEGD as an operator.

Definition 7 *Let $\mathcal{E}_{\mathcal{T}_\varphi}^{b_1=b_2} = \langle \mathcal{T}_\varphi, (b_1, b_2)\rangle$ be a TEGD and $\mathcal{A}$ be a global instance. For all valuation ρ, such that $\rho(\mathcal{T}_\varphi)$ is in $\mathcal{A}$ and $\rho(b_1) \neq \rho(b_2)$, $\mathcal{E}_{\mathcal{T}_\varphi}^{b_1=b_2}$ modifies $\mathcal{A}$ by the following rules:*

$$\mathcal{E}_{\mathcal{T}_\varphi}^{b_1=b_2}(\mathcal{A}) = \begin{cases} Update(\mathcal{A}, \rho(b1), \rho(b2)) & \text{If both } \rho(b1) \text{ and } \rho(b2) \text{ are null} \\ & \text{variables or } \rho(b1) \text{ is a null} \\ & \text{variable and } \rho(b2) \text{ is a constant.} \\ Update(\mathcal{A}, \rho(b2), \rho(b1)) & \text{If } \rho(b2) \text{ is a null and } \rho(b1) \text{ is a constant.} \\ \emptyset & \text{If both } \rho(b1) \text{ and } \rho(b2) \text{ are constants.} \end{cases}$$

where the the procedure Update() is shown Figure 3.6.

Note that a TEGD does not generate any new tuple like TTGD or MAT. TEGD only modifies some null values of an instance with either another null value or a constant. If for some valuation ρ, $\rho(\mathcal{T}_\varphi)$ is in $\mathcal{A}$ and $\rho(b_1) \neq \rho(b_2)$ and both $\rho(b_1)$ and $\rho(b_2)$ are constants, $\mathcal{A}$ can not be modified to satisfy the EGD represented by the TEGD. In this case, an empty instance is returned.

Example 7 *Let us extend Example 6 by considering an RGD $\mathcal{B}$ as shown in Figure 3.11(b). Now, after applying TEGD $\mathcal{E}_{\varphi}^{b_2=b_4}$ of Figure 3.11 on $\mathcal{B}$ we get a new instance $\mathcal{A}$ =*

	$A1$	$A2$	$A3$	Equality
R	a_1	b_1	b_2	
R	a_1	b_3	b_4	$b_2 = b_4$

(a) TEGD $\mathcal{E}_{\mathcal{T}_\varphi}^{b_2=b_4} = \langle \mathcal{T}_\varphi, (b_2, b_4) \rangle$

R:

$A1$	$A2$	$A3$
p_1	q_1	$\perp_1$
p_1	q_2	r_2

(b) RGD $\mathcal{B}$

R:

$A1$	$A2$	$A3$
p_1	q_1	r_2
p_1	q_2	r_2

(c) $\mathcal{A} = \mathcal{E}_{\mathcal{T}_\varphi}^{b_2=b_4}(\mathcal{B})$

Figure 3.11: Example of a TEGD $\mathcal{E}_{\mathcal{T}_\varphi}^{b_2=b_4}$ and its application on an RGD

$\mathcal{E}_{\mathcal{T}_\varphi}^{b_2=b_4}(\mathcal{B})$ *as shown in Figure 3.11(c). Note that TEGD can help to make some incomplete information complete.*

In definition 8 we give the semantics of set of a TEGD when they are used as a single operator on a global instance.

Definition 8 *Consider a set of TEGDs* $\mathcal{E} = \{\mathcal{E}_1 \dots \mathcal{E}_r\}$. *For an RGD* $\mathcal{B}$, $\mathcal{E}(\mathcal{B})$ *is defined as follows*

$$\mathcal{E}(\mathcal{B}) = \mathcal{E}_r(\mathcal{E}_{r-1}(\dots(\mathcal{E}_1(\mathcal{B}))\dots))$$

Proposition 4 *Let* $\mathcal{A} = \mathcal{E}_\Upsilon(\mathcal{B})$ *where* $\mathcal{B}$ *is an RGD and* $\mathcal{E}_\Upsilon$ *is a set of TEGDs for the set of global EGD constraints* Υ. *We have that* $\mathcal{A} \models \Upsilon$.

Proof: Let $\Upsilon = \{e_1, e_2, \dots, e_r\}$ and $\mathcal{E}_\Upsilon = \{\mathcal{E}_1, \mathcal{E}_2 \dots, \mathcal{E}_r\}$. Now, let $\mathcal{B}_1 = \mathcal{E}_1(\mathcal{B})$. From definition 7 it is obvious that $\mathcal{B}_1 \models \{e_1\}$. Again, let $\mathcal{B}_2 = \mathcal{E}_2(\mathcal{B}_1)$. We have, $\mathcal{B}_2 \models \{e_2\}$ and in addition to that $\mathcal{E}_2$ make changes to $\mathcal{B}_1$ in such a way that it does not change it's property of satisfying e_1. So, $\mathcal{B}_2 \models \{e_1, e_2\}$. From Definition 8, we can deduce that $\mathcal{E}_\Upsilon(\mathcal{B}) \models \{e_1, e_2, \dots, e_r\}$. i.e. $\mathcal{A} \models \Upsilon$. □

In Theorem 8 we show how the tabular representation of mapping assertion, TGD and EGD constraints can be used on a source instance of a data integration system.

Theorem 8 *Let* $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$ *be a data integration system where* $\mathcal{G}$ *has TGD constraints* Σ *and EGD constraints* Υ. *Again, let* $\mathcal{T}_\mathcal{M}$ *be the set of MATs for* $\mathcal{M}$, $\mathcal{T}_\Sigma$ *be the set of TTGDs for* Σ *and* $\mathcal{E}_\Upsilon$ *be the set of TEGDs for* Υ. *For a source instance* $\mathcal{C}$ *of* $\mathcal{S}$, $\mathcal{A} = \mathcal{E}_\Upsilon(FixedPoint(\ \mathcal{T}_\Sigma(\mathcal{C}), \mathcal{T}_\Sigma))$ *is a minimal universal model of* $\mathcal{I}$ *wrt* $\mathcal{C}$ *if* $\mathcal{A}$ *is not empty.*

Proof: Let $\mathcal{A}' = FixedPoint(\mathcal{T}_{\mathcal{M}}(\mathcal{C}), \mathcal{T}_{\Sigma})$ and $\mathcal{A} = \mathcal{E}_{\Upsilon}(\mathcal{A}')$. Now, to prove theorem 8, we have to prove all of the following.

1. $(\mathcal{C}, \mathcal{A}) \models \mathcal{M}$
2. $\mathcal{A} \models \{\Sigma, \Upsilon\}$
3. $\mathcal{A}$ is a smallest of all URGD of $\mathcal{I}$ wrt $\mathcal{C}$.

Now we prove the conditions one by one.

1. From the definition of TEGD, we observe that when a TEGD is applied to an instance $\mathcal{A}'$, it changes only the null variables in $\mathcal{A}'$ and does not delete any tuple from $\mathcal{A}'$. Also, when a null variable $\perp_i$ is changed in $\mathcal{A}'$, then all copies of $\perp_i$ are changed with the same value. So, if $(\mathcal{C}, \mathcal{A}') \models \mathcal{M}$ holds then $(\mathcal{C}, \mathcal{A}) \models \mathcal{M}$ also holds. From Theorem 7, we have that $(\mathcal{C}, \mathcal{A}') \models \mathcal{M}$. Thus we get $(\mathcal{C}, \mathcal{A}) \models \mathcal{M}$.
2. Using the above arguments we can also prove that $\mathcal{A} \models \Sigma$ and from proposition 4 we get that $\mathcal{A} \models \Upsilon$. So, we get $\mathcal{A} \models \{\Sigma, \Upsilon\}$
3. In theorem 7 we showed that $\mathcal{A}' = FixedPoint(\mathcal{T}_{\mathcal{M}}(\mathcal{C}), \mathcal{T}_{\Sigma})$ is a minimal universal model of $\mathcal{I}$ wrt $\mathcal{C}$ if we consider Υ to be empty. When $\mathcal{E}_{\Upsilon}$ is applied to $\mathcal{A}'$ to get $\mathcal{A}$, $\mathcal{E}_{\Upsilon}$ makes minimum changes to the null variables of $\mathcal{A}'$ to have $\mathcal{A} \models \Upsilon$ (remember again, neither any addition of tuples nor any deletion of tuples is made to $\mathcal{A}'$). So, $\mathcal{A}$ must be the smallest universal RGD of $\mathcal{I}$ if we consider $\{\mathcal{M}, \Sigma, \Upsilon\}$ all together. That means $\mathcal{A}$ is a minimal universal model of $\mathcal{I}$ wrt $\mathcal{C}$. □

3.5 Summary

In this chapter, we first showed the mechanism for converting schema mappings of a data integration system into tabular forms, called MATs. Next, we presented algorithms for converting TGD and EGD constraints of the global schema of a data integration

system into tabular forms, called TTGD and TEGD, respectively. We gave the detailed semantics of MAT, TTGD and TEGD. Moreover, we showed how those tabular forms of mappings and constraints can be used as operators to produce minimal universal model for a given source instance in a data integration system. We put down some theories along with their proofs regarding the usefulness of MAT, TTGD and TEGD.

Chapter 4

Tableaux-Based Schema Mapping in Data Exchange

In chapter 3, we showed how mappings and constraints of a data integration system can be represented by tabular forms. We also showed how those tabular mappings and constraints can be used to produce minimal universal model for a data integration system. In this chapter, we show that mappings and constraints of a data exchange system can also be represented by tabular forms and these tabular forms can help to produce universal solutions and cores.

4.1 Data Exchange System (DES)

In this section we define some terms related to data exchange system.

Data exchange is the problem of taking data structured under a source schema and translating them into data structured under a target schema. In data exchange, a target instance is materialized that conforms to the source data as much as possible. Data exchange is used in many tasks that require data to be transferred between existing, independently created applications.

Formally, A *data exchange system* $\Omega = (S, \mathcal{T}, \Sigma_{st}, \Sigma_t)$ consists of a source schema S, a

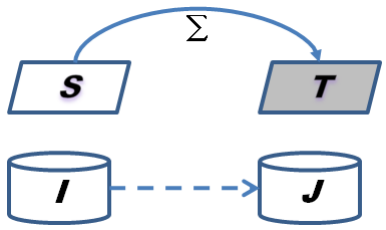


Figure 4.1: Data Exchange Problem

target schema $\mathcal{T}$, a set Σ_{st} of source-to-target mappings, and a set Σ_t of target constraints. In a data exchange system mappings and constraints are sometimes commonly termed as *dependencies*. The *data exchange problem* associated with this system is the following: given a finite source instance I, find a target instance J such that (I, J) satisfies Σ_{st} and J satisfies Σ_t. Such a J is called a solution of Ω wrt I. This problem is depicted in Figure 4.1.

There are obvious similarities, but also clear differences, between data integration and data exchange. In both frameworks, schema mappings are used to specify the relationships between the schemas involved. In data integration, the goal is to synthesize data from different sources into a unified view under a global schema; this view is virtual, in that the data remain in the sources and are accessed by users symbolically via the global schema. In data exchange, the goal is to take a given source instance and transform it to a target instance such that it satisfies the specifications of the schema mapping and also reflects the given source data as accurately as possible; unlike data integration, this target instance is a materialized instance, not a virtual view.

Universal solution. For a data exchange system $\Omega = (S, T, \Sigma_{st}, \Sigma_t)$ and a source instance I of S , a solution J is said to be a *universal solution* of Ω wrt I if for every solution I' of Ω wrt I, we have that $J \rightarrow J'$.

Core. For a data exchange system $\Omega = (S, T, \Sigma_{st}, \Sigma_t)$ and a source instance I of S , a universal solution J is said to be a *core* of Ω wrt I if $\nexists J''(J'' \subset J \wedge J \rightarrow J'')$ holds.

In Chapter 3 we show that when our proposed tabular representation of schema mapping can be used as an operator on a source instance of a data exchange system to generate core.

4.2 Tableaux for Data Exchange Mappings and Constraints

The mappings and constraints that are used to define a data exchange system are very similar to the mappings and constraints of a data integration system. Source-to-target tgds do the same thing in a data exchange system as mapping assertions do in a data integration system. Both mapping assertions and source-to-target tgds are expressed by GLAV mappings. Similarly, the purpose of using target constraints in a data exchange system is very similar to the purpose of using global constraints in a data integration system. Target tgds are expressed by the GLAV mappings. Target egds have the same form as the global egd constraints of a data integration system.

Since the structure of a mapping assertion is the same as the structure of a source-to-target tgd, they can be represented by similar tabular representation. Let us call the tabular representation of a source-to-target tgd an ST-TTGD. ST-TTGD has the same structure as the MAT. A very similar procedure like $MA2MAT()$ in Figure 3.2 can be used to convert a source-to-target tgd to an ST-TTGD. To avoid repetition and save space, we do not describe the structure of ST-TTGD and the conversion process. As a MAT can be used as an operator on a source instance of a data integration system to produce a RGD, an ST-TTGD can be used as an operator on a source instance of a data exchange system to produce a target instance. Following definition gives the semantics of ST-TTGD as an operator on a source instance of a data exchange system.

Definition 9 *Consider a data exchange system* $\Omega = (S, T, \Sigma_{st}, \Sigma_t)$. *For a source-to-*

target tgd $\sigma \in \Sigma_{st}$ *of the form* $\forall\vec{x}(\exists\vec{y}\varphi(\vec{x},\vec{y}) \rightarrow \exists\vec{z}\psi(\vec{x},\vec{z}))$ *and a source instance* I *of* S. *Let* $\mathcal{T}_{\varphi}^{\psi} = \langle\mathcal{T}_{\varphi}, \mathcal{T}_{\psi}\rangle$ *be the ST-TTGD for* σ.*The value of the* $\mathcal{T}_{\varphi}^{\psi}$ *wrt* I, *written as* $\mathcal{T}_{\varphi}^{\psi}(I)$, *is a set of tuples obtained by applying some valuation function* ρ *on the target tableau* $\mathcal{T}_{\psi}$ *such that* ρ *produces some tuples of the source instance* I *while applied on the source tableau* $\mathcal{T}_{\varphi}$. *Formally,*

$\mathcal{T}_{\varphi}^{\psi}(I)$= $\{\rho(\mathcal{T}_{\psi})|$ *for some valuation* ρ, $\rho(\mathcal{T}_{\varphi}) \in I\}$

The following definition gives the semantics of a set of ST-TTGD as an operator on a source instance.

Definition 10 *Consider a data exchange system* $\Omega = (S, T, \Sigma_{st}, \Sigma_t)$, *where* $\Sigma_{st} = \{\sigma_1 \ldots \sigma_r\}$. *Let* $\mathcal{T}_{\Sigma_{st}} = \{\mathcal{T}_1 \ldots \mathcal{T}_r\}$ *be a set of ST-TTGDs for* Σ_{st}, *where* $\mathcal{T}_i$ *is the ST-TTGD for* σ_i. *Now, if* I *is an instance of* S, $\mathcal{T}_{\Sigma_{st}}(I)$ *denotes the union of all the tuples obtained by applying each* $\mathcal{T}_i \in \mathcal{T}_{\Sigma_{st}}$ *on* I *for* $1 \leq i \leq r$. *Formally,*

$$\mathcal{T}_{\Sigma_{st}}(I) = \cup_{i=1}^{r}\mathcal{T}_i(I)$$

Again, the structure of a global tgd constraint of a data integration system is the same as target tgd of a data exchange system and so, they can be represented by similar tabular representation. Let us call the tabular representation of a target tgd a T-TTGD. T-TTGD has the same structure as the TTGD of chapter 3. A very similar procedure like $TGD2TTGD()$ in Figure 3.7 can be used to convert a target tgd to a T-TTGD. A T-TTGD can be used as an operator on a target instance of a data exchange system to produce a new target instance. When a T-TTGD is applied on a target instance as an operator, it adds zero or more tuples to the target instance, so that it now satisfies the corresponding constraints. The following definition gives the semantics of a TTGD as an operator on a target instance.

Definition 11 *Given a global instance* J *and a T-TTGD* $\mathcal{T}_{\varphi_t}^{\psi_t} = \langle\mathcal{T}_{\varphi_t}, \mathcal{T}_{\psi_t}\rangle$, $\mathcal{T}_{\varphi_t}^{\psi_t}(J)$ *is formally defined as follows*

$$\mathcal{T}_{\varphi_t}^{\psi_t}(J) = J \cup \{\rho(\mathcal{T}_{\psi_t}) | \textit{ for some valuation } \rho \textit{ we have } \rho(\mathcal{T}_{\varphi_t}) \textit{ in } J\}$$

In Definition 12 we give the semantics of a set of T-TTGDs as an operator on a target instance.

Definition 12 *Let* $\Sigma_t = \{\sigma_{t_1} \ldots \sigma_{t_r}\}$ *be a set of global tgd constraint and* $\mathcal{T}_{\Sigma_t} = \{\mathcal{T}_{t_1} \ldots \mathcal{T}_{t_r}\}$ *be the set of T-TTGD for* Σ_t*, where* $\mathcal{T}_{t_i}$ *is the T-TTGD for the target tgd* σ_{t_i}*. Now, if* J *is a target instance,* $\mathcal{T}_{\Sigma_t}(J)$ *is defined as follows*

$$\mathcal{T}_{\Sigma_t}(J) = \mathcal{T}_{t_r}(\mathcal{T}_{t_{r-1}}(\ldots(\mathcal{T}_{t_1}(J))\ldots)$$

A T-TTGD $\mathcal{T}_{\Sigma_t}$ may have to be repeatedly applied on a global instance J until no more tuple is added to J on the application of $\mathcal{T}_{\Sigma_t}$. To determine up to what point $\mathcal{T}_{\Sigma_t}$ has to be applied on J, we use the same $FixedPoint$ notion as was introduced in Chapter 3. $FixedPoint$ is redefined below in the context of data exchange.

Definition 13 *A global instance* J *is said to be a fixed point under the set of T-TTGDs* $\mathcal{T}_{\Sigma_t}$*, if* $J \simeq FixedPoint(J, \mathcal{T}_{\Sigma_t})$ *where* $FixedPoint(J, \mathcal{T}_{\Sigma_t})$ *is defined as follows:*

$$FixedPoint(J, \mathcal{T}_{\Sigma_t}) = \begin{cases} J & \text{If } J \simeq ReduceBy^{\simeq}(\mathcal{T}_{\Sigma_t}(J)) \\ FixedPoint(ReduceBy^{\simeq}(\mathcal{T}_{\Sigma_t}(J)), \mathcal{T}_{\Sigma_t}) & \text{Otherwise} \end{cases}$$

Procedure $ReduceBy^{\simeq}$ is shown in Figure 3.5.

Again, the structure of a global egd constraint of a data integration system is the same as target egd of a data exchange system and so, they can be represented by similar tabular representation. Let us call the tabular representation of a target egd a T-TEGD. A T-TEGD has the same structure as the TEGD of chapter 3. A very similar procedure like $EGD2TEGD()$ in Figure 3.10 can be used to convert a target egd to a T-TEGD. A T-TEGD can also be used as an operator on a target instance of a data exchange system to produce a new target instance. When a T-TEGD is applied on a target instance as an operator, it changes some of the null values in such a way that the new target instance satisfies the corresponding egd constraint. The following definition gives the semantics of a T-TEGD as an operator on a target instance.

Definition 14 *Let* $\mathcal{E}^{b_1=b_2}_{\mathcal{T}_\varphi} = \langle \mathcal{T}_\varphi, (b_1, b_2) \rangle$ *be a T-TEGD and J be a target instance. For all valuation* ρ*, such that* $\rho(\mathcal{T}_\varphi)$ *in J and* $\rho(b_1) \neq \rho(b_2)$*,* $\mathcal{E}^{b_1=b_2}_{\mathcal{T}_\varphi}$ *modifies J by the following rules:*

$$\mathcal{E}^{b_1=b_2}_{\mathcal{T}_\varphi}(J) = \begin{cases} Update(J, \rho(b1), \rho(b2)) & \text{If both } \rho(b1) \text{ and } \rho(b2) \text{ are null} \\ & \text{variables or } \rho(b1) \text{ is a null} \\ & \text{variable and } \rho(b2) \text{ is a constant.} \\ Update(J, \rho(b2), \rho(b1)) & \text{If } \rho(b2) \text{ is a null and } \rho(b1) \text{ is a constant.} \\ \emptyset & \text{If both } \rho(b1) \text{ and } \rho(b2) \text{ are constants.} \end{cases}$$

where the procedure Update() is shown Figure 3.6.

Example 8 *Consider a data exchange system* $\Omega = (S, T, \Sigma_{st}, \Sigma_t)$*. The source schema S has one relation DeptEmp(* D, M, E*) listing departments with their manager names and their employee ids. The target schema T has two relations Dept(* D, M, MN*) and* $Emp(E, D)$*. Dept*(D, M, MN) *lists departments, their manager ids and manager names, and* $Emp(E, D)$ *relates employee ids with their departments.* Σ_{st} *and* Σ_t *are defined as follows:*

$\Sigma_{st} = \{\sigma\}$

$\Sigma_t = \{\sigma_{t_1}, \sigma_{t_2}\}$

$\sigma \equiv \forall d \forall n \forall e(DeptEmp(d, n, e) \rightarrow \exists m(Dept(d, m, n) \wedge Emp(e, d)))$

$\sigma_{t_1} \equiv \forall d \forall m(\exists N.Dept(d, m, N) \rightarrow Emp(m, d))$

$\sigma_{t_2} \equiv \forall d(\exists E.Emp(E, d) \rightarrow \exists M \exists N.Dept(d, M, N))$

ST-TTGD $\mathcal{T}_\sigma$ *and T-TTGDs* $\mathcal{T}_{\sigma_{t_1}}$ *and* $\mathcal{T}_{\sigma_{t_2}}$ *are shown in Figure 4.2(a),4.2(b) and 4.2(c), respectively. For a source instance I of Figure 4.2(d), we have, target instance* $J = \mathcal{T}_{\Sigma_{st}}(I)$ *as shown in Figure 4.2(e) and target instance* $J' = FixedPoint(J, \mathcal{T}_{\Sigma_t})$ *as shown in Figure 4.2(f).* □

	D	M	E	D	M	MN	E	
DeptEmp	a_1	a_2	a_3	a_1	$\perp_1$	a_2		Dept
				a_1			a_3	Emp

(a) ST-TTGD $\mathcal{T}_\sigma$

	D	M	MN	E	D	
Dept	a_1	a_2	b_1	a_2	a_1	Emp

(b) T-TTGD $\mathcal{T}_{\sigma_{t_1}}$

	E	D	D	M	MN	
Emp	b_1	a_3	a_3	$\perp_2$	$\perp_3$	Dept

(c) T-TTGD $\mathcal{T}_{\sigma_{t_2}}$

DeptEmp

D	M	E
CS	$Mary$	$E003$

(d) Source instance I

Dept

D	M	MN
CS	$\perp_{1_1}$	$Mary$

Emp

E	D
$E003$	CS

(e) Target instance $J = \mathcal{T}_\sigma(I)$

Dept

D	M	MN
CS	$\perp_{1_1}$	$Mary$
CS	$\perp_{2_1}$	$\perp_{3_1}$

Emp

E	D
$E003$	CS
$\perp_{1_1}$	CS

(f) Target instance $J' = FixedPoint(J, \mathcal{T}_{\Sigma_t})$

Figure 4.2: Example of ST-TTGD and target T-TTGD

4.3 Properties of Tabular Mappings and Constraints

In this section we describe some properties of ST-TTGD, T-TTGD and T-TEGD in the form of propositions and theorems.

Proposition 5 *Consider a data exchange system $\Omega = (S, T, \Sigma_{st}, \Sigma_t)$. Let $\sigma \in \Sigma_{st}$ be a source-to-target tgd and $\mathcal{T}_{\varphi}^{\psi} = \langle \mathcal{T}_{\varphi}, \mathcal{T}_{\psi} \rangle$ be the ST-MAT for σ. For an instance I of S and a valuation ρ, whenever $\rho(\mathcal{T}_{\varphi}) \subseteq I$ holds, then we have $\rho(\mathcal{T}_{\psi}) \equiv \mathcal{T}_{\varphi}^{\psi}(I)$.*

Proof: This result can directly be derived from the definition of $\mathcal{T}_{\varphi}^{\psi}(I)$. □

In Theorem 9 we show that when a ST-TTGD $\mathcal{T}_{\phi_S}^{\phi_G}$ is applied on a source instance it produces a target instance that satisfies the corresponding st-tgds.

Theorem 9 *Consider a data exchange system $\Omega = (S, T, \Sigma_{st}, \Sigma_t)$. For a source-to-target tgd $\sigma \in \Sigma$ of the form $\forall \vec{x}(\exists \vec{y} \varphi(\vec{x}, \vec{y}) \rightarrow \exists \vec{z} \psi(\vec{x}, \vec{z}))$ and a source instance I of S, we have that $(I, \mathcal{T}_{\varphi}^{\psi}(I)) \models \sigma$, where $\mathcal{T}_{\varphi}^{\psi}$ is the ST-TTGD for σ.*

Proof: ST-TTGD $\mathcal{T}_{\varphi}^{\psi} = \langle \mathcal{T}_{\varphi}, \mathcal{T}_{\psi} \rangle$ is constructed in such a way that, both $\mathcal{T}_{\varphi}$ and $\mathcal{T}_{\psi}$ have the same distinguished variable for representing $x_i \in \vec{x}$. So, for any valuation ρ, $\rho(\mathcal{T}_{\varphi})$ and $\rho(\mathcal{T}_{\psi})$ will have the same value for the $\vec{x}$ components. The $\vec{z}$ components are represented by null variables and $\vec{y}$ components are represented by nondistinguished variables. So, they have no effect on each other. To prove Theorem 9, it is enough to prove that whenever φ is satisfied by some tuples in I, ψ is also satisfied by some tuples in $\mathcal{T}_{\varphi}^{\psi}(I)$. Now, let I satisfies φ. In this case for some valuation ρ, $\rho(\mathcal{T}_{\varphi}) \subseteq I$ must hold. From Proposition 5, we get $\rho(\mathcal{T}_{\psi}) \equiv \mathcal{T}_{\varphi}^{\psi}(I)$. Since, $\rho(\mathcal{T}_{\psi})$ satisfies ψ, $\mathcal{T}_{\varphi}^{\psi}(I)$ must satisfy ψ. Thus we prove the theorem. □

In Theorem 10 we claim and prove that the set of ST-TTGD $\mathcal{T}_{\Sigma_{st}}$ corresponding to a set of source-to-target tgds Σ_{st} can be used as an operator on a source instance I to generate a universal solution for I.

Theorem 10 *Consider a data exchange system $\Omega = (S, T, \Sigma_{st}, \Sigma_t)$, where Σ_t is empty and $\mathcal{T}_{\Sigma_{st}}$ is the set of ST-TTGDs for Σ_{st}. If I is an instance of S and $J = \mathcal{T}_{\Sigma_{st}}(I)$, then J is a universal solution for Ω wrt I.*

Proof. For J to be a universal solution of Ω wrt I, the following conditions should hold.

1. $(I, J) \models \Sigma_{st}$ (In other words, J is a solution of Ω wrt I).
2. For any other solution of J' of Ω wrt I (i.e. $(I, J') \models \Sigma_{st}$), there is a homomorphism $h : J \to J'$.

Condition 1: Let $\Sigma_{st} = \{\sigma_1, \ldots, \sigma_r\}$ and $\mathcal{T}_\Sigma = \{\mathcal{T}_1, \ldots, \mathcal{T}_r\}$, where $\mathcal{T}_i$ is the ST-TTGD for σ_i. According to Theorem 9 we have that $(I, \mathcal{T}_i(I)) \models \sigma_i$ for $1 \leq i \leq r$. Combining the statement for all σ_i together, we have

$$\begin{aligned} & (I, \cup_{i=1}^{r} \mathcal{T}_i(I)) \models \{\sigma_1, \ldots, \sigma_r\} \\ \Rightarrow\ & (I, \mathcal{T}_{\Sigma_{st}}(I)) \models \Sigma_{st} \\ \Rightarrow\ & (I, J) \models \Sigma_{st} \end{aligned}$$

Condition 2: Consider an arbitrary $\sigma_i \in \Sigma_{st}$ where $\sigma_i \equiv \forall \vec{x}(\exists \vec{y} \varphi_i(\vec{x}, \vec{y}) \to \exists \vec{z} \psi_i(\vec{x}, \vec{z}))$. If J' is a solution of I, we have that $(I, J') \models \sigma$. Since the source instance I consists of constants only, $\vec{x}$ components must be constants in J'. This is also the case for J according to the construction rules of $TGD2TTGD$. So, we have a homomorphism $h_i : J \to J'$, where $h_i(c) = c$ for all $\vec{x}$ components. All $\vec{z}$ components are labeled nulls in J. On the other hand, the $\vec{z}$ components of J' would either be a constant or a labeled null. Let $z_l \in \vec{z}$ component is represented by $\perp_m$ in J. For J' we have the following two cases.

Case 1: z_l is represented by a constant c_n. In this case, extend h_i by adding $h_i(\perp_m) = c_n$.

Case 2: z_l is represented by a labeled null $\perp_o$. In this case, extend h_i by adding $h_i(\perp_m) = \perp_o$. This completes the homomorphism $h_i : J \to J'$ for the tuples of J that come from the valuation of $\mathcal{T}_{\psi_i}$. In this way, we can get homomorphisms $\{h_j : 1 \leq j \leq r\}$ for the tgds $\{\sigma_j : 1 \leq j \leq r\}$. Since, fresh null variables are used whenever a valuation is

applied to a target tableau, there should not be any conflict among the homomorphisms $\{h_j : 1 \leq j \leq r\}$ and we can take union of them to build a new homomorphism. Let $h = \cup_{i=1}^{r} h_i$. Now, the homomorphism $h : J \rightarrow J'$ maps each element of J to some elements of J'. □

In Theorem 10 we have shown that a set of T-TGDs generated by the procedure *TGD2TTGD* produces universal solution for a given source instance. But there is no guarantee that the universal solution produced by the set of ST-TTGDs is a core (i.e. the minimal universal solution). Algorithm *ReduceBy*$^{\simeq}$ of Figure 3.5 in Chapter 3 can be adapted to take a target instance (which is a universal solution) as input and to produce a core. This adapted algorithm can be applied to a universal solution to convert it into a core.

Now, in theorem 11 we claim and prove that the algorithm *ReduceBy*$^{\simeq}$ can be used along with the set of ST-TTGDs to produce a core solution for a given source instance.

Theorem 11 *Consider a data exchange system* $\Omega = (S, T, \Sigma_{st}, \Sigma_t)$*, where* Σ_t *is empty and* $\mathcal{T}_{\Sigma_{st}}$ *is the set of TTGDs for* Σ_{st}*. If* I *is an instance of* S *and* $J = ReduceBy^{\simeq}(\mathcal{T}_{\Sigma}(I))$ *then* J *is a core for* Ω *wrt* I*, where the procedure ReduceBy*$^{\simeq}$*() is shown in Figure 3.5.*

Proof: In theorem 10 we proved that $\mathcal{T}_{\Sigma_{st}}(I)$ returns a universal solution for Ω wrt I. But this solution may contain a group of tuples which are equivalent up to the renaming of the nulls. *ReduceBy*$^{\simeq}$ eliminates such redundancy by keeping a single tuple representing the group and deleting the rest. Let t and t' be two tuples which are equivalent up to the renaming of the nulls. Before deleting t' from $\mathcal{T}_{\Sigma_{st}}(I))$, if any other tuple t'' of $\mathcal{T}_{\Sigma_{st}}(I)$ has any common null variable as t' in the attribute A then $t''[A]$ is replaced by the value $t[A]$. This ensures that the universality of the solution is not lost while reducing it. So, $J = ReduceBy^{\simeq}(\mathcal{T}_{\Sigma_{st}}(I))$ is the smallest universal solution of Ω wrt I i.e. J is a core solution of Ω wrt I. □

So far we have considered data exchange systems with empty target constraints. In the following theorem we investigate how ST-TTGD and T-TTGD can combindly be

used to produce a core of a data exchange system for a given source instance. We use a procedure $FixedPoint()$ in that theory which is the same as the procedure $FixedPoint()$ of chapter 3 except it takes a target instance and T-TTGD as input and outputs a target instance that satisfies the corresponding target tgd.

Theorem 12 *Consider a data exchange system* $\Omega = (S, T, \Sigma_{st}, \Sigma_t)$*, where* Σ_t *consists of only target tgds. For a source instance* I *of* S*, if* $J = FixedPoint(\mathcal{T}_{\Sigma_{st}}(I), \mathcal{T}_{\Sigma_t})$ *then,* J *is a core for* Ω *wrt* I*.*

Proof: This proof can be divided into three subproofs as follows

1. J is a solution of Ω wrt I
2. J is a universal solution
3. J is a core

In the following we prove them one by one.

1. J is a solution of Ω wrt I: Let $J_{st} = \mathcal{T}_{\Sigma_{st}}(I)$. From theory 10 we know that $(I, J_{st}) \models \Sigma_{st}$. The algorithm $FixedPoint(J, \mathcal{T}_{\Sigma_t})$ halts when $J \simeq ReduceBy^{\simeq}(\mathcal{T}_{\Sigma_t}(J))$. i.e. when $\mathcal{T}_{\Sigma_t}(J)$ adds no tuple to J except copying some existing tuples with fresh nulls with new labels. So, if $J = FixedPoint(J_{st}, \mathcal{T}_{\Sigma_t})$ we have $J \models \Sigma_t$. Again, as $J = FixedPoint(J_{st}, \mathcal{T}_{\Sigma_t})$, we have $\forall t(t \in J_{st} \rightarrow \exists t'(t' \in J \wedge t \simeq t'))$. This implies that $(I, J) \models \Sigma_{st}$ is also true. Combining the above two results we have $(I, J) \models \{\Sigma_{st}, \Sigma_t\}$. So, J is a solution of Ω wrt I.

2. J is a universal solution: Since the structure of tabular TGD for both s-t tgd and target tgd are the same, we can use the same arguments as given for the proof of theorem 10 to deduce that J is a universal solution.

3. J is a core: Minimization of J by using $ReduceBy^{\simeq}$ in $FixedPoint$ gives the guarantee that J is the smallest universal solution. So, J is a core. $\square$

The conversion process of target egd of a data exchange system into TEGD is similar to the conversion process of global egd constraints of a data integration system into TEGD as shown in section 3.4. In the following theorem we consider all of s-t tgs, target tgd and target egd of a data exchange system together.

Proposition 6 *Let $J' = \mathcal{E}_{\Sigma_e}(J)$ where J is a target instance and $\mathcal{E}_{\Sigma_e}$ is a set of T-TEGD for the set of target egds Σ_e. We have that $J' \models \Sigma_e$.*

Theorem 13 *Consider a data exchange system $\Omega = (S, T, \Sigma_{st}, \Sigma_t)$, where $\Sigma_t = \{\Sigma_{tt}, \Sigma_{te}\}$ consists of a set of target tgds Σ_{tt} and a set of target egds Σ_{te} . Again, let $\mathcal{T}_{\Sigma_{st}}$ be the set of ST-TTGDs for Σ_{st}, $\mathcal{T}_{\Sigma_{tt}}$ be the set of T-TTGDs for Σ_{tt} and $\mathcal{E}_{\Sigma_{te}}$ be the set of T-TEGDs for Σ_{te}. For a source instance I of S, $J = \mathcal{E}_{\Sigma_{te}}(FixedPoint(\ \mathcal{T}_{\Sigma_{st}}(I), \mathcal{T}_{\Sigma_{tt}}))$ is a core for Ω wrt I.*

Proof: Let $J' = FixedPoint(\mathcal{T}_{\Sigma_{st}}(I), \mathcal{T}_{\Sigma_{tt}})$ and $J = \mathcal{E}_{\Sigma_{te}}(J')$. Now, to prove theorem 13, we have to prove all of the followings.

1. $J \models \Sigma_{te}$
2. $(I, J) \models \{\Sigma_{st}, \Sigma_{tt}\}$
3. J is a universal solution and also a core wrt $\{\Sigma_{st}, \Sigma_{tt}\}$

1. We get that $J \models \Sigma_{te}$ directly from proposition 6.
2. From Theorem 12, we have that $(I, J') \models \{\Sigma_{st}, \Sigma_{tt}\}$. Now, from the definition of T-TEGD, we observe that when a T-TEGD is applied to an instance J, it changes only the null variables in J and does not delete any tuple from J. Also, when a null variable $\perp_i$ is changed in J, then all copies of $\perp_i$ are changed with the same value. So, $J = \mathcal{E}_{\Sigma_e}(J')$ may not be a universal solution wrt $\{\Sigma_{st}, \Sigma_{tt}\}$ but still we have that $J \models \{\Sigma_{st}, \Sigma_{tt}\}$ (i.e. J is a solution wrt $\{\Sigma_{st}, \Sigma_{tt}\}$).

3. In theorem 12 we showed that $J' = FixedPoint(\mathcal{T}_{\Sigma_{st}}(I), \mathcal{T}_{\Sigma_{tt}})$ is a core solution wrt $\{\Sigma_{st}, \Sigma_{tt}\}$. When $\mathcal{E}_{\Sigma_e}$ is applied to J' to get J, $\mathcal{E}_{\Sigma_{te}}$ make the minimum changes to the null variables of J' to have $J \models \Sigma_e$ (remember that neither any addition of tuples nor any deletion of tuples is made to J). So, J must be a core solution (which is universal too) of I if we consider $\{\Sigma_{st}, \Sigma_{tt}, \Sigma_{te}\}$ all together. That is J is a core solution of Ω wrt I. □

4.4 Summary

In this chapter, we first presented the semantics of the tabular representation of the constraints of a data exchange system. We then claimed and proved that, when used as an operator on a source instance, these tabular constraints produce universal solution and core for a data exchange system.

Chapter 5

Optimization of Schema Mappings

In chapters 3 and 4 we showed that the constraints of a schema mapping of a data integration and a data exchange system can be converted into tabular structures which can be used as operators on a source instance to generate a most general and minimal target instance. In this chapter, we define equivalence of schema mappings of two data integration systems and characterize that equivalence in terms of the tabular representation of the mappings and constraints. In this regard, we first review the equivalence of classical tableaux. Based on that, we define the equivalences of MATs, TTGDs, and TEGDs. Second, we use those equivalences to decide whether two schema mappings of a data integration system are equivalent. Finally, we present an optimizing algorithm that converts a schema mapping to an equivalent optimized one.

5.1 Equivalence of Tableaux

The following definitions are related to the equivalence of classical tableaux.

Let w_1 and w_2 be two rows in a tableau $\mathcal{T}$ with scheme R. If for every attribute A in R, $w_2(A)$ is a distinguished variable implies $w_1(A)$ is a distinguished variable and the tag of both w_1 and w_2 are the same, then w_1 is said to subsume w_2.

	X	Y	Z
	a_1	a_2	a_3
P	a_1	a_2	b_1
P	b_2	a_2	b_3
P	b_4	a_2	a_3

(a) $\mathcal{T}$

	X	Y	Z
	a_4	a_5	a_6
P	a_4	a_5	b_5
P	b_6	a_5	a_6

(b) $\mathcal{T}'$

	X	Y	Z
	a_1	a_2	a_3
P	a_1	a_2	b_1
P	b_4	a_2	a_3

(c) $\mathcal{T}_s = SUB(\mathcal{T})$

	X	Y	Z
	a_4	a_5	a_6
P	a_4	a_5	b_5
P	b_6	a_5	a_6

(d) $\mathcal{T}'_s = SUB(\mathcal{T}')$

Figure 5.1: Example of two equivalent tableaux $\mathcal{T}$ and $\mathcal{T}'$

Example 9 *Row 2 of the tableau $\mathcal{T}$ in Figure 5.1(a) (the shadowed row) is subsumed by row 1 of $\mathcal{T}$.* □

For a tableau $\mathcal{T}$, $SUB(\mathcal{T})$ denotes a tableau consisting of the set of rows in $\mathcal{T}$ that are not subsumed by any other row of $\mathcal{T}$.

Example 10 *By deleting row 2 of $\mathcal{T}$ we get the tableau $\mathcal{T}_s$ of Figure 5.1(c) which has no rows subsumed by other rows. So, we can say $\mathcal{T}_s = SUB(\mathcal{T})$.* □

Definition 15 *Two tableaux $\mathcal{T}_1$ and $\mathcal{T}_2$ are said to be identical except for possibly a one-to-one renaming of the distinguished and nondistinguished (or null) symbols, denoted as $\mathcal{T}_1 \cong \mathcal{T}_2$ if the following conditions hold.*

1. $\mathcal{T}_1.Columns = \mathcal{T}_2.Columns$
2. $\mathcal{T}_1.Tags = \mathcal{T}_2.Tags$
3. *There exists a one-to-one correspondence among the distinguished and nondistinguished(or null) symbols preserving their respective columns and rows.* □

Example 11 *Consider the two tableaux $\mathcal{T}_s$ and $\mathcal{T}'_s$ of Figure 5.1(c) and Figure 5.1(d), respectively. Tags and Columns of the two tableaux are the same. And also, there exists*

a one-to-one correspondence μ between the variables of the two tableaux as follows: $\mu(a_1) = a_4$, $\mu(a_2) = a_5$, $\mu(a_3) = a_6$, $\mu(b_1) = b_5$, *and* $\mu(b_4) = b_6$. *So, we can say that,* $\mathcal{T}_s \cong \mathcal{T}'_s$. □

Two tableaux on the same schema are said to be equivalent if they produce same result for every instance of that schema. Formal definition of equivalence of two tableaux is given below.

Definition 16 *Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two tableaux over schema R. $\mathcal{T}_1 \sqsupseteq \mathcal{T}_2$ expresses the fact that, $\mathcal{T}_1(r) \supseteq \mathcal{T}_2(r)$ for all instance r of R. Tableaux $\mathcal{T}_1$ and $\mathcal{T}_2$ are equivalent, written $\mathcal{T}_1 \equiv \mathcal{T}_2$, if $\mathcal{T}_1 \sqsupseteq \mathcal{T}_2$ and $\mathcal{T}_2 \sqsupseteq \mathcal{T}_1$. That is, $\mathcal{T}_1 \equiv \mathcal{T}_2$ if $\mathcal{T}_1(r) = \mathcal{T}_2(r)$ for every instance r of R.* □

Theorem 14 *[44] $\mathcal{T}_1 \equiv \mathcal{T}_2$ if and only if $SUB(\mathcal{T}_1) \cong SUB(\mathcal{T}_2)$.*

Example 12 *In Figure 5.1 we have $\mathcal{T}_s = SUB(\mathcal{T})$ and $\mathcal{T}'_s = SUB(\mathcal{T}')$. We also have $\mathcal{T}_s \cong \mathcal{T}'_s$. i.e. $SUB(\mathcal{T}) \cong SUB(\mathcal{T}')$. So, by theory 14 we can conclude that $\mathcal{T} \equiv \mathcal{T}'$.* □

5.2 Equivalence of Tabular Mappings and Constraints

In the previous section we reviewed the definitions and theories regarding tableaux query. In this section we define equivalence of MATs, TTGDs and TEGDs which we use in the following section to decide the equivalence of schema mappings of data integration systems.

5.2.1 Equivalence of MAT

We adapt the conditions of tableaux equivalence of section 5.1 to define the equivalence of two MATs. ϱ denotes a function that takes a RGD $\mathcal{B}$ as input and outputs another RGD $\mathcal{B}'$ such that $\mathcal{B}'$ is exactly like $\mathcal{B}$ except some of the nulls are renamed. The purpose of the function ϱ is to rename the nulls of the target instances so that they can be compared for subsumption, equality or isomorphism.

Definition 17 *Let* $\mathcal{T}_{\phi_S}^{\phi_G} = \langle \mathcal{T}_{\phi_S}, \mathcal{T}_{\phi_G} \rangle$ *and* $\mathcal{T}_{\phi_{S'}}^{\phi_{G'}} = \langle \mathcal{T}_{\phi_{S'}}, \mathcal{T}_{\phi_{G'}} \rangle$ *be two MATs for a data integration system* $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$. $\mathcal{T}_{\phi_S}^{\phi_G} \sqsupseteq \mathcal{T}_{\phi_{S'}}^{\phi_{G'}}$ *expresses the fact that,* $\mathcal{T}_{\phi_S}^{\phi_G}(\mathcal{C}) \geq \mathcal{T}_{\phi_{S'}}^{\phi_{G'}}(\mathcal{C})$ *for any source instances* $\mathcal{C}$ *of* $\mathcal{S}$. *Two MATs* $\mathcal{T}_{\phi_S}^{\phi_G}$ *and* $\mathcal{T}_{\phi_{S'}}^{\phi_{G'}}$ *are said to be equivalent, written* $\mathcal{T}_{\phi_S}^{\phi_G} \equiv \mathcal{T}_{\phi_{S'}}^{\phi_{G'}}$, *if* $\mathcal{T}_{\phi_S}^{\phi_G} \sqsupseteq \mathcal{T}_{\phi_{S'}}^{\phi_{G'}}$ *and* $\mathcal{T}_{\phi_{S'}}^{\phi_{G'}} \sqsupseteq \mathcal{T}_{\phi_S}^{\phi_G}$.

Note that the symbol ' $\geq'$ refers to 'subsumption' defined in section 2.1.

Proposition 7 *Let* $\mathcal{T}_{\phi_S}^{\phi_G}$ *and* $\mathcal{T}_{\phi_{S'}}^{\phi_{G'}}$ *be two MATs for a data integration system* $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$. $\mathcal{T}_{\phi_S}^{\phi_G} \equiv \mathcal{T}_{\phi_{S'}}^{\phi_{G'}}$ *holds iff for every instance* $\mathcal{C}$ *of* $\mathcal{S}$, $ReduceBy^{\simeq}(\mathcal{T}_{\phi_S}^{\phi_G}(\mathcal{C})) \simeq ReduceBy^{\simeq}(\mathcal{T}_{\phi_{S'}}^{\phi_{G'}}(\mathcal{C}))$.

Proof: $\Rightarrow$ Consider a source instance $\mathcal{C}$ of $\mathcal{S}$. When $\mathcal{T}_{\phi_S}^{\phi_G} \equiv \mathcal{T}_{\phi_{S'}}^{\phi_{G'}}$, we have $\mathcal{T}_{\phi_S}^{\phi_G} \sqsupseteq \mathcal{T}_{\phi_{S'}}^{\phi_{G'}}$ and $\mathcal{T}_{\phi_{S'}}^{\phi_{G'}} \sqsupseteq \mathcal{T}_{\phi_S}^{\phi_G}$. So, we have $\mathcal{T}_{\phi_S}^{\phi_G}(\mathcal{C}) \geq \mathcal{T}_{\phi_{S'}}^{\phi_{G'}}(\mathcal{C})$ and $\mathcal{T}_{\phi_{S'}}^{\phi_{G'}}(\mathcal{C}) \geq \mathcal{T}_{\phi_S}^{\phi_G}(\mathcal{C})$. This is possible when for some renaming function ϱ, we have $\varrho(ReduceBy^{\simeq}(\mathcal{T}_{\phi_S}^{\phi_G}(I))) = ReduceBy^{\simeq}(\mathcal{T}_{\phi_{S'}}^{\phi_{G'}}(I))$. In other words, $ReduceBy^{\simeq}(\mathcal{T}_{\phi_S}^{\phi_G}(I)) \simeq ReduceBy^{\simeq}(\mathcal{T}_{\phi_{S'}}^{\phi_{G'}}(I))$.
$\Leftarrow$ The proof for the opposite direction is obvious. □

Definition 18 *Let* $\mathcal{T}_{\phi_S}^{\phi_G} = \langle \mathcal{T}_{\phi_S}, \mathcal{T}_{\phi_G} \rangle$ *and* $\mathcal{T}_{\phi_{S'}}^{\phi_{G'}} = \langle \mathcal{T}_{\phi_{S'}}, \mathcal{T}_{\phi_{G'}} \rangle$ *be two MATs. We say,* $\mathcal{T}_{\phi_S}^{\phi_G} \cong \mathcal{T}_{\phi_{S'}}^{\phi_{G'}}$ *when the following conditions hold*

1. $\mathcal{T}_{\phi_S} \cong \mathcal{T}_{\phi_{S'}}$,
2. $\mathcal{T}_{\phi_G} \cong \mathcal{T}_{\phi_{G'}}$, *and*
3. *The correspondence between the distinguished symbols of* $\mathcal{T}_{\phi_S}$ *and* $\mathcal{T}_{\phi_{S'}}$ *(say* μ^{ϕ_S}*), and the correspondence between the distinguished symbols of* $\mathcal{T}_{\phi_G}$ *and* $\mathcal{T}_{\phi_{G'}}$ *(say* μ^{ϕ_G}*) are the same (i.e.* $\mu^{\phi_S}(a_i) = a_j$ *iff* $\mu^{\phi_G}(a_i) = a_j$*).*

Proposition 8 *Let* $\mathcal{T}_{\phi_S}^{\phi_G}$ *and* $\mathcal{T}_{\phi_{S'}}^{\phi_{G'}}$ *be two MATs. If* $\mathcal{T}_{\phi_S}^{\phi_G} \cong \mathcal{T}_{\phi_{S'}}^{\phi_{G'}}$ *then* $\mathcal{T}_{\phi_S}^{\phi_G} \equiv \mathcal{T}_{\phi_{S'}}^{\phi_{G'}}$.

Proof: From definition 18 we observe that when $\mathcal{T}_{\phi_S}^{\phi_G} \cong \mathcal{T}_{\phi_{S'}}^{\phi_{G'}}$, $\mathcal{T}_{\phi_S}^{\phi_G}$ is same as $\mathcal{T}_{\phi_{S'}}^{\phi_{G'}}$ up to the renaming of the symbols preserving the same correspondence of symbols on both sides of the partitions of the two MATs (i.e. condition 3 of the definition 18). This

guarantees that when $\mathcal{T}_{\phi_{\mathcal{S}}}^{\phi_{\mathcal{G}}}$ and $\mathcal{T}_{\phi_{\mathcal{S}}'}^{\phi_{\mathcal{G}}'}$ are applied on the same instance $\mathcal{C}$ of $\mathcal{S}$, they will produce the same RGD up to the renaming of the nulls. So, according to the definition of equivalence of MATs, $\mathcal{T}_{\phi_{\mathcal{S}}}^{\phi_{\mathcal{G}}} \equiv \mathcal{T}_{\phi_{\mathcal{S}}'}^{\phi_{\mathcal{G}}'}$. □

Proposition 9 *Consider a MAT* $\mathcal{T}_{\phi_{\mathcal{S}}}^{\phi_{\mathcal{G}}} = \langle \mathcal{T}_{\phi_{\mathcal{S}}}, \mathcal{T}_{\phi_{\mathcal{G}}} \rangle$. *If* $\mathcal{T}_{sub} = \langle SUB(\mathcal{T}_{\phi_{\mathcal{S}}}), SUB(\mathcal{T}_{\phi_{\mathcal{G}}}) \rangle$, *then we have* $\mathcal{T}_{\phi_{\mathcal{S}}}^{\phi_{\mathcal{G}}} \equiv \mathcal{T}_{sub}$.

Proof: The claim of proposition 9 is a direct consequence of the fact that, the subsumption operation does not affect the distinguished variables of the source tableau $\mathcal{T}_{\phi_{\mathcal{S}}}$ and target tableau $\mathcal{T}_{\phi_{\mathcal{S}}}$. A MAT carries values from the source instance to the RGD only through distinguished variables. □

Theorem 15 *Consider two MATs* $\mathcal{T}_{\phi_{\mathcal{S}}}^{\phi_{\mathcal{G}}} = \langle \mathcal{T}_{\phi_{\mathcal{S}}}, \mathcal{T}_{\phi_{\mathcal{G}}} \rangle$ *and* $\mathcal{T}_{\phi_{\mathcal{S}}'}^{\phi_{\mathcal{G}}'} = \langle \mathcal{T}_{\phi_{\mathcal{S}}'}, \mathcal{T}_{\phi_{\mathcal{G}}'} \rangle$. *Let* $\mathcal{T}_{sub} = \langle SUB(\mathcal{T}_{\phi_{\mathcal{S}}}), SUB(\mathcal{T}_{\phi_{\mathcal{G}}}) \rangle$ *and* $\mathcal{T}'_{sub} = \langle SUB(\mathcal{T}_{\phi_{\mathcal{S}}'}), SUB(\mathcal{T}_{\phi_{\mathcal{G}}'}) \rangle$. *If* $\mathcal{T}_{sub} \cong \mathcal{T}'_{sub}$ *then* $\mathcal{T}_{\phi_{\mathcal{S}}}^{\phi_{\mathcal{G}}} \equiv \mathcal{T}_{\phi_{\mathcal{S}}'}^{\phi_{\mathcal{G}}'}$.

Proof: From proposition 9 we have,
(1)$\mathcal{T}_{\phi_{\mathcal{S}}}^{\phi_{\mathcal{G}}} \equiv \mathcal{T}_{sub}$, and
(2)$\mathcal{T}_{\phi_{\mathcal{S}}'}^{\phi_{\mathcal{G}}'} \equiv \mathcal{T}'_{sub}$
Now, if $\mathcal{T}_{sub} \cong \mathcal{T}'_{sub}$ then from proposition 8 we get
(3)$\mathcal{T}_{sub} \equiv \mathcal{T}'_{sub}$
Combining (1), (2) and (3), we get $\mathcal{T}_{\phi_{\mathcal{S}}}^{\phi_{\mathcal{G}}} \equiv \mathcal{T}_{\phi_{\mathcal{S}}'}^{\phi_{\mathcal{G}}'}$. □

Example 13 *Consider two data integration systems* $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$ *and* $\mathcal{I}' = (\mathcal{S}, \mathcal{G}, \mathcal{M}')$ *as follows. In both of the cases, source schema* $\mathcal{S}$ *has one relation* $P(X, Y, Z)$ *and the global schema* $\mathcal{G}$ *has a relation* $Q(X, Y, Z)$. *The Mapping assertions* $\mathcal{M} = \{m\}$ *and* $\mathcal{M}' = \{m'\}$ *are singletons.* m *and* m' *are given below:*

$m : \exists y_1 y_2 y_3 y_4 (P(x_1, x_2, y_1) \wedge P(y_2, x_2, y_3) \wedge P(y_4, x_2, x_3)) \rightsquigarrow Q(x_1, x_2, x_3)$

$m' : \exists y_1 y_2 (P(x_4, x_5, y_1) \wedge P(y_2, x_5, x_6)) \rightsquigarrow Q(x_4, x_5, x_6)$

For saving space we omitted the universal quantifiers. Both the mappings are equivalent because for every source instance $\mathcal{C}$, *if* $\mathcal{B}$ *and* $\mathcal{B}'$ *are MURGD of* $\mathcal{I}$ *and* $\mathcal{I}'$ *wrt* $\mathcal{C}$, *we have*

	X	Y	Z	X	Y	Z	
P	a_1	a_2	b_1				
P	b_2	a_2	b_3				
P	b_4	a_2	a_3	a_1	a_2	a_3	Q

(a) MAT $\mathcal{T}_{\phi_S}^{\phi_\mathcal{G}} = \langle \mathcal{T}_{\phi_S}, \mathcal{T}_{\phi_\mathcal{G}} \rangle$

	X	Y	Z	X	Y	Z	
P	a_4	a_5	b_5				
P	b_6	a_5	a_6	a_4	a_5	a_6	Q

(b) MAT $\mathcal{T}_{\phi_{S'}}^{\phi_\mathcal{G}'} = \langle \mathcal{T}_{\phi_{S'}}, \mathcal{T}_{\phi_{\mathcal{G}'}} \rangle$

P:

X	Y	Z
x_0	y_0	z_0
x_1	y_0	z_1

(c) Source instance $\mathcal{C}$

Q:

X	Y	Z
x_0	y_0	z_0
x_1	y_0	z_1
x_1	y_0	z_0
x_0	y_0	z_1

(d) MURGD $\mathcal{B}$

Figure 5.2: Two equivalent MATs and their solution $\mathcal{B}$ for the input instance $\mathcal{C}$

that $\mathcal{B} \simeq \mathcal{B}'$. For example, let $\mathcal{C}$ be a source instance as shown in Figure 5.2(c). The MURGD of both the systems $\mathcal{I}$ and $\mathcal{I}'$ wrt $\mathcal{C}$ under is $\mathcal{B}$ as shown in Figure 5.2(d).

This equivalence can be deduced from the equivalence of two MAT representing the above two mappings. m and m' can be expressed by the MATs $\mathcal{T}_{\phi_S}^{\phi_\mathcal{G}} = \langle \mathcal{T}_{\phi_S}, \mathcal{T}_{\phi_\mathcal{G}} \rangle$ and $\mathcal{T}_{\phi_{S'}}^{\phi_\mathcal{G}'} = \langle \mathcal{T}_{\phi_{S'}}, \mathcal{T}_{\phi_{\mathcal{G}'}} \rangle$, respectively, as shown in Figure 5.2(a) and 5.2(b).

We observe that in Figure 5.2 that $\langle SUB(\mathcal{T}_{\phi_S}), SUB(\mathcal{T}_{\phi_\mathcal{G}}) \rangle \cong \langle SUB(\mathcal{T}_{\phi_{S'}}), SUB(\mathcal{T}_{\phi_{\mathcal{G}'}}) \rangle$, and so can deduce that $\mathcal{T}_{\phi_S}^{\phi_\mathcal{G}} \equiv \mathcal{T}_{\phi_{S'}}^{\phi_\mathcal{G}'}$. □

Definition 19 *For two sets of MATs $\mathcal{T}_\mathcal{M} = \{\mathcal{T}_1 \dots \mathcal{T}_r\}$ and $\mathcal{T}_{\mathcal{M}'} = \{\mathcal{T}_1' \dots \mathcal{T}_{r'}'\}$, $\mathcal{T}_\mathcal{M} \sqsubseteq \mathcal{T}_{\mathcal{M}'}$ denotes the fact that $\mathcal{T}_{\mathcal{M}'}(\mathcal{C}) \geq \mathcal{T}_\mathcal{M}(\mathcal{C})$ for every instance $\mathcal{C}$. $\mathcal{T}_\mathcal{M} \equiv \mathcal{T}_{\mathcal{M}'}$ if $\mathcal{T}_\mathcal{M} \sqsubseteq \mathcal{T}_{\mathcal{M}'}$ and $\mathcal{T}_{\mathcal{M}'} \sqsubseteq \mathcal{T}_\mathcal{M}$.*

Theorem 16 *For two sets of MATs $\mathcal{T}_\mathcal{M} = \{\mathcal{T}_1 \dots \mathcal{T}_r\}$ and $\mathcal{T}_{\mathcal{M}'} = \{\mathcal{T}_1' \dots \mathcal{T}_{r'}'\}$, $\mathcal{T}_\mathcal{M} \equiv \mathcal{T}_{\mathcal{M}'}$ if for every $\mathcal{T}_i \in \mathcal{T}_\mathcal{M}$ there exists $\mathcal{T}_j' \in \mathcal{T}_{\mathcal{M}'}$ such that $\mathcal{T}_i \equiv \mathcal{T}_j'$ and vice versa.*

5.2.2 Equivalence of TTGD

The structure of MAT and TTGD are same. As operator, MAT is applied to a source instance to generate RGD and TTGD is applied to a RGD to generate a model for a data integration system. So, the theorems in section 5.2.1 for deducing the equivalence of two MATs, can also be applied to deduce the equivalence of two TTGD. The following two definitions are related to the equivalence of two sets of TTGDs.

Definition 20 *Consider two sets of global TTGD $\mathcal{T}_\Sigma$ and $\mathcal{T}_{\Sigma'}$. $\mathcal{T}_\Sigma \sqsubseteq \mathcal{T}_{\Sigma'}$ denotes the fact that $FixedPoint(\mathcal{B}, \mathcal{T}_{\Sigma'}) \simeq FixedPoint(\mathcal{B}, \mathcal{T}_\Sigma)$ for every RGD $\mathcal{B}$.*

Theorem 17 *For two sets of TTGD $\mathcal{T}_\Sigma = \{\mathcal{T}_1 \dots \mathcal{T}_r\}$ and $\mathcal{T}_{\Sigma'} = \{\mathcal{T}'_1 \dots \mathcal{T}'_{r'}\}$, $\mathcal{T}_\Sigma \equiv \mathcal{T}_{\Sigma'}$ if for every $\mathcal{T}_i \in \mathcal{T}_\Sigma$ there exists $\mathcal{T}'_j \in \mathcal{T}_{\Sigma'}$ such that $\mathcal{T}_i \equiv \mathcal{T}'_j$ and vice versa.*

5.2.3 Equivalence of TEGD

Tableaux equivalence can be used for TEGD equivalence also. Here, we investigate the conditions for defining equivalence of two TEGDs.

Definition 21 *Let $\mathcal{E}^{b_i=b_j}_{\mathcal{T}_{\phi_\mathcal{G}}} = \langle \mathcal{T}_{\phi_\mathcal{G}}, (b_i, b_j) \rangle$ and $\mathcal{E}^{b_k=b_l}_{\mathcal{T}_{\phi_{\mathcal{G}'}}} = \langle \mathcal{T}_{\phi_{\mathcal{G}'}}, (b_k, b_l) \rangle$ be two TEGDs. $\mathcal{E}^{b_i=b_j}_{\mathcal{T}_{\phi_\mathcal{G}}} \equiv \mathcal{E}^{b_k=b_l}_{\mathcal{T}_{\phi_{\mathcal{G}'}}}$ expresses the fact that, $\mathcal{E}^{b_i=b_j}_{\mathcal{T}_{\phi_\mathcal{G}}}(\mathcal{B}) \simeq \mathcal{E}^{b_k=b_l}_{\mathcal{T}_{\phi_{\mathcal{G}'}}}(\mathcal{B})$ for all instances $\mathcal{B}$.*

Theorem 18 *For two TEGDs $\mathcal{E}^{b_i=b_j}_{\mathcal{T}_{\phi_\mathcal{G}}} = \langle \mathcal{T}_{\phi_\mathcal{G}}, (b_i, b_j) \rangle$ and $\mathcal{E}^{b_k=b_l}_{\mathcal{T}_{\phi_{\mathcal{G}'}}} = \langle \mathcal{T}_{\phi_{\mathcal{S}'}}, (b_k, b_l) \rangle$, $\mathcal{E}^{b_i=b_j}_{\mathcal{T}_{\phi_\mathcal{G}}} \equiv \mathcal{E}^{b_k=b_l}_{\mathcal{T}_{\phi_{\mathcal{G}'}}}$ if the following conditions hold.*

1. $\mathcal{T}_{\phi_\mathcal{G}} \cong \mathcal{T}_{\phi_{\mathcal{G}'}}$
2. $\mu(b_i) = b_k$
3. $\mu(b_j) = b_l$

where μ is the one to one correspondence between the symbols of $\mathcal{T}_{\phi_\mathcal{G}}$ and $\mathcal{T}_{\phi_{\mathcal{G}'}}$

	A	B	C	Equality
P	a_1	b_1	b_2	
P	a_1	b_3	b_4	$b_2 = b_4$

(a) $\mathcal{E}^{b_2=b_4}_{\mathcal{T}_{\phi_\mathcal{G}}} = \langle \mathcal{T}_{\phi_\mathcal{G}}, (b_2, b_4) \rangle$

	A	B	C	Equality
P	a_2	b_5	b_6	
P	a_2	b_7	b_8	$b_6 = b_8$

(b) $\mathcal{E}^{b_6=b_8}_{\mathcal{T}_{\phi_{\mathcal{G}'}}} = \langle \mathcal{T}_{\phi_{\mathcal{G}'}}, (b_6, b_8) \rangle$

Figure 5.3: Example of two equivalent TEGD $\mathcal{E}^{b_2=b_4}_{\mathcal{T}_{\phi_\mathcal{G}}}$ and $\mathcal{E}^{b_6=b_8}_{\mathcal{T}_{\phi_{\mathcal{G}'}}}$

Example 14 *In Figure 5.3, we can obtain $\mathcal{T}_{\phi_\mathcal{G}}$ by a renaming μ of variables of $\mathcal{T}_{\phi_{\mathcal{G}'}}$, where $\mu(a_2) = a_1$, $\mu(b_5) = b_1$, $\mu(b_6) = b_2$, $\mu(b_7) = b_3$, and $\mu(b_8) = b_4$. So, we get $\mathcal{E}^{b_2=b_4}_{\mathcal{T}_{\phi_\mathcal{G}}} \equiv \mathcal{E}^{b_6=b_8}_{\mathcal{T}_{\phi_{\mathcal{G}'}}}$.*

Theorem 19 *Let $\mathcal{E}_\Upsilon = \{\mathcal{E}_1 \dots \mathcal{E}_r\}$ and $\mathcal{E}_{\Upsilon'} = \{\mathcal{E'}_1 \dots \mathcal{E'}_{r'}\}$ be two sets of TEGD. $\mathcal{E}_\Upsilon \equiv \mathcal{E}_{\Upsilon'}$ if for every $\mathcal{E}_i \in \mathcal{E}_\Upsilon$ there exists $\mathcal{E}'_j \in \mathcal{E}_{\Upsilon'}$ such that $\mathcal{E}_i \equiv \mathcal{E}'_j$ and vice versa.*

5.3 Equivalence of Schema Mapping

Schema mappings can be seen as a combination of mapping assertions, global tgds and global egds. Formally, a schema mapping Θ for a data integration system $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$ is a triple $(\mathcal{M}, \Sigma, \Upsilon)$ where $\mathcal{M}$ is the set of mapping assertions, Σ is the set of global tgds and Υ is the set of global egds of the global schema $\mathcal{G}$. For simplification, here we do not explicitly mention the source schema and the global schema in the specification of the schema mapping. Let $\Theta = (\mathcal{M}, \Sigma, \Upsilon)$ and $\Theta' = (\mathcal{M}', \Sigma', \Upsilon')$ are two schema mappings for the data integration systems $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$ and $\mathcal{I}' = (\mathcal{S}, \mathcal{G}', \mathcal{M}')$. We say that Θ and Θ' are equivalent if for every source instance they produce the same minimal universal models up to the renaming of the nulls. We denote this fact by $\Theta \equiv \Theta'$. In this section we use the equivalence of MAT, TTGD and TEGD for defining the equivalence of two schema mappings.

Definition 22 *Let $\Theta = (\mathcal{M}, \Sigma, \Upsilon)$ be a schema mapping of a data integration system. Again, let $\mathcal{T}_\mathcal{M}$ be the set of MATs for $\mathcal{M}$, $\mathcal{T}_\Sigma$ be the set of TTGD for Σ and $\mathcal{E}_\Upsilon$ be the*

set of TEGD for Υ. $\mathcal{T}_\Theta = \langle \mathcal{T}_\mathcal{M}, \mathcal{T}_\Sigma, \mathcal{E}_\Upsilon \rangle$ *is called the* schema mapping tableaux *(SMT) for* Θ.

Definition 23 *For two SMTs* $\mathcal{T}_\Theta = \langle \mathcal{T}_\mathcal{M}, \mathcal{T}_\Sigma, \mathcal{E}_\Upsilon \rangle$ *and* $\mathcal{T}_{\Theta'} = \langle \mathcal{T}_{\mathcal{M}'}, \mathcal{T}_{\Sigma'}, \mathcal{E}_{\Upsilon'} \rangle$, $\mathcal{T}_\Theta \equiv \mathcal{T}_{\Theta'}$ *if* $\mathcal{T}_\mathcal{M} \equiv \mathcal{T}_{\mathcal{M}'}$, $\mathcal{T}_\Sigma \equiv \mathcal{T}_{\Sigma'}$ *and* $\mathcal{E}_\Upsilon \equiv \mathcal{E}_{\Upsilon'}$.

Proposition 10 *Let* $\mathcal{T}_\Theta = \langle \mathcal{T}_\mathcal{M}, \mathcal{T}_\Sigma, \mathcal{E}_\Upsilon \rangle$ *and* $\mathcal{T}_{\Theta'} = \langle \mathcal{T}_{\mathcal{M}'}, \mathcal{T}_{\Sigma'}, \mathcal{E}_{\Upsilon'} \rangle$ *be two SMTs representing two schema mappings* $\Theta = (\mathcal{M}, \Sigma, \Upsilon)$ *and* $\Theta' = (\mathcal{M}', \Sigma', \Upsilon')$. *If* $\mathcal{T}_\Theta \equiv \mathcal{T}_{\Theta'}$ *then for every source instance* $\mathcal{C}$, $\mathcal{E}_\Upsilon(FixedPoint(\mathcal{T}_\mathcal{M}(\mathcal{C}), \mathcal{T}_\Sigma)) \simeq \mathcal{E}_{\Upsilon'}(FixedPoint(\mathcal{T}_{\mathcal{M}'}(\mathcal{C}), \mathcal{T}_{\Sigma'}))$

Definition 24 *Schema mappings of two systems are equivalent if the most general and minimal target instances for a given source instance satisfying those schema mappings are identical.*

The above definition is a general one. We can use that definition to define equivalence of schema mappings in data integration and data exchange systems. *Schema mappings of two data integration systems are equivalent if they produce identical minimal universal model for a given source instance.* Similarly, *Schema mappings of two data exchange systems are equivalent if they produce identical core for a given source instance.*

Theorem 20 *Let* $\Theta = (\mathcal{M}, \Sigma, \Upsilon)$ *and* $\Theta' = (\mathcal{M}', \Sigma', \Upsilon')$ *be two schema mappings of the integration systems* $\mathcal{I}$ *and* $\mathcal{I}'$, *respectively. Again, let* $\mathcal{T}_\Theta$ *and* $\mathcal{T}_{\Theta'}$ *be the SMTs representing* Θ *and* Θ' *respectively.* $\Theta \equiv \Theta'$ *if* $\mathcal{T}_\Theta \equiv \mathcal{T}_{\Theta'}$.

Proof: Let $\mathcal{T}_\Theta = \langle \mathcal{T}_\mathcal{M}, \mathcal{T}_\Sigma, \mathcal{E}_\Upsilon \rangle$ and $\mathcal{T}_{\Theta'} = \langle \mathcal{T}_{\mathcal{M}'}, \mathcal{T}_{\Sigma'}, \mathcal{E}_{\Upsilon'} \rangle$. If $\mathcal{T}_\mathcal{I} \equiv \mathcal{T}_{\mathcal{I}'}$, from proposition 10 we get, $\mathcal{E}_\Upsilon(FixedPoint(\mathcal{T}_\mathcal{M}(\mathcal{C}), \mathcal{T}_\Sigma)) \simeq \mathcal{E}_{\Upsilon'}(FixedPoint(\mathcal{T}_{\mathcal{M}'}(\mathcal{C}), \mathcal{T}_{\Sigma'}))$ for any source instance $\mathcal{C}$. From theorem 8, $\mathcal{E}_{\Sigma_e}(FixedPoint(\mathcal{T}_\mathcal{M}\ (\mathcal{C}), \mathcal{T}_\Sigma)$ is a minimal universal model of $\mathcal{I}$ wrt $\mathcal{C}$ and $\mathcal{E}_{\Upsilon'}(FixedPoint(\mathcal{T}_{\mathcal{M}'}(\mathcal{C}), \mathcal{T}_{\Sigma'}))$ is a minimal universal model of $\mathcal{I}'$ wrt $\mathcal{C}$. Since $\mathcal{C}$ is an arbitrary source instance, we can say that $\mathcal{I}$ and $\mathcal{I}'$ have the same minimal universal model (up to the renaming of the null variables). So, from the definition of equivalence of schema mappings of two data integration system we get that $\Theta \equiv \Theta'$. □

5.4 Optimizing Schema Mappings

In this section, we present an algorithm that optimizes schema mapping by reducing the number of joins in the mapping assertions and constraints. The mappings and the constraints are converted into tabular forms and those tabular forms are then reduced by deleting their redundant rows on the basis of subsumption. Finally, the tabular mappings are converted back to the GLAV mappings and constraints. We prove that the resultant optimized schema mapping is equivalent to the original one.

Our proposed algorithm $OptimizeSM()$ for optimizing schema mappings is shown in Figure 5.4. In this algorithm mapping assertions and global tgd constraints are separately converted into MATs and TTGDs. Then each row of the source tableau and the target tableau of each MAT and TTGD is checked whether it is subsumed by some of the rows of the tableau. If so, then it is deleted. To do this job, $OptimizeSM()$ calls $OptmzTab()$. Every distinguished variable of the deleted source row must be in some other rows of the tableau (by the definition of subsumption). Since only the distinguished variables of the source tableau are shared by the target tableau of the MAT, the MAT is not affected by such row deletion as an operator. On the other hand, the target tableaux has either distinguished variable or null variables. So, when a subsumed row is deleted from the target tableau, it only reduces some redundant null values. Since the egds usually have only two conjuncts and they are not subsumed by each other, no change is made to them. Then MATs and TTGDs are converted into mapping assertions and tgds. Finally, the data integration system is redefined using modified mapping assertions and tgd, and unchanged egds.

Example 15 *Consider the data integration system $\mathcal{I}$ of Example 13. After converting $\mathcal{M}$ into a set of MATs we get a singleton $\mathcal{T}_{\mathcal{M}}$ containing $\mathcal{T}_{\varphi}^{\psi}$ as shown in Figure 5.5(a). r_1, r_2 and r_3 are the labels of the rows of $\mathcal{T}_{\varphi}$. The algorithm of $OptimizeSM()$ of Figure 5.4 finds r_2 to be subsumed by r_1. So, it deletes r_2 from $\mathcal{T}_{\varphi}$ to produce $\mathcal{T}'_{\varphi}$ as shown on the left side of Figure 5.5(b). Now, let m'' denote the mapping assertion converted from*

Procedure $OptimizeSM(\Theta)$
Input : A schema mapping Θ.
Output: An optimized schema mapping Θ'
begin
Let $\Theta = \{\mathcal{M}, \Sigma, \Upsilon\}$
$\mathcal{T}_{\mathcal{M}} \leftarrow ConvertSetOfMA2SetOfMAT(\mathcal{M})$
$\mathcal{T}_{\Sigma} \leftarrow ConvertSetOfTgd2SetOfTTGD(\Sigma)$
for each $\mathcal{T}_i \in (\mathcal{T}_{\mathcal{M}} \cup \mathcal{T}_{\Sigma})$ **do**
Let $\mathcal{T}_i = \langle \mathcal{T}_{\varphi_i}, \mathcal{T}_{\psi_i} \rangle$
$\mathcal{T}_i = \langle OptmzTab(\mathcal{T}_{\varphi_i}), OptmzTab(\mathcal{T}_{\psi_i}) \rangle$
endfor
$\mathcal{M}' \leftarrow ConvertSetOfMAT2SetOfMA(\mathcal{T}_{\mathcal{M}})$
$\Sigma' \leftarrow ConvertSetOfTTGD2SetOfTgd(\mathcal{T}_{\Sigma})$
$\Theta' \leftarrow (\mathcal{M}', \Sigma', \Upsilon)$
return Θ'
end

Procedure $OptmzTab(\mathcal{T})$
Input : A Tableau $\mathcal{T}$.
Output: An optimized schema mapping $\mathcal{T}'$
for each $r_j \in \mathcal{T}.Rows$ **do**
for each $r_k \in \mathcal{T}.Rows$ **do**
// Check if r_j subsumes r_k
if $(r_j \neq r_k \wedge r_j \geq r_k)$ **then**
delete r_k from $\mathcal{T}$
endif
endfor
endfor
return $\mathcal{T}'$
end

Figure 5.4: Procedure $OptimizeSM()$ and procedure $OptimzTab()$

	X	Y	Z	X	Y	Z	
r_1	P	a_1	a_2	b_1			
r_2	P	b_2	a_2	b_3			
r_3	P	b_4	a_2	a_3	a_1	a_2	a_3	Q

(a) MAT $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}} = \langle \mathcal{T}_{\phi_\mathcal{S}}, \mathcal{T}_{\phi_\mathcal{G}} \rangle$

	X	Y	Z	X	Y	Z	
r_1	P	a_1	a_2	b_1			
r_3	P	b_4	a_2	a_3	a_1	a_2	a_3	Q

(b) MAT $\mathcal{T}_{\phi'_\mathcal{S}}^{\phi'_\mathcal{G}} = \langle \mathcal{T}_{\phi'_\mathcal{S}}, \mathcal{T}_{\phi'_\mathcal{G}} \rangle$

Figure 5.5: Example of MAT Optimization

the MAT $\mathcal{T}_{\varphi'}^{\psi'}$ of Figure 5.5(b). Note that m'' is the same as m' of Example 13. Set of constraints $\mathcal{M}''$ is formed by using σ''. This $\mathcal{M}''$ is the same as $\mathcal{M}'$ of Figure 13 which we already have shown to be equivalent to $\mathcal{M}$. □

In theorem 21 we show the correctness of the algorithm $OptimizeSM()$.

Theorem 21 *Let $\Theta = (\mathcal{M}, \Sigma, \Upsilon)$ be a schema mapping for a data integration system $\mathcal{I} = (\mathcal{S}, \mathcal{G}, \mathcal{M})$ and also let $\Theta' = OptimizeSM(\Theta)$. For a source instance $\mathcal{C}$, if $\mathcal{A}$ and $\mathcal{A}'$ are minimal universal models produced by Θ and Θ' respectively, then $\mathcal{A}$ and $\mathcal{A}'$ are equivalent up to the renaming of the labeled nulls.*

Proof: Let $m : \forall\vec{x}(\exists\vec{y}\phi_\mathcal{S}(\vec{x},\vec{y}) \rightarrow \exists\vec{z}\phi_\mathcal{G}(\vec{x},\vec{z}))$ be an arbitrary mapping assertion in $\mathcal{M}$ and $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}} = \langle \mathcal{T}_{\phi_\mathcal{S}}, \mathcal{T}_{\phi_\mathcal{G}} \rangle$ be the MAT for m. In $OptimizeSM()$, when we find a row r_1 of $\mathcal{T}_{\phi_\mathcal{S}}$ to be subsumed by another row r_2 of $\mathcal{T}_{\phi_\mathcal{S}}$, we delete r_1. Similarly, when we find a row r_3 of $\mathcal{T}_{\phi_\mathcal{G}}$ to be subsumed by another row r_4 of $\mathcal{T}_{\phi_\mathcal{G}}$, we delete r_3.When the modified MAT $\mathcal{T}_{\phi_\mathcal{S}}^{\phi_\mathcal{G}}$ is converted to a mapping assertion $m' : \forall\vec{x}(\exists\vec{y'}\phi'_\mathcal{S}(\vec{x},\vec{y'}) \rightarrow \exists\vec{z'}\phi'_\mathcal{G}(\vec{x},\vec{z'}))$, $\phi'_\mathcal{S}$ may have less conjunct than $\phi_\mathcal{S}$ but the universally quantified variables remain the same. Only universally quantified variables of $\phi_\mathcal{S}$ and $\phi'_\mathcal{S}$ are shared by $\phi'_\mathcal{G}$. On the other hand, $\phi'_\mathcal{G}$ may have less conjunct than $\phi_\mathcal{G}$ but it affects in reducing some redundant null values. So, any minimal model satisfying m also satisfies m'. This is also true for the TTGDs. When we count all the mapping assertions and global constraints together, we can say that any target instance that is model for Θ is also a model for Θ'. Thus we can conclude that, if $\mathcal{A}$ and $\mathcal{A}'$ are minimal universal models produced by Θ and Θ', respectively, then $\mathcal{A}$ and $\mathcal{A}'$ are equivalent up to the renaming of the labeled nulls. □

Theorem 22 *The complexity of the procedure* $OptimizeSM(\Theta)$ *is bounded by a polynomial in the size of the constraints.*

Proof: Let l be the number of constraints in Θ and m be the maximum number of conjuncts in a constraint and n be the maximum number of attributes in a constraint. Then the complexity of $OptimizeSM(\Theta)$ is $O(lm^2n)$. □

5.5 Summary

In this chapter, we first defined equivalence of schema mappings. Then, we characterized the equivalences of MATs, TTGDs, and TEGDs by finding out the conditions when two MATs, two TTGDs and two TEGDs are equivalent. After that we used those conditions to decide whether two schema mapping of a data integration system are equivalent. Finally, we presented our proposed algorithm for optimizing schema mappings. We showed that our optimization algorithm has polynomial time complexity. We also showed that this optimization process does not affect in producing the same minimal model up to the renaming of nulls. As a future work, we have plan to characterize the equivalence of schema mappings of two data exchange systems.

Chapter 6

Tableaux-Based Bi-Level Mappings in P2P Data Sharing Systems

P2P data sharing systems use either schema-level or data-level mappings to resolve schema as well as data heterogeneity among data sources (peers). Schema-level mappings create structural relationships among different schemas. On the other hand, data-level mappings associate data values in two different sources. These two kinds of mappings are complementary to each other. However, existing peer database systems have been based solely on either one of these mappings. We believe that if both mappings are addressed simultaneously in a single framework, the resulting approach will enhance data sharing in a way such that we can overcome the limitations of the non-combined approaches.

In this chapter, we introduce a model of a peer database management system (PDMS) which uses a mapping that combines schema-level and data-level mappings. We call this new kind of mapping *bi-level mapping*. We present the syntax and semantics of bi-level mappings. We also provide a query evaluation procedure for the PDMS that uses the bi-level mappings. Moreover, we introduce tabular representation of the bi-level mappings and show how this representation helps query translation process.

6.1 P2P Data Sharing System (P2P DSS)

In this section, we briefly discuss about the architecture of p2p data sharing system.

Recall from Chapter 2 that P2P data sharing system consists of an open-ended network of distributed computational peers, where each peer can exchange data with a set of other peers. A sample scenario of a P2P data sharing system is depicted in Figure 6.1. Each peer has its own local source. The local source in a peer is designed independently during the creation of the local database of that peer. In order to provide a unique access view of local as well as remote data to the users, each peer defines a schema called *peer schema*. The result of a local query posed in a peer is produced from the local source of the peer. A peer also defines *external sources* that are used to access data from its acquainted peers. An external source is a view of a peer schema of an acquainted peer and is defined through some GLAV mappings called *P2P mappings* or *peer mappings*. Mapping tables may be used in P2P mappings for resolving data-level heterogeneity. Presence of mapping tables on the dotted lines (i.e. P2P mappings) in Figure 6.1 expresses the fact that, when data crosses the border between the source and destination peer, it is changed in the data-level using the mapping tables associated to the P2P mappings. Peer schema is defined in terms of the local sources and external sources by some local GLAV mappings.

6.2 Motivating Example: Need For Bi-level Mappings

Consider two peers P_1 and P_2 in a P2P system as shown in Figure 6.2. Assume that peers store employee information to be shared with each other. P_1 has a local source *Empl_List(Id, Name, Position, salary)*. The attributes *Id*, *Name*, *Position*, and *Salary*

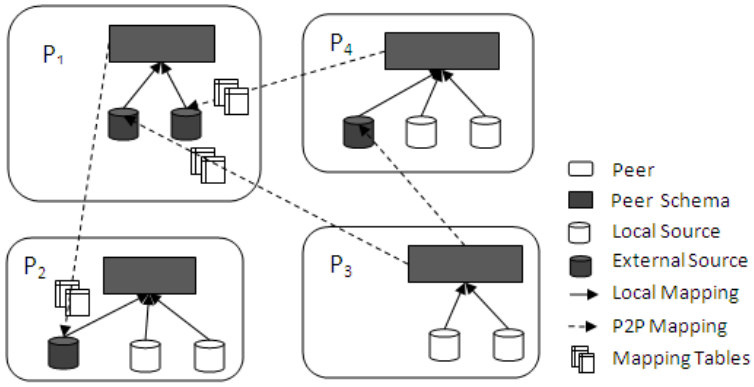


Figure 6.1: General scenario of P2P system

represent identification number, name, position, and the salary of an employee, respectively. Similarly, P_2 stores its employee information by the local sources *Employee(Id, Name, Jid)* and *Job_Desc(Jid, Job_Description)*. Attributes *Jid* and *Job_Description* represent the job identification number and the title of the job, respectively.

Now, assume that P_1 has an external source *E(Name, Position)* that illustrates that peer P_1 is interested in only the names and positions of employees stored in P_2. In order to give a unique access view to the data stored in P_1 and P_2, peer P_1 defines a peer schema $PS1$ which contains a single view *N_P(Name, Position)* for its users. Similarly, peer P_2 creates its peer schema $PS2$ which contains two views *P2E(Name, Jid)* and *P2J(Jid, Job_Description)*. This schema is designed considering its local source and other external sources from other peers (not mentioned in the figure). In Figure 6.2, L_{11}, L_{12}, L_{21}, L_{22} are local mappings those define the views of the peer schema and $P2P_{21}$ is a P2P mapping that defines the external source E of P_1 as follows:

$$L_{11} : \forall xy(\exists wz\ Empl\_List(w, x, y, z) \rightsquigarrow N\_P(x, y))$$
$$L_{21} : \forall yz(\exists x\ Employee(x, y, z) \rightsquigarrow P2E(y, z))$$
$$L_{22} : \forall xy(Job\_Desc(x, y) \rightsquigarrow P2J(x, y))$$
$$L_{12} : \forall xy(E(x, y) \rightsquigarrow N\_P(x, y))$$
$$P2P_{21} : \forall xz(\exists y(P2E(x, y) \wedge P2J(y, z)) \rightsquigarrow E(x, z))$$

From the Figure 6.2, we also notice that P_2 has a mapping table mt that maps the

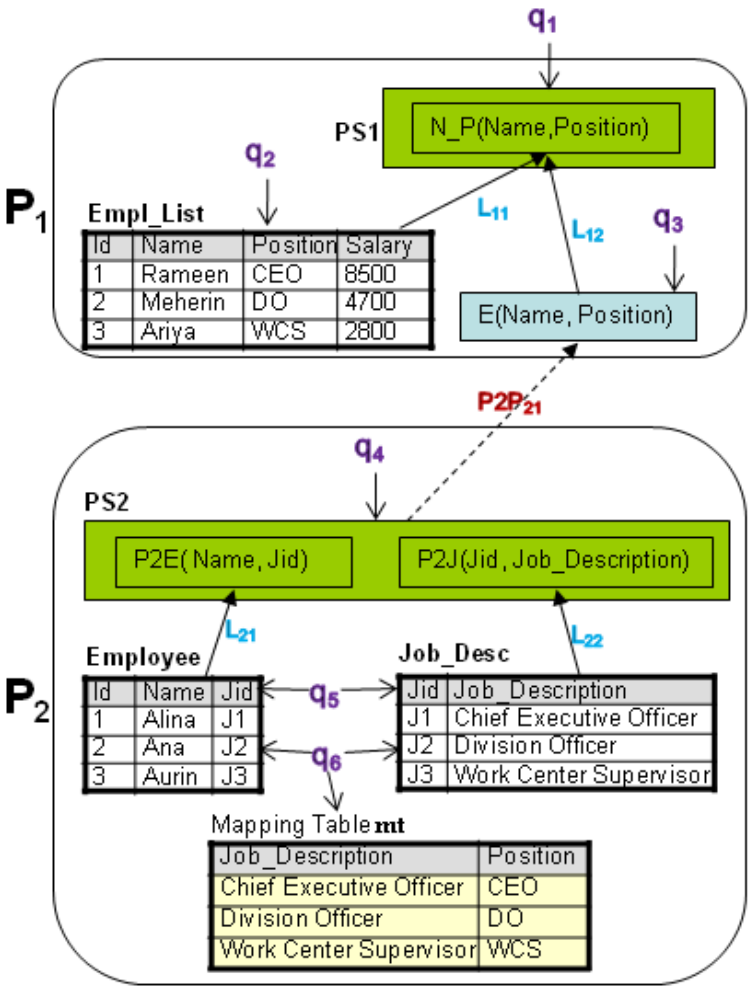


Figure 6.2: Motivating example

data vocabularies of the attribute *Job_Description* in source *Job_Desc* of P_2 with the attribute *Position* in source *Empl_List*. This mapping table is created since the two peers store job information using two different vocabularies. External source E in P_1 is defined as a view on the relations of the peer schema of P_2. In the following, we illustrate different situations that may occur when a query is posed to a peer. The examples show the need for the bi-level mappings that this chapter advocates for P2P systems.

Example 16 (Considering only schema mappings) *Assume that the mapping table mt is absent in peer P_2. In this case, peer P_1 has only the schema-level mappings with P_2. Suppose the query*

$$q_1 : \{x | N\_P(x,' CEO')\}$$

is posed at P_1 through its peer schema. Considering the mappings between N_P and Empl_List, q_1 is translated for the local source at P_1 as

$$q_2 : \{x | Empl\_List(x,' CEO')\}$$

Moreover, using the mappings between N_P and E, q_1 is translated for the external source at P_1 as

$$q_3 : \{x | E(x,' CEO')\}$$

Based on the mappings between E and the peer schema at P_2, query q_3 is translated as

$$q_4 : \{x | \exists y(P2E(x, y) \wedge P2J(y,' CEO'))\}$$

which is finally translated according to the local vocabulary of P_2 as

$$q_5 : \{x | \exists y(Employee(x, y) \wedge Job\_Desc(y,' CEO'))\}$$

Notice that the final result of the query q_1 is **{ *Rameen*}** *which is returned only from the local source at P_1. If we consider 'CEO' and 'Chief Executive Officer' to be semantically equivalent then q_1 should extract 'Alina' from P_2. Due to absence of data-level mappings, the query can not produce this result. Now assume that the mapping table mt exists. Hence, q_3 is translated for P_2 as*

$$q_6 : \{x|\exists yz(Employee(x, y) \wedge Job\_Desc(y, z) \wedge mt(z,'CEO'))\}$$

In this case, we get more results for the query q_1 and the complete answer to this query becomes **{ *Rameen, Alina*}**

Example 17 (Considering only data mappings) *In Figure 6.2 the external source E of P_1 is defined in terms of P2E and P2J of peer P_2. Projection and Join operators are used in that definition. Mapping tables are not expressive enough to express Projection or Join. Schema mappings are needed for such association between two sources.*

So, a mapping is necessary that is capable of dealing with both the syntactical (schema-level) and the semantic (data-level) heterogeneities at the same time.

6.3 Model of a PDMS

A P2P system Π is defined by a pair $\langle \mathcal{P}, \mathcal{M} \rangle$, where $\mathcal{P} = \{P_1, P_2, \cdots, P_n\}$ is a set of peers and $\mathcal{M} = \{\mathcal{M}_1, \cdots, \mathcal{M}_n\}$ is a set of peer mappings. A set of mappings $M_i^j \subseteq \mathcal{M}_i$ in P_i defines the mappings between peer P_i and P_j. The construction of mappings M_i^j forms an acquaintance (i, j) between P_i and P_j.

Suppose a P2P system $\Pi =< \mathcal{P}, \mathcal{M} >$. Formally, a peer $P_i \in \mathcal{P}$ is a tuple, where $P_i = (PS_i, R_i, L_i)$. Where,

- PS_i is the peer schema through which data in a peer is exposed to the external world.

- R_i is the set of sources comprised of local and external sources. We call it peer source or simply source.

- L_i is the set of GLAV local mappings which define the mappings between R_i and PS_i. Each local mapping, called *mapping assertion* (aka *tuple generating dependency*), in L_i has the form

$$\forall\vec{x}(\exists\vec{y}\varphi(\vec{x},\vec{y}) \rightsquigarrow \exists\vec{z}\psi(\vec{x},\vec{z}))$$

where $\varphi(\vec{x},\vec{y})$ and $\psi(\vec{x},\vec{z})$ are conjunctive queries over the relations in R_i and PS_i respectively.

$\mathcal{M}_i \in \mathcal{M}$ is a set of mappings, called bi-level mapping or *peer mappings*, that define the schema and data-level mappings between peers. Each mapping $m \in \mathcal{M}_i$ is a pair $< m^S_{j,k}, m^D_{j,k} >$, where:

- $m^S_{j,k}$ is a GLAV mapping (practically GAV, since $s(\vec{x})$ is always a single relation) of the form

$$\forall\vec{x}(\exists\vec{y}\varphi(\vec{x},\vec{y}) \rightsquigarrow s(\vec{x}))$$

where $\varphi(\vec{x},\vec{y})$ is a conjunctive query over the peer schema of a peer P_j and $s(\vec{x})$ is the k^{th} external source of P_i.

- $m^D_{j,k}$=MT=$\{mt_1, mt_2, \ldots, mt_q\} \subseteq MT^i_j$ is a set of mapping tables. MT^i_j denotes the set of mapping tables used to map data of P_j to data of P_i.

m can alternatively be represented with the mapping assertion as follows:

$$\forall\vec{x}(\exists\vec{y}\varphi(\vec{x},\vec{y}) \overset{MT}{\rightsquigarrow} s(\vec{x}))$$

We assumed that a curator with expertise in different domains is responsible for generating the mapping tables and the peer administrator maintains them in a peer. Schema mappings between two peers are initially created by the corresponding peer administrators when they agree to share data. Once created, the mappings are activated or deactivated depending on the presence of the corresponding peers in the network. Generating the mappings automatically is another research area and we did not address about this issue in this paper.

The semantics of $\overset{MT}{\rightsquigarrow}$ is described in the following section.

6.3.1 Semantics of Local Mappings

For each peer P_i, we introduce a first order logic (FOL) theory F_i, called peer theory, where alphabet contains all the relation symbols in a peer schema PS_i and the relations in R_i. The axioms of F_i include all the constraints of PS_i and one logical formula representing each local mapping in L_i. For a local mapping of the form $\forall\vec{x}(\exists\vec{y}\varphi(\vec{x},\vec{y}) \rightsquigarrow \exists\vec{z}\psi(\vec{x},\vec{z}))$ in L_i, a formula of the form $\forall\vec{x}(\exists\vec{y}\varphi(\vec{x},\vec{y}) \subseteq \exists\vec{z}\psi(\vec{x},\vec{z}))$ is added to F_i. F_i does not consider peer mappings. Thus, modeling a peerP_i as a GLAV integration system becomes equivalent to modeling the FOL theory F_i (ignoring the peer mappings in $\mathcal{M}_i$).

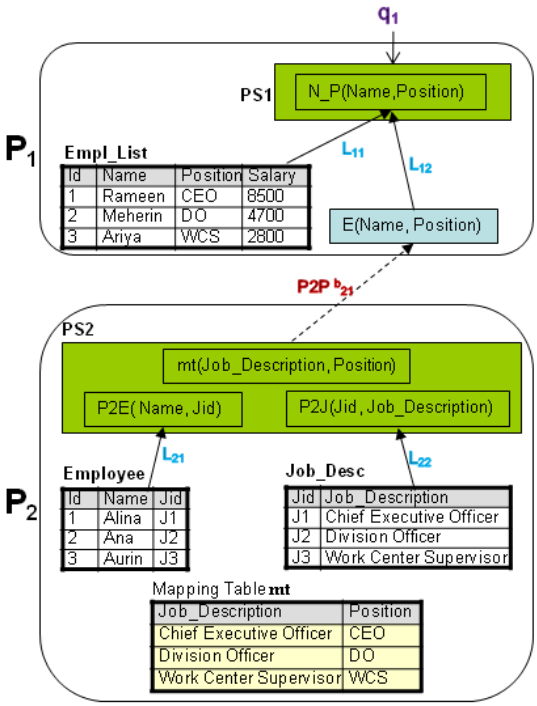
6.3.2 Semantics of Peer Mappings

Similar to the local mappings, the semantics of peer mappings can also be given in terms of FOL. However, to incorporate the mapping tables, we will use notations and definitions provided below.

Definition 25 Given a tuple t and a set of attributes U, $t[U]$ denotes the values of tuple t corresponding to the attributes in U.

Definition 26 (Mapping Table) Assume that U_i and U_j are non-empty set of attributes in two peers P_i and P_j respectively. A mapping table $mt[\vec{P},\vec{Q}]$ is a finite relation over the attributes $\vec{P} \subseteq U_i$ and $\vec{Q} \subseteq U_j$. A tuple $t = (\vec{a},\ \vec{b})$ in the mapping table indicates that the value $\vec{a} \in dom(\vec{P})$ is associated with the value $\vec{b} \in dom(\vec{Q})$. Variables can be used to simplify the expression of value associations. Consider a mapping table $m(L,R)$ where both the domain of L and R are same, say D. A tuple $(v, D-v)$ in m, where v is a variable, can be used to denote that any value of L can be mapped to any value of R except to itself.

Definition 27 (Valuation) A valuation ρ over a mapping table mt is a function that maps each constant value in mt to itself and each variable v of mt to the value in the intersection of the domains of the attributes where v appear.



(a) Bi-Level mappings for P2P mappings

E(Name, Position)

Name	Position
Alina	CEO
Ana	DO
Aurin	WCS

N_P(Name,Position)

Name	Position
Rameen	CEO
Meherin	DO
Ariya	WCS
Alina	CEO
Ana	DO
Aurin	WCS

(b) Extension of E and N_P

Figure 6.3: Bi-Level Mapping Example

Definition 28 ($Map(mt, \vec{p})$) Let $mt[\vec{P}, \vec{Q}]$ be a mapping table from $\vec{P}$ to $\vec{Q}$ and $\vec{p}$ is an element of the domain of $\vec{P}$. $Map(mt, \vec{p})$ returns a set of values $\vec{\Phi}$. $\vec{q} \in \vec{\Phi}$ if for some $t \in mt$ there exists a valuation ρ such that $\rho(t[\vec{P}]) = \vec{p}$ and $\rho(t[\vec{Q}]) = \vec{q}$. If the mapping for the value $\vec{p}$ is not defined in mt and type of $\vec{P}$ and $\vec{Q}$ matches then $\vec{\Phi} = \{\vec{p}\}$. If neither $\vec{p}$ is mapped to any value in mt, nor type of $\vec{P}$ and $\vec{Q}$ matches, then $\vec{\Phi} = Null$.

Definition 29 (*Augmentation function* τ) *Let $\vec{x}$ be a tuple whose schema contains the attributes $\vec{P}$. An augmentation function $\tau(\vec{x}, \vec{P}, \vec{Q}, q)$ returns a tuple $\vec{x'}$, where $\vec{x'}$ is exactly like $\vec{x}$ except the schema of $\vec{x'}$ has the attributes $\vec{Q}$ in place of $\vec{P}$ and $\vec{x'}[\vec{Q}] = q$.*

Definition 30 *Let $mt[\vec{P}, \vec{Q}]$ be a mapping table and $\vec{x}$ be a tuple, $mt[\vec{x}]$ denotes a set of tuples obtained by replacing the values of $\vec{P}$ attributes of $\vec{x}$ by the corresponding mapped values of $\vec{Q}$ in mt. Formally,*

$$mt[\vec{x}] \equiv \{\tau(\vec{x}, \vec{P}, \vec{Q}, \vec{q}) \mid \vec{q} \in Map(mt, \vec{x}[\vec{P}])\}$$

Let $MT = \{mt_1[\vec{P_1}, \vec{Q_1}], mt_2[\vec{P_2}, \vec{Q_2}], \ldots, mt_n[\vec{P_n}, \vec{Q_n}]\}$ *be a set of mapping tables, where for any pairs of mapping tables* $(mt_i[\vec{P_i}, \vec{Q_i}], mt_j[\vec{P_j}, \vec{Q_j}])$, $mt_i \neq mt_j \implies \vec{P_i} \neq \vec{P_j}$, *then* $MT[\vec{x}]$ *denotes a set of tuples resulted from transformation of* $\vec{x}$ *by all the member mapping tables of* MT. *Formally,*

$$MT[\vec{x}] \equiv \{\tau(\ldots\tau(\tau(\vec{x}, \vec{P_1}, \vec{Q_1}, \vec{q_1}), \vec{P_2}, \vec{Q_2}, \vec{q_2})\ldots, \vec{P_n}, \vec{Q_n}, \vec{q_n}) | \\ \vec{q_1} \in Map(mt_1, \vec{x}[\vec{P_1}]) \wedge \quad \ldots \wedge \vec{q_n} \in Map(mt_n, \vec{x}[\vec{P_n}])\}$$

We have overloaded [ ] in many of our definitions; its meaning, however, will be clearly understood from the context.

Definition 31 *Let* $\vec{x}$ *be a tuple. Schema*$(\vec{x})$ *returns the schema of that tuple, i.e. it returns the set of attributes whose values constituted the tuple.* Schema() *can be overloaded by providing its parameter as a* relation/view. *In this case, it would return the schema of the relation.*

Now we define the semantics of peer mappings. We already mentioned that a mapping assertion of a peer mapping is of the form:

$$\forall\vec{x}(\exists\vec{y}\varphi(\vec{x}, \vec{y}) \overset{MT}{\rightsquigarrow} s(\vec{x}))$$

Let us assume that the above assertion defines an external source s of peer P_i in terms of the peer schema of peer P_j. We say that an interpretation of the schema of P_i and P_j satisfies the assertion if that interpretation satisfies the following FOL formula

$$\forall\vec{x}\forall\vec{z}(\exists\vec{y}(\varphi(\vec{x}, \vec{y}) \wedge \vec{z} \in MT[\vec{x}]) \equiv s(\vec{z}))$$

We can interpret a mapping as a definition of how the data of the external source would be instantiated by the data of other peers. The formula also tells us that before instantiating the external source, data is converted using the corresponding mapping tables of MT. However, if there is no data-level heterogeneity, no mapping table is needed. In that case, an empty mapping table ϕ is used in the assertion. In that case, a peer mapping is represented as $\forall\vec{x}(\exists\vec{y}\varphi(\vec{x}, \vec{y}) \overset{\phi}{\rightsquigarrow} s(\vec{x}))$ which satisfies the FOL formula $\forall\vec{x}(\exists\vec{y}\varphi(\vec{x}, \vec{y}) \equiv s(\vec{x}))$.

Example 18 *Let us modify the peer mapping of Figure 6.2 by a bi-level mapping assertion* $P2P_{21}^{b}$ *which is expressed as follows.*

$$P2P_{21}^{b} : \forall xz(\exists y(P2E(x,y) \wedge P2J(y,z)) \overset{mt}{\rightsquigarrow} E(x,z))$$

The new scenario is shown in Figure 6.3(a). To satisfy the assertion, the following formula has to be satisfied.

$$\forall rtt'(\exists s(P2E(r,s) \wedge P2J(s,t) \wedge (r,t') \in mt[(r,t)]) \equiv E(r,t'))$$

Given the source database in Figure 6.3(a), for satisfying the above formula, the extension of the intensional source $E(Name, Position)$ *has to be as shown in Figure 6.3(b). Figure 6.3(b) also shows the ultimate extension of the intentional relation* $N\_P(Name, Position)$ *in the peer schema of peer* P_1*. Consequently, in response to the query* $q_1 : \{x|N\_P(x,'CEO')\}$ *to peer* P_1*, the PDMS will return the result* {***Rameen, Alina***}*.*

6.3.3 Semantics of a P2P System

We give the semantics of a P2P system Π in terms of a set of models that satisfy the local and peer mappings of Π. Let a source database $\mathcal{D}$ for Π be a disjoint union of a set of local databases in each peer P_i of Π. Given a source database $\mathcal{D}$ for Π, the set of models of Π relative to $\mathcal{D}$ is:

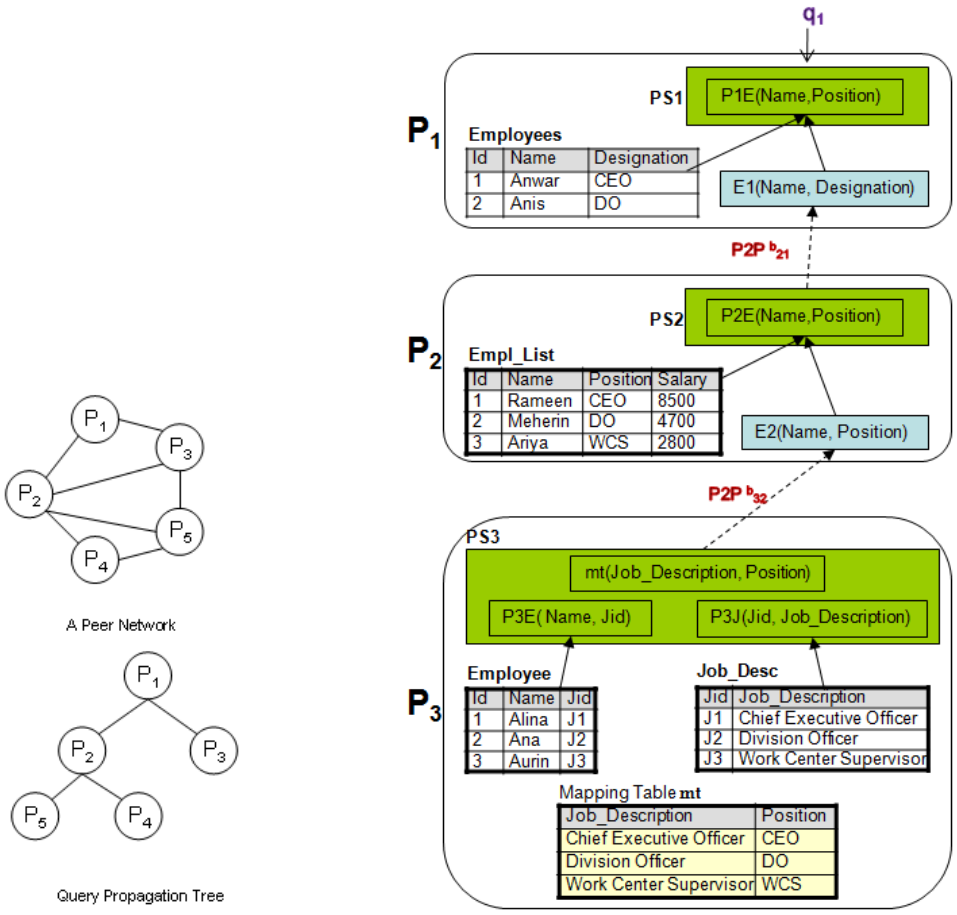
$$sem^{\mathcal{D}}(\Pi) = \{\mathcal{I}|\mathcal{I} \text{ is a finite model of all peer theories } F_i \text{ relative to } \mathcal{D}, \text{ and } \mathcal{I} \text{ satisfies all peer mappings}\}$$

Given a query q of arity k posed to a peer P_i of Π, and a source database $\mathcal{D}$, the certain answers to q relative to $\mathcal{D}$ are

$$ans(q, \Pi, \mathcal{D}) = \{\vec{t}|\vec{t} \in q^{\mathcal{I}}, \text{ for every } \mathcal{I} \in sem^{\mathcal{D}}(\Pi)\}$$

6.4 Query Evaluation

We adopt the gossiping mechanism for query execution [38]. When a query is posed to a peer it is executed in the local database of the peer and is forwarded to the acquaintances of the current peer. Whenever a peer gets a query forwarded by another peer, it executes and forwards the query to it's acquaintances causing in turn the further propagation of the query. This process continues until all the reachable peers have been processed or a fixed number of propagations of the initial query has occurred. Considering how a query propagates through the peers of a peer network, the peer network can be converted to a shortest path spanning tree. We call this tree a *propagation tree*. Figure 6.4(a) shows a peer network and the corresponding query propagation tree with respect to peer P_1 where the query originates. A single peer is visited only once per execution of a query, i.e. in the propagation tree of the query a peer can appear at most once. Note that the propagation tree is constructed dynamically and depends on how the peers are acquainted to one another. When a peer forwards the query to one of its acquaintance, the former is said to be the ancestor of the later in the tree. We assume that each query is defined w.r.t. the schema of a single peer (called *initial peer*). The initial peer executes the query in a straight forward fashion and also propagates it to its acquainted peers. At the time of propagation, the query is transformed to get compatible with the peer schema of the acquaintance peer. The local execution of the query and it's transformation and propagation to other peers goes on parallelly. The transformation of the query is unfolding the definition of the external sources defined in the peer mappings between two peers. Each peer in the propagation path gets the query result from it's descendant, merges the result with it's own result, and then back propagates to it's ancestor. When a result is back propagated to a peer, it is transformed according to the peer schema of the recipient peer, so that it can be merged with the result of the local result. Merging two results is done by simply taking the inner union of them. Since the results may need some semantic translation, instead of directly returning them to the initial peer, they are



(a) Peer network and query propagation tree

(b) Merging example

Figure 6.4: Query propagation tree and Merging example

returned along the reverse way of query propagation. We borrowed the concept of *local query* and *global query* from [38]. A local query is executed using the data in the local peer. On the other hand, a global query uses the peer network to get the amalgamated result of the locally retrieved data (i.e. the result of the local queries). We now formalize the above notions.

Consider a P2P system $\Pi = (\mathcal{P}, \mathcal{M})$ with $\mathcal{P} = \{P_1, P_2, ..., P_n\}$. Assume I_i be an instance of P_i. Let $TranQ(q, M_i^j)$ be a function that translates the query q of peer P_i to the vocabulary (both data and schema) of peer schema of P_j according to the peer mappings M_i^j between P_i and P_j. Now a global query q_{P_i} with respect to a specific query

q_i posed on the peer P_i is defined as a set of queries $\{q_i^1, q_i^2, \ldots, q_i^n, \}$ where q_i^j is defined as follows:

$$q_i^j = \begin{cases} q_i & \text{If } i = j \\ TranQ(q_i, M_i^j) & \text{If } i \neq j \text{ and there exists a mapping } M_i^j \text{ between peers } P_j \text{ and } P_i \\ TranQ(q_i^k, M_k^j) & \text{If } i \neq j \text{ and } P_i \text{ is indirectly mapped to } P_j \text{ through some intermediate peers and } P_k \text{ be the immediate predecessor of } P_j \text{ in the propagation path} \end{cases}$$

Let $TranV(V, M_i^j)$ be a function that translates the view V of the peer P_i to the vocabulary of the peer P_j using the peer mapping M_i^j. Moreover, let $q_i^j(I_j)$ denotes the view resulted from application of query q_i^j to the local instance I_j of peer P_j. Semantics of q_i^j is given above. The result of the global query $Result(q_{P_i})$ is defined as $Result(q_{P_i}) = \cup_{j=1}^{n} I_{i,j}^i$. Where $I_{i,j}^k$ is defined as follows:

$$I_{i,j}^k = \begin{cases} q_i^j(I_j) & \text{If } j = k \\ TranV(q_i^j(I_j), M_j^i) & \text{If } i = k \neq j \text{ and there exists a mapping } M_j^i \text{ between peers } P_j \text{ and } P_i \\ TranV(I_{i,j}^l, M_l^k) & \text{If } i \neq j \neq k \text{ and } P_j \text{ is indirectly mapped to } P_i \text{ through some intermediate peers and } P_l \text{ be the immediate predecessor of } P_i \text{ in the propagation path} \end{cases}$$

Since all the local views are ultimately transformed according to the schema of the

initial peer P_i, an inner union can be taken on the translated results. Our approach differs from the approach of [**?**] where outer union is taken among the local views of peers which provides far less meaningful information. Obviously, computation time can be saved if two or more mappings can be composed together to generate a direct mapping between two peers. We will address this issue in a separate work.

Example 19 *Consider the example in Figure 6.4(b). The query q_1 and the peer mappings $P2P^b_{21}$ and $P2P^b_{32}$ are defined as follows:*

$$q_1 : \{x | P1E(x,'CEO')\}$$
$$P2P^b_{21} : \forall xy(P2E(x,y) \rightsquigarrow E1(x,y))$$
$$P2P^b_{32} : \forall xw(\exists yz(P3E(x,y) \wedge P3J(y,z) \wedge mt(z,w)) \rightsquigarrow E1(x,w))$$

Local mappings are not shown and they are obvious. Now when q_1 is posted against the peer schema of P_1, a global query $q_{P_1} = \{q^1_1, q^2_1, q^3_1\}$ is formed where:

$$q^1_1 : \{x | P1E(x,'CEO')\}$$
$$q^2_1 : \{x | P2E(x,'CEO')\}$$
$$q^3_1 : \{x | \exists yz(P3E(x,y) \wedge P3J(y,z) \wedge mt(z,'CEO'))\}$$

The queries q^1_1,q^2_1, q^3_1 are executed on the local instances of the peers P_1, P_2, and P_3, respectively to produce the results $I^1_{1,1}$,$I^2_{1,2}$ and $I^3_{1,3}$ as follows:

Name
Anwar

(a) $I^1_{1,1}$

Name
Rameen

(b) $I^2_{1,2}$

Name
Alina

(c) $I^3_{1,3}$

Figure 6.5: Local results of q^1_1,q^2_1 and q^3_1

Peer P_3 converts $I^3_{1,3}$ to $I^2_{1,3}$ using the mapping M^2_3 and sends it to peer P_2. P_2 converts $I^2_{1,2}$ and $I^2_{1,3}$ to $I^1_{1,2}$ and $I^1_{1,3}$, respectively using the mapping M^1_2 and sends it to peer P_1. Note that mapping M^j_i is the inverse of mapping M^i_j. The attribute `Name` *is common in all the peers and so the converted views remains the same as the original views. P_1 combines the views $I^1_{1,1}$, $I^1_{1,2}$, and $I^1_{1,3}$ to produce the final result as {**Anwar**, **Rameen**, **Alina**}.*

6.5 Bi-Level Mapping and Tableaux

Like schema mappings, bi-level mappings can also be expressed by tabular representation. This tabular representation can be used not only for optimizing the mappings but also for query translation among different peers of a p2p data sharing system. In Section 6.5.1 we present an algorithm that converts a bi-level mapping into a tabular form. Then in Section 6.5.2 we present another algorithm that utilizes this tabular form for translating queries. Finally, in Section 6.5.3 we show that the translated queries produced by our algorithm are sound and complete.

6.5.1 Representing Bi-Level Mapping by Tableaux

Let $\mathfrak{m} : \forall\vec{x}(\exists\vec{y}\varphi(\vec{x},\vec{y}) \overset{MT}{\rightsquigarrow} E(\vec{x}))$ be an arbitrary bi-level mapping between peers P_1 and P_2, where $\vec{x} = (x_1, \ldots, x_p)$ and $\vec{y} = (x_1, \ldots, x_q)$. Remember that $\varphi(\vec{x},\vec{y})$ must have relations only from the peer P_1 and $E(\vec{x})$ must be an external source of peer P_2. A MAT $\mathcal{T}^E_{\varphi_{MT}}$ for $\mathfrak{m}$ is a partitioned table whose left hand side partition is the summary-free source tableau $\mathcal{T}_{\varphi_{MT}}$ (representing the query $\{\vec{x'}|\forall\vec{x'}\exists\vec{y'}\varphi_{MT}(\vec{x'},\vec{y'})\}$) and right hand side partition is the summary-free target tableau $\mathcal{T}_E$ (representing the query $\{\vec{x'}|\forall\vec{x'}E(\vec{x'})\}$). We have,

$$\vec{x'} \equiv (\vec{x} - \vec{P} + \vec{Q})$$
$$\vec{y'} \equiv (\vec{y} + \vec{P})$$
$$\vec{P} \equiv \bigcup_{mt(\vec{p},\vec{q})\in MT} \vec{p}$$
$$\vec{Q} \equiv \bigcup_{mt(\vec{p},\vec{q})\in MT} \vec{q}$$
$$\varphi_{MT} \equiv \varphi(\vec{x'},\vec{y'}) \bigwedge_{mt(\vec{p},\vec{q})\in MT} mt(\vec{p},\vec{q})$$

An algorithm for computing MAT $T^E_{\varphi_{MT}}$ for the mapping assertion $\mathfrak{m} : \forall\vec{x}(\exists\vec{y}\varphi(\vec{x},\vec{y}) \overset{MT}{\rightsquigarrow} E(\vec{x}))$ is shown in Figure 6.5.1. Procedure $BLM2MAT(\mathfrak{m})$ initially uses the Procedure $MA2MAT()$ of Figure 3.2 to create a MAT $\mathcal{T}^E_\varphi = \langle\mathcal{T}_\varphi, \mathcal{T}_E\rangle$ for the mapping assertion

$\sigma : \forall\vec{x}(\exists\vec{y}\varphi(\vec{x},\vec{y}) \rightsquigarrow E(\vec{x}))$ (without considering the mapping tables). Then it augments $\mathcal{T}_\varphi$ by adding each mapping table $mt \in MT$ to the rows of $\mathcal{T}_\varphi$. When a row is added to $\mathcal{T}_\varphi$ for a mapping table $mt(\vec{p},\vec{q}) \in MT$, the following steps are taken:

1. A new row is added to $\mathcal{T}_\varphi$ with the tag mt.
2. New columns are added to $\mathcal{T}_\varphi$ for the attributes $\vec{q}$.
3. For some row R, if $T_\varphi[R][\vec{p}]$ is a distinguished variable $\vec{a_i}$, then $T_\varphi[R][\vec{q}]$ is assigned $\vec{a_i}$. Remember that $T_\varphi[R][\vec{p}]$ denotes the variable of T_φ having row position R and column position $\vec{p}$.
4. The column $\vec{q}$ of $\mathcal{T}_\varphi$ is unified by putting the same non-distinguished variable in column position $\vec{q}$ for each row of $\mathcal{T}_\varphi$.

When all the modification is done to $\mathcal{T}_\varphi$, we call the modified $\mathcal{T}_\varphi$, $\mathcal{T}_{\varphi_{MT}}$. Finally, $\mathcal{T}_{\varphi_{MT}}$ is merged with $\mathcal{T}_E$ to form the MAT $\mathcal{T}^E_{\varphi_{MT}} = \langle\mathcal{T}_{\varphi_{MT}}, \mathcal{T}_E\rangle$.

Example 20 *Consider the P2P data sharing system* $\Pi =< \mathcal{P}, \mathcal{M} >$ *of Figure 6.3. We have,*

$\mathcal{P} = \{P_1, P_2\}$

$\mathcal{M} = \{\mathrm{m}\}$

$\mathrm{m} = \forall xz(\exists y(P2E(x,y) \wedge P2J(y,z)) \overset{MT}{\rightsquigarrow} E(x,z))$

$MT = \{mt(Job\_Description, Position)\}$

When the algorithm $BLM2MAT()$ *of Figure 6.5.1 is applied to* m*, initially it creates the MAT* $\mathcal{T}^\psi_\varphi = \langle\mathcal{T}_\varphi, \mathcal{T}_\psi\rangle$ *of Figure 6.7(a) using the algorithm* $MA2MAT()$ *of Figure 3.2 on the mapping assertion* $\sigma \equiv \forall xz(\exists y(P2E(x,y) \wedge P2J(y,z)) \rightsquigarrow E(x,z))$. $\mathcal{T}_\varphi$ *is then modified step by step for the mapping table mt. Figure 6.7(b) shows the new* $\mathcal{T}^\psi_\varphi$ *after a new row mt and a new column is added to* $\mathcal{T}_\varphi$. *Figure 6.7(c) shows that a fresh non-distinguished variable* b_2 *has been assigned to* $\mathcal{T}_\varphi[mt][Job\_Description]$. *Figure 6.7(d) depicts that distinguished variable* a_2 *of* $\mathcal{T}_\varphi[mt][Job\_Description]$ *is copied to*

Procedure BLM2MAT($\mathfrak{m}$)
Input : A Bi-Level mapping $\mathfrak{m} : \forall\vec{x}(\exists\vec{y}\varphi(\vec{x},\vec{y}) \overset{MT}{\rightsquigarrow} E(\vec{x}))$
Output: A MAT $T^{\psi}_{\varphi_{MT}}$ representing $\mathfrak{m}$
begin
Let $\sigma \equiv \forall\vec{x}(\exists\vec{y}\varphi(\vec{x},\vec{y}) \rightsquigarrow E(\vec{x}))$
$\mathcal{T}^{E}_{\varphi} = \langle\mathcal{T}_{\varphi}, \mathcal{T}_{\mathcal{E}}\rangle = MA2MAT(\sigma)$
for each $mt[\vec{p},\vec{q}] \in MT$ **do**
add a column $\vec{q}$ to T_{φ}
add a new row to T_{φ} with tag mt
$\mathcal{T}_{\varphi}[mt][\vec{p}] \leftarrow FreshNonDistinguishedV()$
for each row $R \in \mathcal{T}_{\varphi}.Tags$ **do**
if $IsDistinguished(\mathcal{T}_{\varphi}[R][\vec{p}])$ **then**
$temp \leftarrow \mathcal{T}_{\varphi}[R][\vec{p}]$
$\mathcal{T}_{\varphi}[R][\vec{p}] \leftarrow \mathcal{T}_{\varphi}[mt][\vec{p}]$
$\mathcal{T}_{\varphi}[mt][Q] = temp$
endif
endfor
endfor
$\mathcal{T}^{\psi}_{\varphi_{MT}} \leftarrow \langle\mathcal{T}_{\varphi}, \mathcal{T}_{\psi}\rangle$
return $\mathcal{T}^{\psi}_{\varphi_{MT}}$
end

Figure 6.6: Procedure BLM2MAT()

$\mathcal{T}_{\varphi}[mt][Position]$. *Finally, all the symbols of the column Job_Descrion of* $\mathcal{T}_{\varphi}$ *is replaced by the same variable* b_2 *as shown in Figure 6.7(e).* □

Definition 32 *Let* $m : \forall\vec{x}(\exists\vec{y}\varphi(\vec{x},\vec{y}) \overset{MT}{\rightsquigarrow} E(\vec{x}))$ *be a bi-level mapping and* I *be a database instance having the relations in* $\varphi(\vec{x},\vec{y})$ *and* MT. $\mathbb{CI}(m, I)$ *is a set of all consistent instances, w.r.t.* m *and* I, *such that for any instance* $I' \in \mathbb{CI}(m, I)$ *the following conditions hold*

- I' *has the schema of* E
- I *and* I' *together satisfies the formula*
  $\forall\vec{x}\forall\vec{z}((\exists\vec{y}\varphi(\vec{x},\vec{y}) \wedge \vec{z} \in MT[\vec{x}]) \equiv E(\vec{z}))$

	Name	*Jid*	*Job_Description*	*Name*	*Position*	
P2E	a_1	b_1		a_1	a_2	*E*
P2J		b_1	a_2			

(a) Initial MAT $\mathcal{T}_\varphi^\psi = \langle \mathcal{T}_\varphi, \mathcal{T}_\psi \rangle$ excluding the mapping tables

	Name	*Jid*	*Job_Description*	*Position*	*Name*	*Position*	
P2E	a_1	b_1			a_1	a_2	*E*
P2J		b_1	a_2				
mt							

(b) New rows and columns added to $\mathcal{T}_\varphi$

	Name	*Jid*	*Job_Description*	*Position*	*Name*	*Position*	
P2E	a_1	b_1			a_1	a_2	*E*
P2J		b_1	a_2				
mt			b_2				

(c) Fresh non-distinguished symbol added to the left column of mt

	Name	*Jid*	*Job_Description*	*Position*	*Name*	*Position*	
P2E	a_1	b_1			a_1	a_2	*E*
P2J		b_1	a_2				
mt			b_2	a_2			

(d) Distinguished symbol is copied to the right column of mt

	Name	*Jid*	*Job_Description*	*Position*	*Name*	*Position*	
P2E	a_1	b_1			a_1	a_2	*E*
P2J		b_1	b_2				
mt			b_2	a_2			

(e) Final MAT after symbols in the left column of mt being unified

Figure 6.7: An example showing the steps for converting a bi-level mapping to MAT

Proposition 11 *Let* $\mathrm{m} \equiv \forall\vec{x}(\exists\vec{y}\varphi(\vec{x},\vec{y}) \overset{MT}{\rightsquigarrow} E(\vec{x}))$ *be a bi-level mapping.* $\mathcal{T}^{E}_{\varphi_{MT}} = \langle\mathcal{T}_{\varphi_{MT}}, \mathcal{T}_E\rangle$ *be the MAT for* m. *Let* I *be an instance of the relations of* φ. *It is valid that*

$$\forall_{I'}((I' \in \mathbb{CI}(\mathrm{m}, I)) \Rightarrow (\mathcal{T}_{\varphi_{MT}}(I) \equiv \mathcal{T}_E(I')))$$

6.5.2 Query Translation by Tableaux

When a query or sub-query is placed on an external source E of a peer, the query has to be translated according to the schema of the source peer to which E is mapped. The translated query is then passed to the source peer to get the result back to the peer where the query was originally posted. Tableaux representation of the bi-level mappings can help this query translation process. In algorithm $QueryTranslation()$ of Figure 6.8 we show such a technique. The algorithm $QueryTranslation(q, \mathrm{m}_{\varphi\rightsquigarrow E})$ first converts the given query q into a tableau $\mathcal{T}_q$ and the mapping $\mathrm{m}_{\varphi\rightarrow E}$ into a MAT $\mathcal{T}^{E}_{\varphi} = \langle\mathcal{T}_{\varphi}, \mathcal{T}_E\rangle$. Remember that $\mathcal{T}_E$ is the summary-free tableaux for a query which produces all the tuples of the external source E. The algorithm $QueryTranslation()$ then matches the tableaux $\mathcal{T}_q$ to $\mathcal{T}_E$ and find out the differences between them. Using those differences $\mathcal{T}_{\varphi}$ is modified. Afterward, a summary is added to $\mathcal{T}_{\varphi}$. Finally, the modified $\mathcal{T}_{\varphi}$ is converted back to a query which is the desired translated query.

Example 21 *Again, consider the P2P data sharing system* $\Pi =< \mathcal{P}, \mathcal{M} >$ *of Figure 6.3. If a query* $q_1 : \{x|N\_P(x,"CEO")\}$ *is posed at* P_1. *A sub-query* $q : \{x|E(x,'CEO')\}$ *is posed to the external source* E. *We know from example 16 that the proper translation of* q *(w.r.t. the mapping* $\mathrm{P2P}^{\mathrm{b}}_{21} : \forall xz(\exists y(P2E(x,y) \wedge P2J(y,z)) \overset{\{mt\}}{\rightsquigarrow} E(x,z))$ *between* E *and the peer schema of* P_2*) is* $q' : \{x \mid \forall x\exists yz(P2E(x,y) \wedge P2J(y,z) \wedge mt(z,`CEO'))\}$.

In Figure 6.9 we show step by step how this translation is done by the algorithm $QueryTranslation()$. *Query* q *is translated to a tableaux* $\mathcal{T}_q$ *as in Figure 6.9(b). The bi-level mapping* $P2P^{b}_{21}$ *is converted to a MAT* $\mathcal{T}^{E}_{\varphi} = \langle\mathcal{T}_{\varphi}, \mathcal{T}_E\rangle$ *as shown in Figure 6.9(c).* $\mathcal{T}_{\varphi}$ *is separated and is termed as* $\mathcal{T}_{q'}$ *as in Figure 6.9(d). Figure 6.9(e) depicts* $\mathcal{T}_{q'}$ *with an*

Algorithm QueryTranslation$(q, m_{S \rightsquigarrow E})$
Input: A query q on external source E and a bi-level mapping
$m_{S \to E}$ from schema S to external source E.
Output: A query q' on schema S which is a sound and complete
translation of q w.r.t. $m_{S \to E}$
begin
$\mathcal{T}_q \leftarrow$ **ConvertQueryToTableau**(q)
$\mathcal{T}_\varphi^E \leftarrow \langle \mathcal{T}_\varphi, \mathcal{T}_E \rangle =$ **BLM2MAT**$(m_{S \rightsquigarrow E})$
$\mathcal{T}_{q'} \leftarrow \mathcal{T}_\varphi$
add an empty summary to $\mathcal{T}_{q'}$
for each column $C \in \mathcal{T}_q.Columns$ **do**
temp=$\mathcal{T}_\psi.Rows[E][C]$
for each row $R \in \mathcal{T}_{q'}.Tags$ **do**
for each column $Col \in \mathcal{T}_{q'}.Columns$ **do**
if $\mathcal{T}_{q'}[R][Col] = temp$ **then**
if $IsNotBlank(\mathcal{T}_q[Summary][C])$ **then**
if $IsConstant(\mathcal{T}_q[E][C])$ **then**
$\mathcal{T}_{q'}[R][Col] \leftarrow \mathcal{T}_q[E][C]$
$\mathcal{T}_{q'}[Summary][Col] \leftarrow \mathcal{T}_q[E][C]$
else
$\mathcal{T}_{q'}[Summary][Col] \leftarrow temp$
endif
else
if $IsConstant(\mathcal{T}_q[E][C])$ **then**
$\mathcal{T}_{q'}[R][Col] = \mathcal{T}_q[E][C]$
else
$\mathcal{T}_{q'}[R][Col] = FreshNonDistinguished()$
endif
endif
endif
endfor
endfor
endfor
$q' \leftarrow$ ***ConvertTableauToQuery***$(\mathcal{T}_{q'})$
return q'
end

Figure 6.8: Algorithm for query translation

empty summary added to it. Each column of $\mathcal{T}_q$ is compared to the corresponding column of $\mathcal{T}_E$ and whenever an mismatch is found, $\mathcal{T}_{q'}$ is changed accordingly. Summary of $\mathcal{T}_{q'}$ is also changed according to the summary of $\mathcal{T}_q$. Final $\mathcal{T}_{q'}$, after modifying summary and rows, is shown in Figure 6.9(f). $\mathcal{T}_{q'}$ is then converted to the query q' as shown in Figure 6.9(g).

6.5.3 Sound and Complete Translation of Queries

Definition 33 *Let q and q' be two queries over schema S and S', respectively. $m : \forall\vec{x}(\exists\vec{y}\varphi(\vec{x},\vec{y}) \overset{MT}{\rightsquigarrow} E(\vec{x}))$ be a bi-level mapping between S' and S. Given an instance I' of S', q' is said to be a sound and complete translation of q w.r.t m if*

$$\forall_I((I \in \mathbb{CI}(m, I')) \Rightarrow (q(I) \equiv q'(I')))$$

Theorem 23 *Let $m : \forall\vec{x}(\exists\vec{y}\varphi(\vec{x},\vec{y}) \overset{MT}{\rightsquigarrow} E(\vec{x}))$ be a bi-level mapping and q be a query on E. $q' = QueryTranslation(q, m_{S\to T})$ implies that query q' is a sound and complete translation of the query q w.r.t. m .*

Proof: Let I and I' be two database instances such that $I \in \mathbb{CI}(m, I')$. According to the definition 33, q' to be a sound and complete translation of q, $q(I) \equiv q'(I')$ should hold. To prove theorem 23 we have to prove that $q(I) \equiv q'(I')$.

Let $T^E_{\varphi_{MT}} = \langle \mathcal{T}_{\varphi_{MT}}, \mathcal{T}_E \rangle$ be the MAT for m. Let q_E be the query corresponding to the tableau $\mathcal{T}_E$ and T_q be the tableaux for the query q. When posed on E, q_E returns all tuples of E and q returns a subset of E. So, we have $q \subset q_E$. Now, according to the theory 1 there exist a homomorphism θ_1 between $\mathcal{T}_E$ and $\mathcal{T}_q$ such that $\theta_1(\mathcal{T}_E) = \mathcal{T}_q$. Again, in the $QueryTranslation()$ algorithm $\mathcal{T}_{\varphi_{MT}}$ is converted to $\mathcal{T}_{q'}$ by converting some distinguished variables to constants. So, there exist another homomorphism θ_2 between $\mathcal{T}_{\varphi_{MT}}$ and $\mathcal{T}_{q'}$ such that $\theta_2(\mathcal{T}_{\varphi_{MT}}) = \mathcal{T}_{q'}$. $QueryTranslation()$ works in a way such that these two homomorphisms are same, i.e. $\theta_1 = \theta_2$.

$q \equiv \{n \mid \forall n(E(n, `CEO'))\}$

(a) Query q

	$Name$	$Position$
	a_3	
E	a_3	$`CEO'$

(b) Tableaux $\mathcal{T}_q$ for q

	$Name$	Jid	$Job\_Description$	$Position$	$Name$	$Position$	
$P2E$	a_1	b_1					
$P2J$		b_1	b_2		a_1	a_2	E
mt			b_2	a_2			

(c) MAT $\mathcal{T}_\varphi^E = \langle \mathcal{T}_\varphi, \mathcal{T}_E$ for the mapping $P2P_{21}^b$ of Figure 6.3(a)

	$Name$	Jid	$Job\_Description$	$Position$
$P2E$	a_1	b_1		
$P2J$		b_1	b_2	
mt			b_2	a_2

(d) Initial $\mathcal{T}_{q'} = \mathcal{T}_\varphi$

	$Name$	Jid	$Job\_Description$	$Position$
$P2E$	a_1	b_1		
$P2J$		b_1	b_2	
mt			b_2	a_2

(e) Empty Summary added to $\mathcal{T}_{q'}$

	$Name$	Jid	$Job\_Description$	$Position$
	a_1			
$P2E$	a_1	b_1		
$P2J$		b_1	b_2	
mt			b_2	$`CEO'$

(f) Final $\mathcal{T}_{q'}$ after modifying Summary and Rows

$q' \equiv \{x \mid \forall x \exists yz(P2E(x, y) \wedge P2J(y, z) \wedge mt(z, `CEO'))\}$

(g) Query q' obtained from $\mathcal{T}_{q'}$

Figure 6.9: Conversion of query q wrt mapping $P2P_{21}^b$ of Figure 6.3(a)

According to proposition 11, we get $\mathcal{T}_{\varphi_{MT}}(I) \equiv \mathcal{T}_E(I')$. Applying θ_1 on both sides of the equivalency we get,

$$\begin{aligned}
&\theta_1(\mathcal{T}_{\varphi_{MT}})(I') \equiv \theta_1(\mathcal{T}_E)(I) \\
\Rightarrow\ &\theta_2(\mathcal{T}_{\varphi_{MT}})(I') \equiv \theta_1(\mathcal{T}_E)(I) \\
\Rightarrow\ &\mathcal{T}_{q'}(I') \equiv \mathcal{T}_q(I) \\
\Rightarrow\ &\mathcal{T}_{q'}(I') \equiv \mathcal{T}_q(I)
\end{aligned}$$

Finally, converting the tableaux back to algebraic query representation we get the desired equivalence $q'(I')) \equiv q(I)$. □

6.6 Summary

In this chapter, we introduced a model of a peer database management system (PDMS) which uses *bi-level mapping* that combines schema-level and data-level mappings. We presented the syntax and semantics of bi-level mappings. We also provided a query evaluation procedure for the PDMS that uses bi-level mappings. Moreover, we presented algorithms for converting bi-level mappings into MATs and showed that MATs help query translation process.

Chapter 7

Evaluation and Experimental Results

In this chapter, we first present the settings of our experiments. Second, we describe the prototype that we developed for performing the experiments. The algorithms proposed in this thesis for checking equivalence of mappings and their optimization rely on the tableaux-based representation of those mappings. Therefore, third, we show the evaluation results of several algorithms that convert schema-level and bi-level mappings expressed in first order logic to mappings expressed in our proposed tableaux-based representation. Fourth, we discuss experimental results of the proposed algorithm for checking equivalence of two mappings. Fifth, we evaluate our mapping optimization algorithm. Finally, we show how optimization of mappings saves data exchange time.

7.1 Experiment Settings

The experimental environment consists of a single windows 7 machine with Intel(R) Core(TM) 2 Duo CPU 2.2GHz and 4GB of RAM. The databases are created within the MySQL 5.0 database environment. For the experiments, we built a prototype of a data exchange system that consists of a source database and a target database.

DeptEmp

$DName\_S$	$MName\_S$	EId_S
CS	$Mary$	100
CS	$Mary$	101
CS	$Mary$	102
EE	$Karim$	103
EE	$Karim$	104

dname2dname

$DName\_S$	$DName\_T$
CS	SCS
EE	EEE

(a) Source instance with the mapping table

Dept

$DName\_T$	MId_T	$MName\_T$
EEE	11	$John$
SE	12	$Baron$
SCS	13	$Rameen$
CE	14	$Anand$

Emp

EId_T	$DName\_T$
500	EEE
501	EEE
502	SE
503	SCS
504	SE
505	CE

(b) $TargetInstance$

Figure 7.1: Sample of source database instance and target database instance

Both the databases contain employee information having different schema. The source database contains a single relation $deptemp(DName\_S, MName\_S, EId\_S)$ listing the Department Name ($DName\_S$), Managaer Name ($MName\_S$) and Employee Identifier (Eid_S). The target database uses two relations $dept(DName\_T, MId\_T, MName\_T)$ and $emp(EId\_T, DName\_T)$. $dept(DName\_T, MId\_T, MName\_T)$ lists Department Name($DName\_T$), Manager Id (MId_T) and Manager Name ($MName\_T$). $emp(EId\_T$, $DName\_T)$ lists Employee Id (EId_T) and Department Name ($DName\_T$). We experimented using both schema-level mappings and bi-level mappings between the two databases. We conducted the experiments with relatively small databases. Initially, the *deptemp* relation has 1000 records, *dept* relation has 200 records and *emp* relation has 500 records. Our algorithms are scalable and will behave in the same pattern with larger databases also. Figure 7.1 shows part of the database instances. Both of the source database and the target database reside in the same machine. Therefore, there is no network delay when data is exchanged from the source to target.

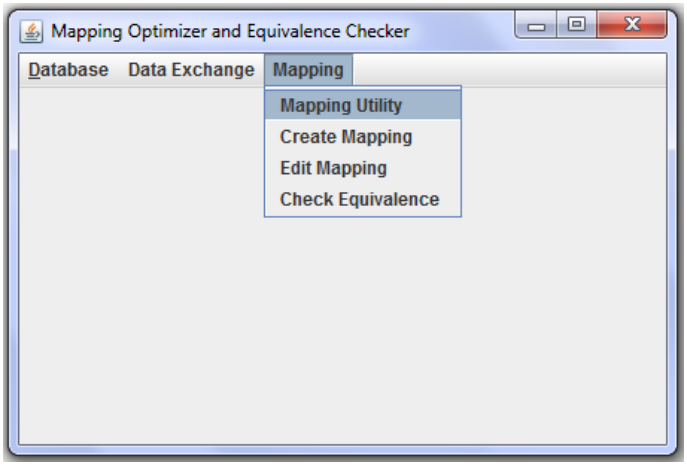


Figure 7.2: Main Screen of MOEC

7.2 Mapping Optimizer and Equivalence Checker

We performed several experiments for evaluating our proposed tableaux-based optimization of mappings and equivalence checking algorithms. We developed a software tool to carry out these experiments. We call this tool *Mapping Optimizer and Equivalence Checker* (MOEC). MOEC provides several functionalities for generating source database, target database, creating and modifying schema mappings and bi-level mappings between the source and the target databases, and populates the databases with synthetic data. It also provides a general framework for exchanging data between the source database and the target database using the predefined mappings. MOEC, developed using Java, is a prototype with a library of Java classes. It is developed using Java programming language for the portability and ease of extensibility. The tool uses real world clock (system clock time obtained using the Java function System.nanoTime) for measuring execution time of various mapping manipulation algorithms. Figure 7.2 shows the main screen of MOEC and Figure 7.3 depicts the GUI of MOEC for various manipulations on mappings.

The GUI shown in Figure 7.3 allows to create and modify mappings, save it to a file, open a mapping from a saved file, optimize a mapping, check the equivalence of

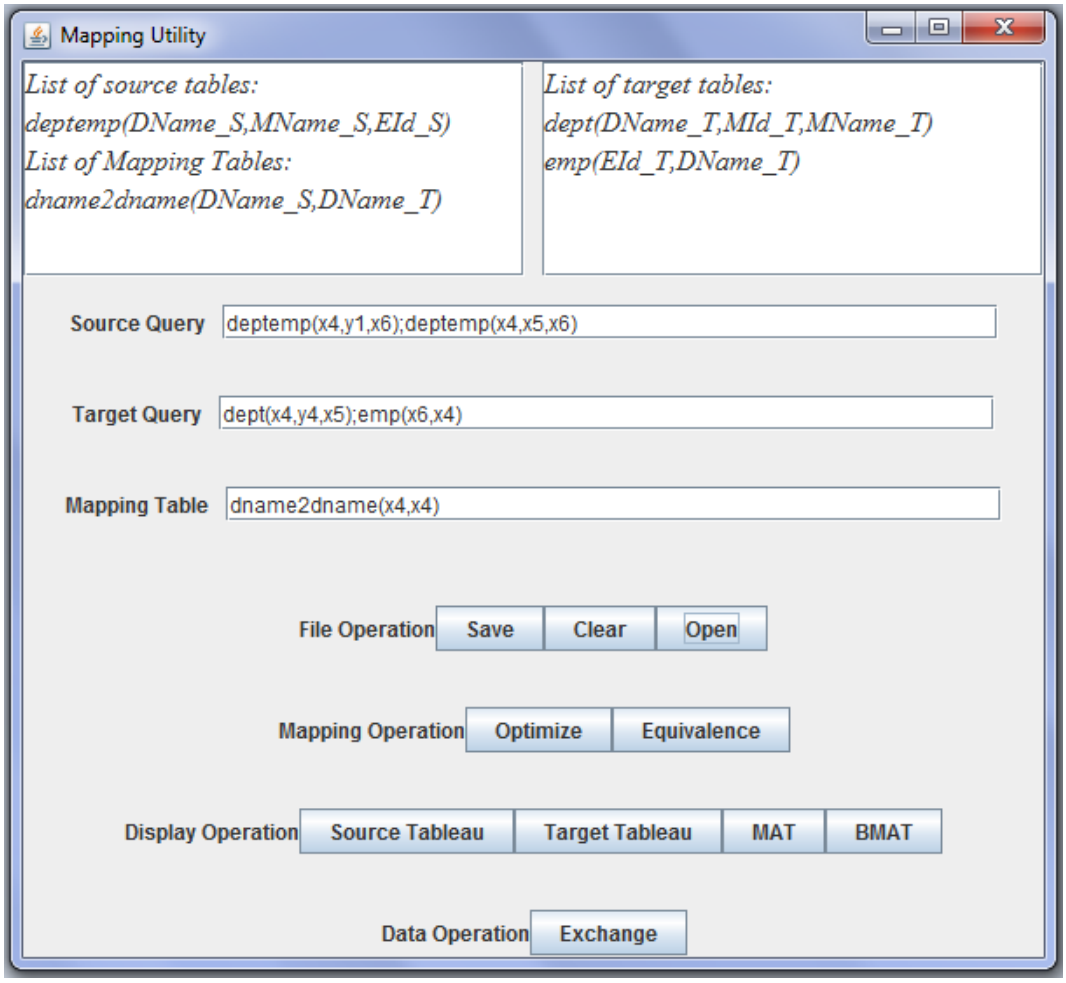


Figure 7.3: Mapping Utility GUI

$m1 : deptemp(x4, x5, x6) \rightsquigarrow dept(x4, y4, x5) \wedge emp(x6, x4)$
$m2 : deptemp(x4, y1, x6) \wedge deptemp(x4, x5, x6) \rightsquigarrow dept(x4, y4, x5) \wedge emp(x6, x4)$
$m3 : deptemp(x4, y1, y2) \wedge deptemp(x4, y1, x6) \wedge deptemp(x4, x5, x6) \rightsquigarrow$
$dept(x4, y4, x5) \wedge emp(x6, x4)$
$m4 : deptemp(x4, y1, y2) \wedge deptemp(x4, y1, x6) \wedge deptemp(x4, x5, x6) \rightsquigarrow$
$dept(x4, y4, x5) \wedge dept(x4, y4, y5) \wedge emp(x6, x4)$
$m5 : deptemp(x4, y1, y2) \wedge deptemp(x4, y1, x6) \wedge deptemp(x4, x5, x6) \rightsquigarrow$
$dept(x4, y4, x5) \wedge dept(x4, y4, y5) \wedge emp(x6, x4) \wedge emp(x6, y6)$
$m6 : deptemp(x4, x5, x6) \overset{dname2dname(x4,x4)}{\rightsquigarrow} dept(x4, y4, x5) \wedge emp(x6, x4)$

Figure 7.4: A List of Mappings

two mappings, display the tableau of the source query and target query of a mapping, shows the Mapping Assertion Tableau (MAT) and Bi-level Mapping Assertion Tableau (BMAT). For simplicity, we insert the source query, target query and the mapping tables separately to define a mapping. e.g. to create the mapping
$\forall x_1, x_2, x_3((deptemp(x_1, x_2, x_3) \overset{dname2dname(x_1,x_1)}{\rightsquigarrow} (\exists y_1(dept(x1, y1, x2) \wedge emp(x3, x1)))))$,
we insert $deptemp(x_1, x_2, x_3)$ into the *Source Query* text field, $dept(x1, y1, x2); emp(x3, x1)$ into the *Target Query* text field and $dname2dname(x_1, x_1)$ into the *Mapping Tables* text field. To avoid the use of universal and existential quantifiers, we assume that all x-variables are universally quantified and all y-variables are existentially quantified. We use semicolon to join two conjuncts instead of $\wedge$.

7.3 Experiments

We performed all the experiments five times and considered the average of the results. In the first experiment, we show how the time for converting a mapping into MAT varies with the number of conjuncts in that mapping. For this experiment, we converted the mappings of Figure 7.4 where, for simplicity, the universal and existential quantifiers are omitted. Remember that a MAT consists of two tableaux: a source tableau and a target tableau.

Mapping Assertion Tableau

	DName_S	MName_S	EId_S	DName_T	MId_T	MName_T	EId_T	DName_T	
deptemp	x4	y1	y2	x4	n4	x5			dept
deptemp	x4	y1	x6	x4	n4	n5			dept
deptemp	x4	x5	x6				x6	x4	emp

Figure 7.5: MAT for mapping $m4$

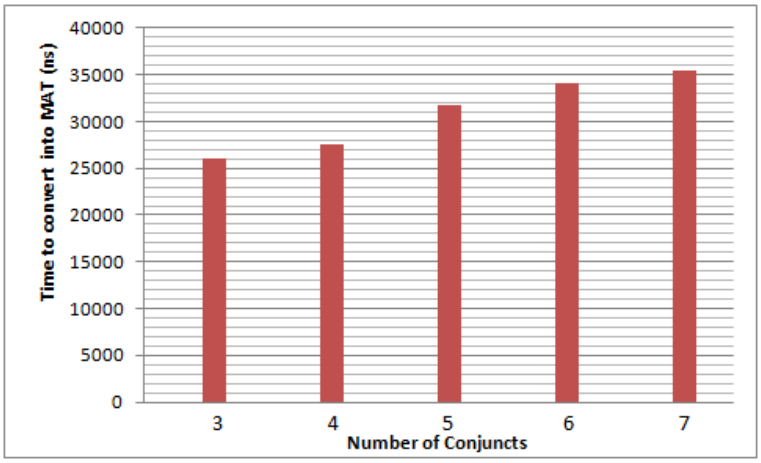


Figure 7.6: Time for converting mappings with various number of conjuncts into MATs

Figure 7.5 shows the MAT corresponding to the mapping $m4$ of Figure 7.4. Note, the left side of the bold vertical line of the MAT is the summary-free tableau corresponding to the source query of the mapping and the right side of the bold vertical line is the summary-free tableau corresponding to the target query of the mapping. The topmost row depicts the attribute names. The leftmost column shows the source relation names and the rightmost column shows the target relation names. To represent nulls on the target tableau we used n-variables (variables having name starting with n) instead of $\perp$.

In Figure 7.6, we show execution times needed for converting some mapping into MATs. Here, we observe that the time is not high and it increases with the increase of the number of conjuncts almost linearly.

In the second experiment, we show how the time for converting a bi-level mapping

Mapping Assertion Tableau

	DName_S	MName_S	EId_S	DName_S	DName_T	DName_T	MId_T	MName_T	EId_T	DName_T	
deptemp	y50	x5	x6			x4	n4	x5			dept
dname2dname				y50	x4				x6	x4	emp

Figure 7.7: MAT for bi-level mapping $m6$

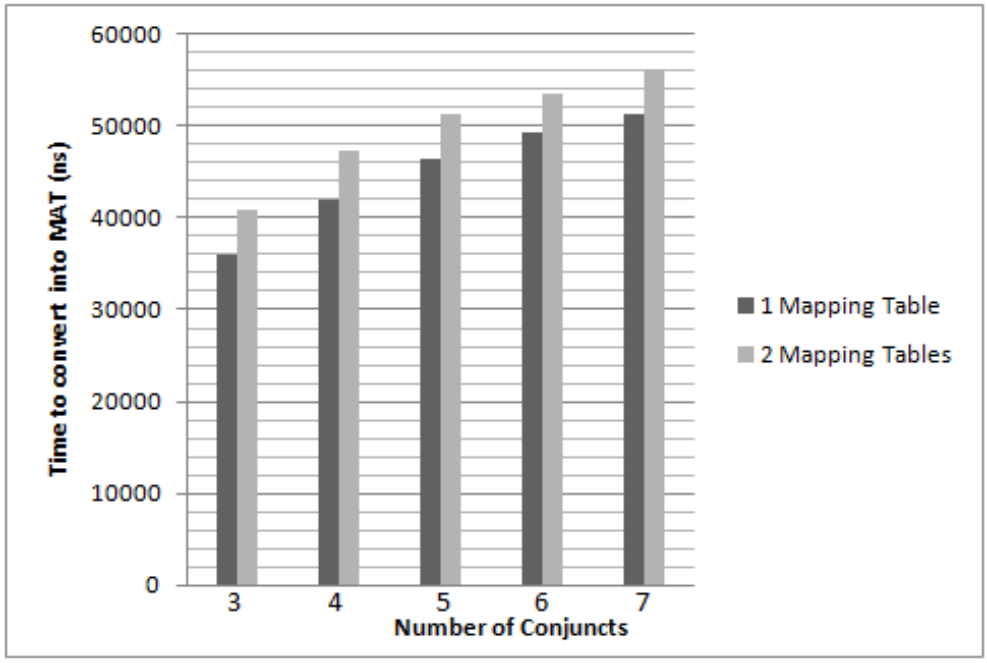


Figure 7.8: Time for converting bi-level mappings with various number of conjuncts and mapping tables into MATs

into a MAT varies with the number of conjuncts and mapping tables in that mapping. MAT for the bi-level mapping $m6$ of Figure 7.4 is shown in Figure 7.7. Note that for each mapping table a row is added in the source tableau.

In Figure 7.8, we again observe that the time needed for converting a bi-level mapping into a MAT is insignificant and the time increases with the increase of the number of conjuncts and mapping tables almost linearly.

In the third experiment we evaluate our mapping optimization algorithm. To optimize a mapping, we first convert the mapping into a MAT. Then we optimize this MAT by deleting some rows from the source tableau and the target tableau if they are subsumed

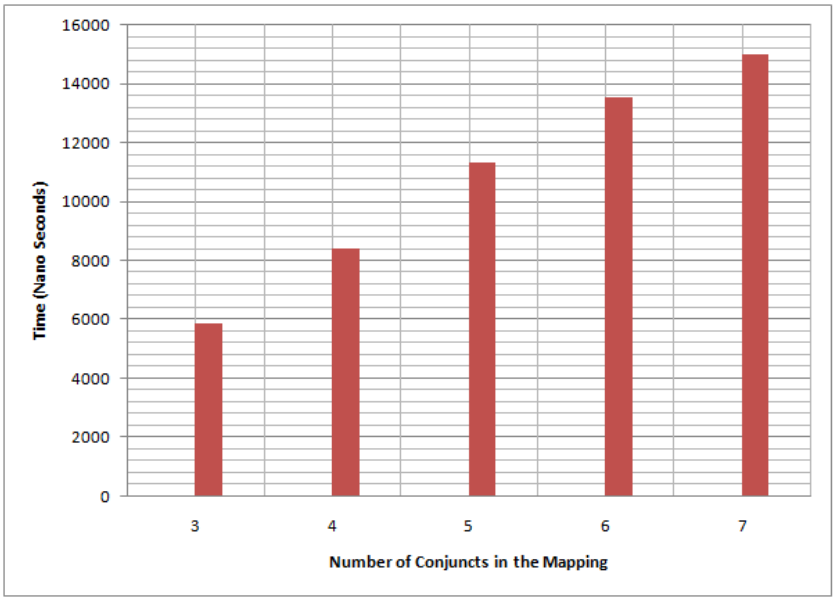


Figure 7.9: Optimization time of mappings with various number of conjuncts

by some other rows in those tableaux. Finally, we convert the MAT back to the mapping. In this way, if we optimize mapping $m4$ of Figure 7.4, we get mapping 1 of Figure 7.4. In Figure 7.9, we show times for optimizing the MATs. As expected, the execution time increases gradually with the increase of number of conjuncts. However, the change of the optimization time is not rapid, which proves the efficiency of the optimization algorithm.

With the fourth experiment, we examined the efficiency of the proposed algorithm for

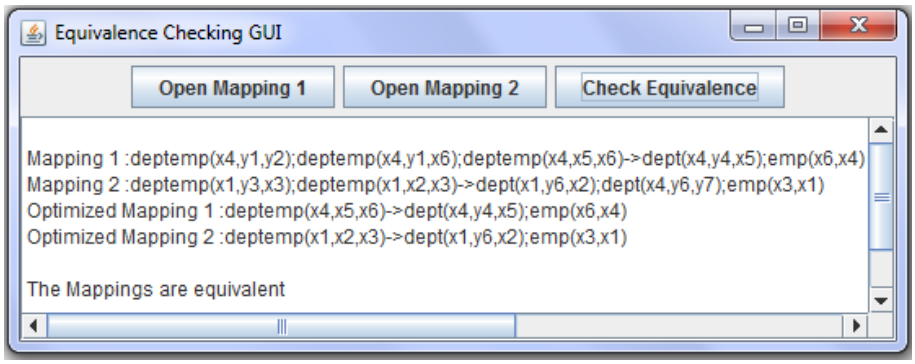


Figure 7.10: Equivalence Checking GUI

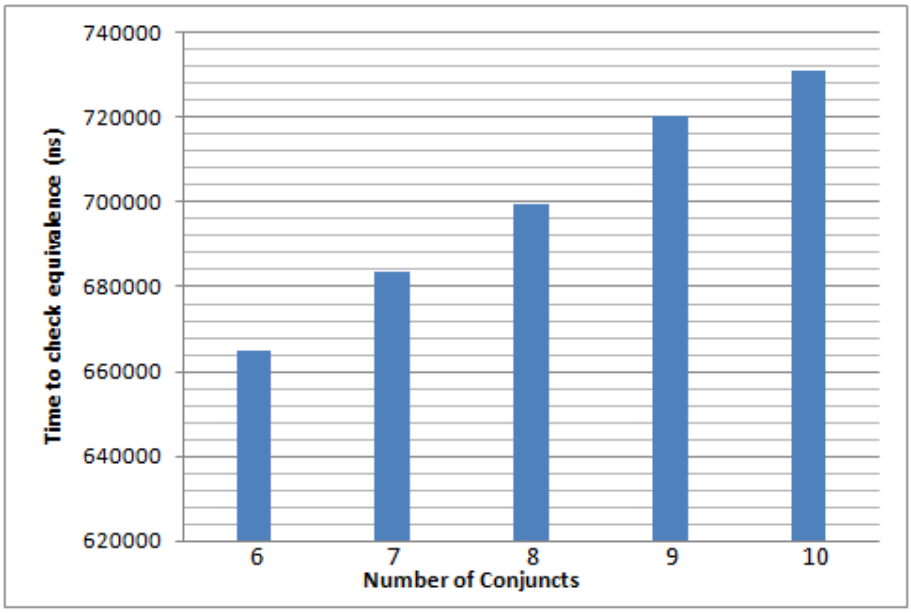


Figure 7.11: Time needed for checking equivalence of two mappings

checking equivalence of two mappings. Figure 7.10 shows a sample GUI for equivalence checking. At first, the mappings are converted into MATs. Next, those MATs are optimized using the proposed MAT optimization algorithm. Then, comparison is made between the two optimized MATs based on the matched position of the distinguished variables (i.e. x-variables). e.g. assume that the variable x_1 occupies the position [R2][C1] in source tableau and the positions [R1][C2] and [R2][C3] in the target tableau of MAT1. Now, if x_2 occupies the position [R2][C1] in the source tableau of MAT2, x_2 must also occupy the places [R1][C2] and [R2][C3] in the target tableau of MAT2 for MAT2 be equivalent to MAT1. Again, we considered the execution time as measuring criteria for evaluating the efficiency of the algorithm. Figure 7.11 shows the execution time for checking equivalence of pair of mappings with different size. X-axis of the graph shows the total number of conjuncts in the mapping pairs. From the nature of the graph, we can assume the efficiency of the equivalence checking algorithm.

Finally, in our fifth experiment, we tested how optimization of mappings affects data exchange activity between the source database and the target database. To illus-

Target Attribute	*Matched Source Attribute*
$dept.DName\_T$	$deptemp.DName\_S$
$dept.MName\_T$	$deptemp.DName\_S$
$emp.EId\_T$	$deptemp.EId\_S$
$emp.DName\_T$	$deptemp.DName\_S$

Table 7.1: STP for $m1$

trate how data exchange takes place in the database lavel, let us consider the mapping

$m1: \ deptemp(x4, x5, x6) \rightsquigarrow dept(x4, y4, x5) \wedge emp(x6, x4)$

At first, we generate a SQL-Select command for the source database as follows:

$SC1 \equiv SELECT \ a0.DName\_S, a0.MName\_S, a0.EId\_S \ FROM \ deptemp \ a0$

Then we create a list of pairs of attributes, which we call Source-Target-Pairs (STP). Each pair in the STP has a attribute from a target relation and a corresponding attribute from a source relation. The components of each pair is determined by the relative position of the distinguished variables. e.g. the variable $x4$ is placed in the 1st position of *dept* conjunct and also the 1st position of *deptemp* conjunct in the mapping $m1$. Name of the 1st attribute of *dept* relation is $DName\_T$ and the name of the 1st attribute of *deptemp* is $DName\_S$. So, these two attributes are paired and kept in the STP for $m1$. Complete STP for $m1$ is shown in Table 7.1.

The $SQL - Select$ statement $SC1$ is executed on the source database. Let us call this result set R. For each record of R, an $SQL - Insert$ statement is generated for each target conjunct of $m1$ using the information stored in the STP and the content of the record. e.g. if $([DName\_S] =' CSE', [MName\_S] =' John', [EId\_S] = 100)$ be a record in R, then the following two $SQL - Insert$ statements are generated and executed on the target database.

$INSERT \ INTO \ dept \ (DName\_T, MName\_T) \ VALUES \ ('CSE', 'John')$

$INSERT \ INTO \ emp \ (EID\_T, DName\_T) \ VALUES \ (100, 'CSE')$

When a bi-level mapping is considered, then LEFT OUTER JOIN is used to combine the mapping tables with the source relations. e.g. for the mapping

Target Attribute	*Matched Source Attribute*
$dept.DName\_T$	$dname2dname.DName\_T$
$dept.MName\_T$	$deptemp.DName\_S$
$emp.EId\_T$	$deptemp.EId\_S$
$emp.DName\_T$	$dname2dname.DName\_T$

Table 7.2: STP for $m6$

$m6 : deptemp(x4, x5, x6) \overset{dname2dname(x4,x4)}{\rightsquigarrow} dept(x4, y4, x5) \wedge emp(x6, x4)$, the $SQL - Select$ is as follows:

$SELECT * FROM\ (select\ a0.DName\_S, a0.MName\_S, a0.EId\_S\ FROM\ deptemp\ a0) e0$
$LEFT\ OUTER\ JOIN\ dname2dname\ ON\ e0.DName\_S = dname2dname.DName\_S.$

The STP for $m6$ is shown in Table 7.2.

When there exists unnecessary join of a conjunct to itself in the source query, the $SQL - Select$ command becomes complicated and it makes the data exchange operation more time consuming. e.g. for the mapping $m2 : deptemp(x4, y1, x6) \wedge deptemp(x4, x5, x6) \rightsquigarrow dept(x4, y4, x5) \wedge emp(x6, x4)$ the $SQL - Select$ command becomes:

$SC2 \equiv SELECT\ a0.DName\_S, a0.EId\_S, a1.MName\_S\ FROM\ deptemp\ a0, deptemp\ a1$
$WHERE\ a0.DName\_S = a1.DName\_S\ AND\ a0.EId\_S = a1.EId\_S$

We observe that SELECT statements $SC1$ and $SC2$ produces the same set of data with data exchange point of view, where $SC2$ is much more complicated. Now, if we optimize $m2$ before using it to exchange data, then a significant amount of data exchange time can be saved.

Moreover, unnecessary repetition of the same relation in the target query of a mapping invokes unnecessary INSERT statements on the target database. Our proposed optimization algorithm helps to remove this type of inefficiency too.

In Figure 7.12, we show the times needed for exchanging data using the mappings of Figure 7.4. For each of the mappings, we took the data two times, one before optimizing

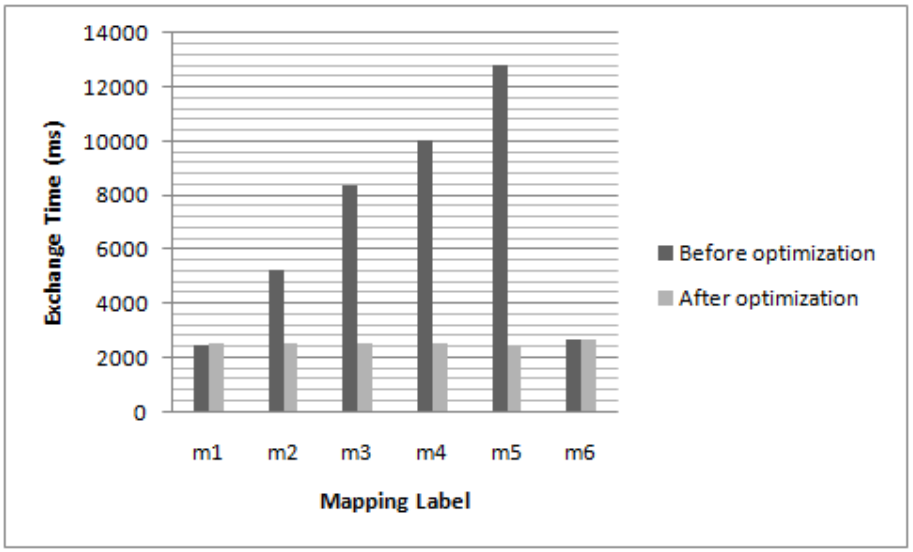


Figure 7.12: Data exchange time before and after optimization of mappings

it and the other after optimizing it. For mapping $m1$, no more optimization can be done, so both the time are same. But for other mappings we observe that exchange time before optimizing them are much higher than after optimizing them. In fact, optimization of the mappings $m2 - m5$ produces the mappings equivalent to $m1$. For this reason, after-optimization time is vary similar in cases of the mappings m1-m5. Since $m6$ is a bi-level mapping, it takes a little bit more time than $m1$. This experimental result supports our claim that optimization of the mappings saves data exchange time.

7.4 Summary

In this chapter we first presented the settings of our experiment. Next, we described the prototype MOEC that we developed for implementing and evaluating some of the tableaux-based algorithms proposed in the previous chapters of this thesis. After that, we showed evaluation results of the proposed algorithms. We also evaluated the effect of optimization of the schema mappings in reducing data exchange time. In future, we want to check the effect of optimization in data integration and data sharing scenario.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In this thesis, we proposed a tabular representation of schema mappings and utilized the structure of the new representation to optimize the schema mappings. Schema Mappings constitute the building blocks of data integration system, data exchange system and P2P data sharing system. A schema mapping describes the relationship between two database schemas at high level. *Global-and-local-as-view* (GLAV) is one of the approaches for specifying the schema mappings. Being motivated by the fact that tabular representation of queries (*tableaux*) helps optimization of queries mechanically, we took the idea of expressing GLAV mappings by a tabular representation. We call our proposed tabular representation of GLAV mappings, *Schema Mapping Tableaux* (SMT). We gave the detailed semantics of SMT.

We showed how SMT can be used to produce *minimal universal model* in a data integration system, and to produce *core* in a data exchange system. We also defined equivalence of schema mappings of two data integration systems and characterized that equivalence in terms of the equivalence of the corresponding SMTs of those schema mappings. We proposed algorithms for optimizing schema mappings. These algorithms

eliminate redundant joins from the mappings and thus make the data integration system and data exchange system more efficient.

Peer data sharing systems use either schema-level or data-level mappings to resolve schema as well as data heterogeneity among data sources (peers). Schema-level mappings create structural relationships among different schemas. On the other hand, data-level mappings associate data values in two different sources. These two kinds of mappings are complementary to each other. However, existing peer database systems have been based solely on either one of these mappings. We introduced a model of a peer database management system(PDMS) which uses a mapping that combines schema-level and data-level mappings. We call this new kind of mapping *bi-level mapping*. We presented the syntax and semantics of bi-level mappings. We also provided a query evaluation procedure for the PDMS that uses the bi-level mappings. Moreover, we introduced tabular representation of the bi-level mappings and showed how this representation helps query translation process.

We conducted experiments to quantitatively evaluate the different algorithms presented in this thesis. To do the experiments, we developed a software tool called *Mapping Optimizer and Equivalence Checker* (MOEC). The tool uses real world clock (system clock time obtained using the Java function System.nanoTime) for measuring execution time of various mapping manipulation algorithms. The result obtained from our experiments proves the efficiency of our algorithms and also the utility of schema mapping optimization in the scenario of data exchange.

8.2 Future Work

Based on the work in this thesis, several extensions may be proposed:

- When we convert a schema mapping into a tabular form, we assume that the

mapping is available in the form of GLAV mapping. A more generalized technique can be produced so that it can convert schema mappings of other forms into tabular representations.

- We utilized tabular representation of schema mappings for generating 'minimal' and 'universal' target instances and also for optimizing the mappings themselves. Tabular-representation-based algorithms can be developed for other mapping related operations like *composition of mappings*: given two successive schema mappings, derive a schema mapping between the source schema of the first and the target schema of the second that has the same effect as applying successively the two schema mappings.

- Our optimization technique optimizes the schema mappings by reducing unnecessary *join*s from the mappings. The minimization of tableaux is not a complete algorithm for finding an optimal mapping equivalent to a given mapping. What is needed is a way to go from a tableau to an expression that performs selections and projections as early as possible. The algorithm, presented in [3] for producing an optimal query expression, can be adapted for producing optimal schema mappings in the above sense.

- Methods can be developed to take advantage of functional dependencies in minimization.The problem here is that while we may use dependencies to modify tableaux, the minimal tableau with symbols made equal due to dependencies may not come from an expression over the same set of operands, even when dependencies are applied to the tableau coming from such an expression.

- We conducted the experiments only in the field of data exchange system. Experiments can also be done in the field of data integration and data sharing systems.

Bibliography

[1] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 254–265, 1998.

[2] A. V. Aho, C. Beeri, and J. D. Ulman. The theory of joins in relational databases. *ACM Transaction on Database Systems (TODS)*, 4(3):297–314, 1979.

[3] A. V. Aho, Y. Sagiv, T. G. Szymanski, and J. D. Ulman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. In *Conference on Communication, Control and Computing*, pages 54–63, 1978.

[4] A. V. Aho, Y. Sagiv, and J. D. Ulman. Equivalences among relational expressions. *SIAM Journal of Computing*, 8(2):218–246, 1979.

[5] A. V. Aho and Y. Sagiv J. D. Ulman. Efficient optimization of a class of relational expressions. *ACM Transaction on Database Systems (TODS)*, 4(4):435–454, 1979.

[6] M. Arenas. Xml data exchange: Consistency and query answering. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 13–24, 2005.

[7] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. Miller, and J. Mylopoulos. The hyperion project: From data integration to data coordination. *SIGMOD RECORD*, 32(3):53–58., 2003.

[8] C. Beeri and M. Y. Vardi. A proof procedure for data dependencies. *Journal of the Association for Computing Machinery*, 31(4):718–741, 1984.

[9] D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori, and M. Vincini. Information integration: the momis project demonstration. In *Conference on Very Large Data Bases (VLDB)*, 2000.

[10] P. A. Bernstein. Applying model management to classical meta-data problems. In *Conference on Innovative Data Systems Research (CIDR)*, pages 209–220, 2003.

[11] P. A. Bernstein. Generic model management: A database infrastructure for schema manipulation. In *IDM Workshop*, 2003.

[12] P. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing: A vision. In *International Workshop on the Web and Databases (WebDB)*, 2002.

[13] L. Bertossi and L. Bravo. *Inconsistency tolerance*, chapter Consistent query answers in virtual data integration system, pages 42–81. Springer, 2004.

[14] J. Biskup, U. Dayal, and P. A. Bernstein. Synthesizing independent database schemas. In *ACM SIGMOD*, pages 143–151, 1979.

[15] A. Cali, D. Calvanese, G. D. Giacomo, and M. Lenzerini. Data integration under integrity constraints. *Information Systems*, 29(2):147–163, 2004.

[16] A. Cali and R. Torlone. Checking containment of schema mappings. In *Alberto Mendelzon Workshop on Foundations of Data Management (AMW)*, 2009.

[17] D. Calvanese, G. D. Giacomo, M. Lenzerini, and R. Rosati. Logical foundations of peer-to-peer data integration. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 241–251, 2004.

[18] D. Calvanese, G. D. Giacomo, M. Lenzerini, and M. Y. Vardi. *Lecture Notes on Computer Science*, chapter What is query rewriting, pages 439–457. Springer, 2000.

[19] M. J. Carey, L. M. Haas, P. M. Schwarz, M. Arya, W. F. Cody, R. Fagin, M. Flickner, A. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J. H. Williams, and E. L. Wimmers. Towards heterogeneous multimedia information systems: The garlic approach. In *Workshop on Research Issues in Data Engineering Distributed Object Management (RIDE-DOM)*, pages 124–131, 1995.

[20] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *ACM Symposium on Theory of Computation*, pages 77–90, 1977.

[21] A. Deutsch, A. Nash, and J. Remmel. The chase revisited. In *ACM Symposium on Principles of Database Systems (PODS)*, 2008.

[22] A. Doan and A. Y. Halevy. Efficiently ordering query plans for data integration. In *International Conference on Data Engineering (ICDE)*, pages 393–402, 2002.

[23] O. M. Duschka, M. R. Genesereth, and A. Y. Halevy. Recursive query plans for data integration. *Journal of Logic Programming*, 43(1):49–73, 2000.

[24] R. Fagin, P. G. Kolaitis, , L. Popa, and W. C. Tan. Composing schema mappings: Second-order dependencies to the rescue. *ACM Transaction on Database Systems (TODS)*, 30(4):994–1055, 2005.

[25] R. Fagin, P. G. Kolaitis, and R. J. Miller. *Data Exchange: Semantics and Query Answering*, volume 2572 of *Lecture Notes in Computer Science*. Springer, 2003.

[26] R. Fagin, P. G. Kolaitis, and A. Nash. Towards a theory of schema-mapping optimization. In *ACM Symposium on Principles of Database Systems (PODS)*, 2008.

[27] R. Fagin, P. G. Kolaitis, and L. Popa. Data exchange: Getting to the core. *ACM Transactions on Database Systems (TODS)*, 30(1):174–210, 2005.

[28] M. Friedman, A. Y. Halevy, and T. Millstein. Navigational plans for data integration. In *National Conference on Artificial Intelligence (AAAI)*, pages 67–73. AAAI Press/The MIT Press, 1999.

[29] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Widom. The tsimmis approach to mediation: Data models and languages. *Journal of Intelligent Information Systems*, 8(2):117–132, 1997.

[30] C. H. Goh, S. Bressan, S. E. Madnick, and M. D. Siegel. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Transaction on Information Systems*, 17(3):270–293, 1999.

[31] A. Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 2000.

[32] A. Y. Halevy. Theory of answering queries using views. *SIGMOD RECORD*, 29(4):30, 2000.

[33] A. Y. Halevy, Z. G. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov. The piazza peer data management system. *IEEE Transactions on Knowledge and Data Engineering*, 16:787–798, 2004.

[34] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *International Conference on Data Engineering (ICDE)*, 2003.

[35] R. Heese, S. Herschel, F. Naumann, and A. Roth. Self-extending peer data management. In *Conference on Database Systems in Business, Technology and Web (BTW)*, 2005.

[36] R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *ACM Symposium on Principles of Database Systems (PODS)*, 1997.

[37] D. S. Johnson and A. Klug. Optimizing conjunctive queries that contains untyped variables. *SIAM Journal of Computing*, 12(4):616–640, 1983.

[38] A. Kementsietsidis and M. Arenas. Data sharing through query translation in autonomous sources. In *Conference on Very Large Data Bases (VLDB)*, pages 468–479, 2004.

[39] A. Kementsietsidis, M. Arenas, and R. J. Miller. Mapping data in peer-to-peer systems: Semantics and algorithmic issues. In *ACM International Conference on the Management of Data (ACM SIGMOD)*, pages 325–336, 2003.

[40] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The information manifold. In *Spring Symposium on Information Gathering from Heterogeneous, Distributed Enviroments*, pages 85–91, 1995.

[41] P. G. Kolaitis. Schema mappings, data exchange, and metadata management. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 233–246, 2005.

[42] M. Lenzerini. Data integration: A theoretical perspective. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 233–246, 2002.

[43] J. Madhavan and A. Y. Halevy. Composing mappings among data sources. In *Conference on Very Large Databases (VLDB)*, pages 572–583, 2003.

[44] D. Maier. *The Theory of Relational Databases*. Computer Science Press, 1984.

[45] D. Maier, A. O. Mendelzon, and Y. Sagiv. Testing implications of data dependencies. *ACM Transactions on Database Systems (TODS)*, 4(4):455–469, 1979.

[46] I. Manolescu, D. Florescu, and D. Kossmann. Answering xml queries on heterogeneous data sources. In *Conference on Very Large Data Bases (VLDB)*, pages 241–250, 2001.

[47] R. Meyden. Logical approaches to incomplete information: A survey. *Logics for Databases and Information Systems*, 2:307–356, 1998.

[48] T. Millstein, A. Halevy, and M. Friedman. Query containment for data integration systems. *Journal of Computer and System Sciences*, 66:20–39, 2003.

[49] A. Nash, P. A. Bernstein, and S. Melnik. Composition of mappings given by embedded dependencies. In *ACM Symposium on Principles of Database Systems (PODS)*, pages 172–183, 2005.

[50] R. Pottinger and P. A. Bernstein. Merging models based on given correspondences. In *Conference on Very Large Databases (VLDB)*, pages 826–873, 2003.

[51] R. Pottinger and A. Halevy. Minicon: A scalable algorithm for answering queries using views. *VLDB Journal*, 10:182–198, 2001.

[52] M. A. Rahman, I. Kiringa, and A. E. Saddik. Generalization of an algorithm for checking consistency of mapping tables in a p2p system. In *International Conference on Computer and Information Technology (ICCIT)*, Dhaka, Bangladesh, December 2004.

[53] M. A. Rahman and M. Masud. Foundations and applications of metadata management. *Journal of Electronics & Computer Science*, 12(2):47–56, 2010.

[54] M. A. Rahman, M. Masud, I. Kiringa, and A. E. Saddik. Bi-level mapping: Combining schema and data level heterogeneity in peer data sharing systems. In *Alberto Mendelzon Workshop on Foundations of Data Management (AMW)*, 2009.

[55] M. A. Rahman, M. Masud, I. Kiringa, and A. E. Saddik. A peer data sharing system combining schema and data level mappings. *International Journal of Semantic Computing (IJSC)*, 3(1):105–129, 2009.

[56] M. A. Rahman, M. Masud, I. Kiringa, and A. E. Saddik. Tableaux-based optimization of schema mappings for data integration. *Jounal of Intelligent Information Systems (JIIS)*, 2011. (to appear).

[57] D. J. Rosenkrantz and H. B. Hunt. Processing conjunctive predicates and queries. In *Conference on Very Large Databases (VLDB)*, pages 64–72, 1980.

[58] Y. Sagiv. Quadratic algorithms for minimizing joins in restricted relational expressions. *SIAM Journal of Computing*, 12(2):316–328, 1983.

[59] Y. Sagiv and M. Yannakakis. Equivalence among relational expressions with the union and difference operations. In *Conference on Very Large Data Bases (VLDB)*, pages 535–548, 1978.

[60] L. Serafini, F. Giunchiglia, J. Molopoulos, and P. A. Bernstein. Local relational model: A logocal formalization of database coordination. Technical report, Informatica e Telecomunicazioni, University of Trento, 2003.

[61] N. C. Shu, B. C. Housel, R. W. Taylor, S. P. Ghosh, and V. Y. Lum. Express: A data extraction, processing, amd restructuring system. *ACM Transactions on Database Systems (TODS)*, 2(2):134–174, 1977.

[62] W. Siong, B. C. Ooi, K. Tan, and A. Zhou. Peerdb: A p2p-based system for distributed data sharing. In *International Conference on Data Engineering (ICDE)*, 2003.

[63] J. W. Stroet and R. Engman. Manipulation of expressions in a relational algebra. *Information Systems*, 4(4):195–203, 1979.

[64] J. D. Ullman. *Lecture Notes in Computer Science*, volume 1186, chapter Information integration using logical views, pages 19–40. Springer, 1997.

[65] G. Zhou, R. Hull, R. King, and J. C. Franchitti. Using object matching and materialization to integrate heterogeneous databases. In *Conference on Cooperative Information Systems (CoopIS95)*, pages 4–18, 1995.

[66] M. M. Zloof. Query-by-example: The invocation and definition of tables and forms. In *Conference on Very Large Data Bases (VLDB)*, pages 1–24, 1975.