Integration of Web Data Sources for E-Commerce Transactions

by

Chima Adiele

A thesis
Presented to the University of Manitoba
in partial fulfilment of the
requirements for the degree of
Doctor of Philosophy
in
Computer Science

Winnipeg, Manitoba, Canada, November 2004

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

Integration of Web Data Sources for E-Commerce Transactions

BY

Chima Adiele

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirements of the degree of

DOCTOR OF PHILOSOPHY

Declaration

I hereby declare that I am the sole author of this thesis.

I authorize the University of Manitoba to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the University of Manitoba to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Thesis use Form

The University of Manitoba requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Date	Name	Signature	Address

Dedication

I owe it all to God.

Abstract

Recent advances in information and communication technologies have given impetus to Web-centric applications such as e-commerce. E-commerce involves business activities that deal with the exchange of values for goods and services on the Internet where all transactions are driven electronically. Access to data distributed over the Web is becoming increasingly difficult, as determining semantically equivalent data remains a major problem. Web data integration aggregates data from diverse data sources into a cohesive, coherent, and global e-commerce data set. However, Web data integration is a complex process that requires unambiguous explanations to elucidate the underlying integration concepts. Unfortunately, most existing efforts involve significant manual input and adopt *ad hoc* approaches to data integration, ignoring the theoretical foundations of, and the necessary formalisms to specify, the integration process.

This thesis presents a flexible and correctness preserving approach to Web data integration. This approach leverages algebraic signatures with sound integration principles and algorithms to provide a theoretical foundation that facilitates scalable Web data integration. To identify a suitable integration mechanism, a characterization of e-commerce data sets is presented. We provide an algebraic data model that reduces the topological structure of Web data sources to regular expressions and define algebraic operators and functions to manipulate objects in the algebraic model. A common-term vocabulary is used to semantically describe objects from local data sources to eliminate multiple views of objects. We present a high-level architecture of a Web integration system for e-commerce (WISE) and formally specify the basic components of WISE. Formally specifying an integration system for e-commerce transactions provides a clear understanding of the system and also reveals ambiguities, incompleteness, and contradictions in the informal definition, and thus, permits the correctness verification of the integration components. We provide a graph-based integration algorithm that dynamically identifies and analyses correspondence assertions to integrate local data sources. Finally, we show that our integration result is complete, correct, minimal, and understandable.

Acknowledgement

I am grateful to my supervisor Dr. Sylvanus A. Ehikioya for his support and useful advice. His insistence on hard work, though challenging, has been very rewarding. I would like to recognize the efforts of my thesis committee for their constructive criticisms to improve the focus of this thesis. I am particularly grateful to Dr. John Bate (Head, Department of Computer Science) for his immense contributions to the successful completion of my program. I would like to thank Dr. Peter C. J. Graham for his insight, time and patience in guiding me through the corrections of the final document. I appreciate the financial support provided by the University of Manitoba, Dr. Sylvanus A. Ehikioya, Dr. Peter C. J. Graham, and Dr. Helen Cameron. A lot of faculty members encouraged me in many ways. Their support is appreciated. Finally, I want to thank my family for their continued support, especially my lovely wife.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter is organized in the following way. First, we define e-commerce, state the research problem and give the motivation for integrating Web data sources for e-commerce transactions. The definition of e-commerce and motivation will put our discussion in context, and hence, pave the way for us to state the objectives of the thesis; list the significance of this research and its contributions; and also describe the limitations of the research. Next, we give a detailed description of the problem domain to simplify our exposition, thus providing a benchmark to measure successful integration systems. Finally, we provide an overview of the organization of the thesis.

The Internet has grown rapidly due mainly to the advent of the World Wide Web (Web) and increased number of Web-centric applications, such as e-commerce. E-commerce is any form of trade that is supported by the Internet where all interactions between the participating parties are carried out electronically. E-commerce creates a huge economic potential that is feasible because of the seamless global access that the Internet provides, the open environment the Internet supports, and the user-friendly graphical interfaces that Web tools support.

The Web has become a very popular tool for publishing and disseminating information due to its high level of connectivity, ease of use, and relatively low cost of access. Consequently, a large amount of data now resides on the Web. E-commerce data on the Web are represented in different formats that range from data in rigid-schema

structured databases[1], to entirely unstructured free text documents and the growing volume of semistructured data (largely instantiated as eXtensible Markup Language (XML) [W3C98] documents). E-commerce data represents information about e-commerce activities or entities in the domain of e-commerce.

Running Example

We now introduce one of the examples used in this thesis (see Figure 1.1). We consider two different data sources. The first source is an XML database, Airline-A (S_1) containing information about baggage handling devices of Airline-A. The second source is a relational database, Airline-B (S_1) containing information about baggage handling devices of Airline-B. The two airlines have a service agreement with respect to baggage handling in their operational bases. In Figure 1.1, the BH-Devices schema represents the XML database of Airline-A, while the BHD relation represents the relational database of Airline-B. There is a need for the two databases to communicate to achieve contractual agreements.

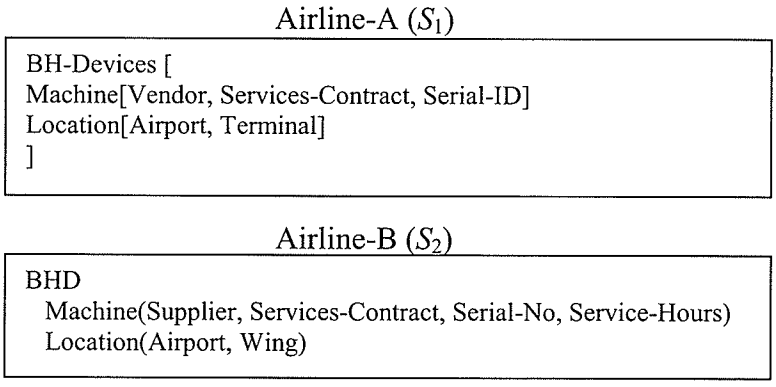

Figure 1.1: Example with Two Source Schemas

Observe the conflicts that exist (both structural and naming) between the two databases. One of the main challenges for e-commerce infrastructure designers is information sharing among business partners. Information sharing involves retrieving

[1] A database schema describes the organizational structure of the database.

data located in different sources thereby obtaining a global view to overcome conflicts. Data integration provides a global view of data across data sources.

1.1 Statement of the Problem

There are several open questions with respect to integrating Web data sources for e-commerce transactions. These questions include:

1. What are the data management needs of an e-commerce application?
2. What are the characteristics of e-commerce data sets?
3. How can we semantically describe e-commerce data sets both in legacy systems[2] and on the Web?
4. Can correspondence assertions[3] between objects in different data sources be identified dynamically?
5. Can correspondence assertions between objects in different data sources be analysed dynamically?
6. Can we generate integration results that are mathematically sound?

To answer these questions, we resort to the use of algebraic signatures[4] and sound integration principles. An efficient integration mechanism requires the use of a mathematical framework to provide an unambiguous explanation of the underlying concepts, thus reducing the complexity of the entire integration problem.

In summary, the problem addressed in this thesis is to integrate data from Web data sources so that they can be useful for e-commerce transactions. Web data emanate from data sources that are inherently distributed and potentially heterogeneous. On one hand, enterprises represent their data using disparate data sources. On the other hand, users want to access data from different data sources without understanding the intricacies of the underlying data sources. The integration problem deals with a situation where data from two or more heterogeneous data sources are integrated. The aim of an integration

---

[2] A legacy system is any information system that significantly resists modification and evolution to meet new and constantly changing business requirements [BS95].

[3] A correspondence assertion is a statement that establishes the relationship between the semantics of objects in two different schemas.

[4] An algebraic signature is an implicit, property-based formal framework that implicitly expresses operations by their algebraic equivalences.

system for e-commerce is to make data available and easily accessible among all participants. We need to integrate Web data sources to free the user from the responsibility of locating and interpreting e-commerce data from the broad spectrum of information that exist on the Web. An environment that is able to provide a global view of e-commerce data is thus highly desirable. Such a dedicated environment brings together all the participants in an e-commerce transaction and hence creates more opportunities to exchange information. An e-commerce integration system provides a platform where e-commerce data from different sources can be accessed using a global schema, thus taking away the burden of locating and interpreting e-commerce data from the user. The global schema provides a virtual database that enables the user to query multiple data sources that are inherently distributed without knowing the complexity of the underlying data sources.

1.2 Motivation

The Internet offers a virtual marketplace where buyers and sellers exchange values for goods and services. E-commerce, by nature, is a "self-service model" where users are saddled with the responsibility of identifying and interpreting Web data. In particular, the user is faced with the following challenges [Maa03]:

1. The user has to identify the relevant Web sites to access catalogs of available items;
2. It is necessary for the user to understand how to navigate the Web sites; and
3. The user has to specify his / her needs according to the characteristics or terminology of the Web sites.

Despite the potential the Internet provides, e-commerce activities are often difficult to exploit. Internet access alone lacks the requisite resources to facilitate easy access to the large volume of disparate e-commerce data sets on the Web. Access to data distributed over the Web is becoming increasingly difficult, as identifying semantically equivalent data remains a problem (i.e. How does a consumer know that a supplier's product is what s(he) is looking for?). The heterogeneity of e-commerce data sets makes it difficult to locate data relevant to a query or relate information from different sources. Web users spend a substantial amount of time searching the Web for e-commerce data

and sites where they can locate such data. Commercial search engines, such as Google, Lycos, Excite[5], etc., are inadequate because such systems are restricted to simple keyword techniques. For example, the finest deal in terms of goods and services may be available, but difficult to locate because it is effectively hidden by other irrelevant data. Indeed, e-commerce data should be easy to access to be useful.

A system that provides easy access to Web data sources is necessary. Web data integration presents a common data format and aggregates data from diverse data sources into a cohesive, coherent, and global e-commerce data set, thus making Web data readily available to the numerous participants in e-commerce activities. Unfortunately, most existing efforts involve significant manual input, and adopt *ad hoc* approaches to data integration, ignoring the theoretical foundations of, and the necessary formalisms to specify, the integration process. Sheth and Gala [SG89] argue that the task of identifying attribute[6] semantics for all pairs of attributes between two schemas cannot be automated, and hence requires heuristics to identify a small number of attribute pairs that may be potentially related. Rahm and Bernstein [RB01] observe that existing approaches used in mapping elements of two different schemas are based on heuristics that cannot be formalized and mapping results that are mathematically unsatisfying. We argue very strongly that such mathematically unsatisfying results are unacceptable in the e-commerce domain where correctness of operations and accuracy of results are essential.

1.3 Objectives of the Thesis Research

The primary focus of this research is to provide a data management solution to e-commerce applications. In particular, we leverage data integration concepts to present a flexible and correctness preserving approach to data management for e-commerce applications (an integration approach devoid of the traditional *ad hoc* integration practices that produce mathematically unsound results). To achieve the envisioned objectives, we formally specify components of the integration system to reduce the system's complexity, enhance the understandability of the underlying integration

[5] These common commercial search engines are available at: Google, http://www.google.ca/; Lycos, http://www.lycos.ca/; and Excite, http://www.excite.com/

[6] The property that describes some aspects of the object that we wish to record is called an attribute.

concepts and facilitate the correctness of integration results. Accordingly, this research strives to achieve the following objectives.

- Identify the data management needs of an e-commerce application, and hence provide a suitable integration mechanism.
- Semantically describe objects in local data sources so that multiple views of objects are eliminated.
- Dynamically identify and analyse correspondence assertions to remove the burden of interpreting data from the user.
- Generate an integration result that is mathematically sound and hence be able to show that our integration approach is correct, complete, minimal, and understandable.

We present a data-centric e-commerce application model to bring to focus the data management needs of an e-commerce application. This thesis provides a formal characterization of e-commerce data sets to identify the specific integration needs of e-commerce systems and hence determine a suitable data model and efficient integration algorithm. To semantically describe objects in local data sources, we define an algebraic data model and specify a common-term vocabulary (*CTV*). The algebraic data model provides a hierarchical description of the objects it represents and the *CTV* makes explicit the content of data in the local data sources so that multiple views of objects are eliminated. We also design a graph-based integration algorithm that leverages the semantic description of objects to dynamically identify and analyze correspondence assertions. Finally, we present a high-level integration architecture (WISE), which adopts a two-phase integration strategy that isolates the input process from the output process. Our integration mechanism: 1) guarantees autonomy of data sources because it does not prescribe structures for the local data sources; 2) scales over multiple data sources since the task of identifying and analyzing correspondence assertions, is carried out dynamically without complex user intervention; and 3) guarantees correctness of operations and produces integration results that are mathematically sound because integration components, including correspondence assertions are formally specified. We

also provide a sound theoretical foundation to show the correctness of our integration approach.

1.4 Significance of the Research

This research provides a theoretical foundation to a problem that would ordinarily be classified as an industrial integration problem. Until now, most industrial integration systems [ABC+95, BCE+93, PKGV95, SK93] lacked formal frameworks and depended on significant manual input in their designs. Most research-based integration systems such as Pegasus [ASD+91], TSIMMIS [PGW95] and DISCO [TRV95] rely on human intervention in reconciling conflicts. These research systems also lack the flexibility and the theoretical base to manage data in an e-commerce environment since most of them depend on assertions that are hardly formalised. Therefore, the main significance of this research is the use of a formal framework and theoretical principles to produce integration results that are mathematically satisfying. Mathematically unsatisfying results are unacceptable in the e-commerce domain where correctness of operations is fundamental to e-commerce transactions. This thesis adopts an integration approach that goes beyond representing and reasoning about mapping to dynamically generating mappings of attribute semantics between objects in different data sources. Thus, our approach removes the burden of identifying correspondence assertions from the users.

Contributions

This thesis contributes in the following ways.

1. It provides a characterization of e-commerce data sets with a view to identifying the integration needs of such data. Characterizing e-commerce data sets reveals the nature of such data and hence, facilitates the choice of an appropriate data model. Characterizing e-commerce data sets also enhances the design of a suitable integration mechanism.
2. It provides a data-centric e-commerce application model that brings to focus the data management needs of an e-commerce application. This model shows how the

basic building blocks (data, methods, and activities) of an e-commerce application interact, and hence exposes the challenges in managing data in the model.

3. It provides an algebraic data model capable of representing e-commerce data sets to accommodate both data resident in legacy systems and on the Web. Sheth and Larson [SL90] observe that the choice of a particular data model as an integration platform influences the kind of data sources to be integrated by an integration system. The range of data sources integrated by a given integration system is one of the parameters to measure the success of the integration system. Our model integrates data from disparate data sources.
4. It provides a formal specification of a common-term vocabulary. A common-term vocabulary elucidates the content and meaning of e-commerce data to support the information needs of an organization. The use of a common-term vocabulary enhances data integration, as the meaning of the data to be integrated is made explicit.
5. It provides dynamic identification and analysis of correspondence assertions, and hence removes the burden of identifying correspondence assertions from the users. The use of manual efforts to identify correspondence assertions is complex and error prone.
6. It provides an efficient graph-based integration algorithm. Our algorithm leverages the algebraic data model and the common-term vocabulary to identify more attribute semantics than existing systems, such as [Che00, SP94]. This thesis identifies and formalizes seven correspondence assertions.
7. It provides a correctness preserving integration system, which shows that the integration result is correct, complete, minimal, and understandable. The use of algebraic signatures to specify components of the integration system eliminates the traditional *ad hoc* approach to integration. We combine formalism with a detailed description of the integration system. Furthermore, this research provides a sound theoretical foundation to prove the correctness, completeness, and minimality of our approach.

1.5 Limitations

This research is limited to the use of theoretical methods and formal techniques to provide an abstract integration model for e-commerce data on the Web. We use algebraic signatures to formally specify components of the integration system, and hence show the completeness and correctness of our approach. We also provide proofs and analyses of our integration algorithms. The implementation and deployment of the abstract integration model is outside the scope of this thesis.

1.6 Problem Domain

To have a clearer idea of what data integration entails, we illustrate with a simple example. Figure 1.2 shows a typical e-commerce integration environment involving different participants in a business-to-business (B2B) e-commerce system. Some of the applications involved in these business transactions run on legacy systems, while others are Internet-based systems.

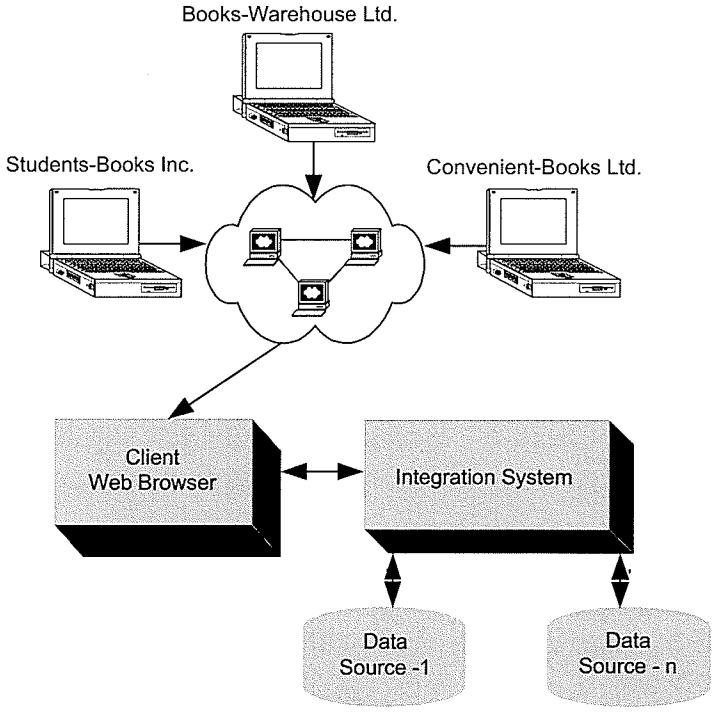

Figure 1.2: E-Commerce Integration Environment

Books-Warehouse Limited is a popular online bookshop that depends on Convenient-Books Limited, a publishing house, for its books supplies. Convenient-Books Limited orders raw materials for its operation from Students-Books Incorporated, which depends on Books-Warehouse Limited for its stationeries. Notice that the three companies are "close neighbours" that are directly or indirectly involved in business transactions. It is expected that these neighbours that have common business interests should naturally exchange information with minimum difficulty. Usually each business partner has its own message format (schema and data format), which may differ in syntax or structures from the others.

Unfortunately, these companies have different means (e.g., ordering formats, payment infrastructure) of exchanging messages that describe business transactions with one another. For example, Books-Warehouse Limited uses an XML database, an Oracle database drives Convenient-Books Limited and Students-Books Incorporated stores its data in a file system. These different systems are expected to communicate with one another to facilitate information exchange. As a result of the different backend systems, the ordering forms used by Books-Warehouse Limited may be different from the ordering form that is used by Convenient-Books Limited. It is not unlikely that these independent autonomous systems have different schemas, which further exacerbates the problem of information exchange. Also, the payment infrastructure for Students-Books Incorporated may be different from the payment infrastructure for Convenient-Books Limited. It would be difficult and costly for these companies to adopt a common system and discard their existing systems, which represent great assets both in terms of financial expense and the amount of time and effort spent in creating such data repositories. Schema integration offers a mechanism for these companies to communicate with one another without discarding their existing systems.

1.6.1 Schema Integration Processes

Schema integration systems can be characterized orthogonally in the directions of input, process, and output. Figure 1.3 shows processes of schema integration designed to show the desired status of the input process (input), integration process (process), and output

process (output) of an integration system. Ideally, an integration system is designed to have an input process that encourages autonomy of data sources, an automated integration process, and an integrated schema that ensures users transparency[7]. The idea behind this characterization is the fact that a "suitable input" applied to an "appropriate process" will always generate the "desired result" as output. A desirable schema integration system should provide easy access and transparency to e-commerce related data on the Web, while retaining the autonomy of the data sources.

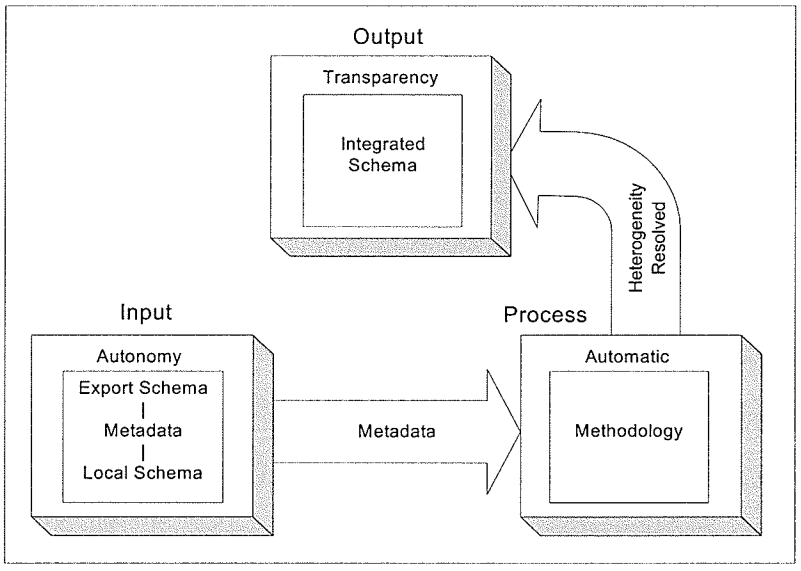

Figure 1.3: The Processes of Schema Integration

This system should have: 1) a flexible data model to accommodate e-commerce data represented in different formats; 2) a formal specification of the components of the integration process to guarantee correctness; and 3) a reasonable level of automation to remove the burden of identifying semantic relationships from the users. We discuss Figure 1.3 with respect to source autonomy, automation of the integration process and transparency of the integration result.

Autonomy. The input to an integration system includes data represented in different formats, and meta-information about the individual data sources. The design goal of an

[7] Transparency in this context is both logical and physical. Logical transparency stipulates that the user does not need to know the format and structure of the underlying data sources. Physical transparency stipulates that the user does not need to know the physical location of the data sources.

integration system is to have autonomous data sources as input to the system regardless of the format they are represented in. Autonomy of data sources implies that each data source can make choices with respect to its data model, naming concepts, universe of discourse, the structure and content of metadata, etc., regardless of the design choices of the other data sources. The independent choices of the different data sources naturally lead to heterogeneity. For example, different designers are likely to design their local schemas using different data models, use the same name to represent different objects, or may consider an entirely different universe of discourse. The autonomy of data sources gives database designers the freedom to design their databases to suit local needs and provide support for legacy systems while still participating in the integration system.

Automation. There are different methodologies for schema integration [SL90], including heuristic algorithms, schema transformations, logical rules, artificial intelligence techniques, and other complex methods that may be a combination of two or more of the methods listed above. However, the ultimate goal is to have an automated integration methodology devoid of complex user intervention. A desirable schema integration system should be able to automatically resolve heterogeneity among data sources, including the identification of assertions, determination of attribute relationships, and the generation of correspondence rules. An automatic integration process that calls for minimal user interaction creates transparency. To facilitate automation therefore, the integration process must: 1) receive adequate meta-information to specify the semantics of the export schema, 2) clearly understand the different contexts to eliminate multiple views of interpretation of a real-world state, and 3) provide suitable integration algorithms that resolve conflicts without user intervention.

Transparency. The output of an integration system ranges from integration tools that aid integrators to achieve integration to logical rules that simply show interschema relationships and global views that are transparent [Law01]. Ideally, the result from an integration system should be an integrated schema that is transparent, at least from the users' point of view. The user should see the global view as a common and reliable interface local to a particular user. Basically, a transparent integrated schema should insulate users and applications from the complexity of the system. Transparency is a by-product of a "suitable input" applied to an "appropriate process" and also a reflection of

the resolution of heterogeneity. According to Busse *et al.* [BKLW99], the success of a schema integration system can be measured by: 1) the data model used for the integration platform; 2) the level of involvement of the user in the integration process (transparency to the end-user); and 3) the level of heterogeneity the system is able to resolve and the integration methodology adopted.

1.6.2 Thesis Approach

To address the issues listed in Section 1.6.1, we leverage algebraic signatures to specify systems' functionalities and then use sound integration principles to generate a global integrated schema of the participating data sources. Our integration approach makes explicit the content of data, so that an efficient integration algorithm can dynamically identify and analyze assertions. An algebraic signature is an implicit, property-based formal framework that implicitly expresses operations by their algebraic equivalences. Our interest is on *what* system's operations are required to accomplish a task, rather than *how* the task is to be carried out. To achieve the envisioned objective, we use common set notations[8] ($\cap$, $\cup$, $\supseteq$, $\subseteq$, $\notin$, $\in$, $\mathbb{N}$, $\mathbb{P}$) to describe structural components, and predicate logic to describe pre- and post-conditions for any functional requirement. We give these conditions (pre- and post-conditions) as predicates. A simple predicate, which usually has one or more arguments, is of the form P(x), where x is an argument that is used in the predicate P. In general, a quantified statement can be written in one of two forms:

1) <quantifier><declaration(s)> • <predicate>
2) <quantifier><declaration(s)>|<constraint> • <predicate>

The universal ($\forall$) and existential ($\exists$) quantifiers are commonly used. Every declaration must satisfy a given constraint. The symbols "|" and "•", which are part of the syntax, mean "satisfying" and "such that", respectively. To create compound predicates, statements can be nested and combined together using one or more logical connectives, such as: *and* ($\wedge$), *or* ($\vee$), *not* ($\neg$), *conditional* ($\Rightarrow$), and *biconditional* ($\Leftrightarrow$). Other symbols used in this thesis are explained in Appendix A. Truth tables for the logical connectives

[8] Algebraic signatures use symbols from set notations and predicate logic to provide specifications.

are available in most discrete mathematics textbooks. The formal specification of a requirement in this thesis follows the general format of a quantified statement.

The framework in this thesis combines formalism and theoretical principles with efficient integration algorithms to achieve integration. The use of theoretical methods provides sound mathematical statements to model the properties of an integration system. In particular, using algebraic signatures for specifying components of the integration system offers the following benefits [Ehi97, PKT92, Win90, Win98].

- Algebraic signatures provide a formal foundation for establishing the correctness of the integration model. A formal foundation offers the framework the ability to prove that the specification of a system is realizable and prove properties of the system without necessarily executing such properties to determine the system's behaviour [Win90].
- Algebraic signatures abstract away irrelevant portions of informal definitions, and emphasize a system's functionalities that precisely specify the behaviour of the model. The use of abstraction facilitates the identification of key properties being modeled while ignoring unimportant details to manage the complexity and promote correctness, reusability, and understandability of the integration system.
- Algebraic signatures support the design of an independent and precise description of the behaviour and effects of correspondence assertions in a given integration system.
- Formally specifying an integration model for e-commerce transactions provides a clear understanding of the model, reveals ambiguities, incompleteness, and contradictions in the informal definition, and thus permits the correctness verification of the integration components.
- A formally verified system can be used with greater confidence. Verification provides additional assurance that a system conforms to its specification [Win98].
- Formally specifying components of an integration system reduces the system's complexity, eliminates redundancies, and elucidates the overall integration process, thus, creating a framework for automatic schema integration.
- The use of theoretical methods and formal techniques to prove correctness of the integration algorithm guarantees integration results that are mathematically sound.

Correctness and completeness are important features for e-commerce systems that may involve huge financial transactions.

- Providing a detailed description of the integration system and formally analysing the building blocks of e-commerce applications gives better understanding of the systems' components. The insights provided by such verbose description could assist systems' developers make appropriate choices in developing an integration system for e-commerce transactions.

1.7 Organization of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 gives background information on e-commerce and integration of Web data sources, thus laying the basic foundation that sets this work in perspective. We contend, however, that existing solutions are inadequate, and suggest that a viable integration system must be flexible, scale over multiple data sources, insulate users from the intricacies of the underlying data sources, and correctly preserve integration results. Chapter 3 describes the characteristics of e-commerce data sets, and provides a data-centric e-commerce application model that exposes the data management needs of an e-commerce application. Characterizing e-commerce data sets aids the identification of an appropriate representation technique, enhances the creation of a common-term vocabulary, and facilitates the choice of a suitable integration mechanism. In Chapter 4, we discuss the semantic description of data sources to make explicit their data content. We present an algebraic data model that reduces the topological structure of Web data sources to regular expressions. We then define algebraic operators and functions to manipulate objects in the algebraic model. Central to the input process is a filter mechanism, which recognizes a "context schema" that eliminates multiple views of objects from all data sources. We show that the process of recognizing a context schema is decidable. Chapter 5 examines the identification and analysis of correspondence assertions. Here, we formally specify correspondence assertions, and provide a fine-grained comparison of objects at different levels of perception to identify more attribute semantics. Chapter 6 examines our high-level integration architecture, WISE. We specify the basic components of WISE, and show that

the integration result is correct, complete, minimal, and understandable. Finally, Chapter 7 concludes the thesis and provides insight into future work.

Chapter 2

Related Work

This chapter consists of three major sections. Section 2.1 provides an overview of e-commerce systems including e-commerce application environments and describes the characteristics of e-commerce systems, infrastructure for e-commerce and the challenges of implementing e-commerce transactions. Section 2.2 discusses relevant data management techniques to solve the challenges behind e-commerce transactions. In particular, we present a background survey of data integration approaches with a view to identifying integration challenges. Section 2.3 examines the tools needed for data integration in an e-commerce environment. These include: 1) a common ontology that gives a clear semantic understanding of objects; 2) a semistructured data model that provides a structure to represent e-commerce data; and 3) algebraic signatures that provide a mathematical framework to formally specify components of the integration system and prove the correctness of our approach. These parts are necessary to understand how to provide reliable data management techniques for e-commerce.

2.1 Electronic Commerce

E-commerce involves business activities that deal with the exchange of money for goods and services on the Internet where all transactions are driven electronically. These business activities consist of different functions including establishing both the means and incentive to sell, collaborating with participants, and managing data emanating from such activities. All these functions together offer great opportunities for organizations to

leverage their customer base and maximize profit [Raj00]. An effective online customer relationship contributes to the lifetime value of customers by reducing acquisition costs, increasing sales, extending the customer life cycle, and strengthening customer retention [AE04a].

E-commerce activities are all-embracing and cut across different fields including the Internet, which provides the global networks between intercommunicating computers; databases used for storing data and information that is transferred over the Internet; operating systems, which provide enabling and secured framework for intercommunicating units to interact; cryptography for securing information that is transferred from both intentional and unintentional abuses; law for providing a legal framework to protect participants from abuses; distributed computing for connecting the different databases over a network; networks for providing the communication links between computers; and social behaviour analysis to study the behaviour of customers and market dynamics with a view to providing services that are tailored towards individual customer's needs.

E-commerce systems fit into various categories. The three prominent categories are: 1) consumer-to-consumer (C2C); 2) business-to-customer (B2C); and 3) business-to-business (B2B) e-commerce models.

1. Customer-to-Customer (C2C) e-commerce involves a situation where customers can sell to other customers using an online market infrastructure. This may be some form of an auction site like eBay[9]. A typical example within the University of Manitoba environment is the sale of used goods that is posted on the Web site of the International Center for Students. Students post their goods to be sold on the Web site and receive responses from interested buyers via email or phone calls. The seller and buyer agree on how the goods will be delivered and paid for. Although this system falls short of a proper e-commerce transaction because there is no standard payment infrastructure (method), it however, shows how the Internet is able to facilitate C2C commercial activities.

[9] Available at: http://www.ebay.com/

The huge success of eBay has proven the demand for C2C e-commerce. eBay operates from a centralized infrastructure with all its scalability problems (network, bandwidth, server load, availability, etc.). However, C2C can also be mapped naturally to the emerging field of peer-to-peer (P2P) [APHS02] systems by following the underlying interaction model of customers, i.e. peers. Thus, a P2P infrastructure for C2C would be a decentralized system similar in function to eBay but without eBay's dedicated, centralized infrastructure.

2. Business-to-Customer (B2C) e-commerce involves business transactions between an organization and public domain customers online over the Internet. In this form of e-commerce, an organization creates a website to sell goods and services to public domain customers online. Such businesses may not have a physical store for walk-in customers. For example, Amazon.com, a well-known virtual bookseller, is an example of B2C e-commerce. The company sells all their books via the Internet, and coordinates deliveries directly with the publishers so they do not have to maintain any inventory. B2C also includes transactions that offer after-sales support and customer service, as well as facilitating communication between business partners.
3. Business-to-Business (B2B) e-commerce involves business transactions between corporate organizations over the Internet. B2B e-commerce provides an organization with the means to source for the best deal that matches the time and cost requirements of the organization. For example, a manufacturer can leverage B2B e-commerce to determine which supplier can provide the best deal and delivery time or source from an alternative supplier. The information flow occurs within a few hours instead of weeks when communicating by ordinary post or fax. If there is a corporate procurement that involves organizations and government, then we have Business-to-Government e-commerce (B2G). For purposes of simplicity, we assume B2G to be part of B2B.

Benefits of E-Commerce

There is increasing interest in e-commerce systems because of the benefits participants derive from such systems. Basically, e-commerce brings comfort and convenience to the customer at reduced cost, while enabling businesses to maximize profit. For example, Cisco Systems saved more than half a billion dollars and reduced error rates from 25 percent to two percent when the company changed from its traditional phone and fax ordering process to an online ordering process [Man00]. Internet-based systems eliminate delays created by traditional business communication methods, such as mail, email, fax and voice mail, and improve upon traditional data capture mechanisms that often require re-entering of data into multiple systems. E-commerce enables a marketplace that has no limitation in terms of time, space and geographical location. For example, customers from any geographical location could access an online mall twenty-four hours a day, seven days a week (twenty four-seven). Keeping a shop open twenty four-seven can be very expensive and has its attendant risks. Even when an enterprise can afford to be open twenty four-seven, it may not be safe and convenient for customers to shop late at night. An online mall provides customers the opportunity to shop in the comfort and convenience of their homes, especially those customers who are not very mobile. E-commerce extends existing distribution channels and may even introduce new distribution possibilities. For example, the use of the Internet to sell and buy music is an innovation enabled by e-commerce to create and facilitate a new industry that supports musicians uploading music to the Internet where customers can download and play such music for a fee.

2.1.1 E-Commerce and E-Business

More often than not, the terms e-business and e-commerce are either used interchangeably or are confused with each other. The difference between e-commerce and e-business depends on the line of argument of the author. Carbone and Stoddard [CS01] define e-business as the digital transaction of business operations, which involves automation, integration and organization of business processes, while e-commerce is the electronic exchange of money for goods and services. This line of argument pre-supposes

that e-commerce is a subset (part) of e-business. On the contrary, Laudon and Traver [LT02] argue that e-business involves using the Internet to enable transactions and processes of a firm internally, which does not involve exchange of values with the outside world (customers, other businesses, government). This line of argument corresponds to what Rajput [Raj00] and Shim *et al.* [SQS+00] called intra-business commerce in their classification of e-commerce systems. A strict adherence to the definitions given in [Raj00, SQS+00] implies that e-business could be seen as a subset of e-commerce. For the purpose of this research, we align our argument with that of [LT02] with some modifications. We accept the definition of e-commerce as buying and selling of goods and services on the Internet, while e-business is the enhancement of business processes within a firm. These definitions are depicted in Figure 2.1.

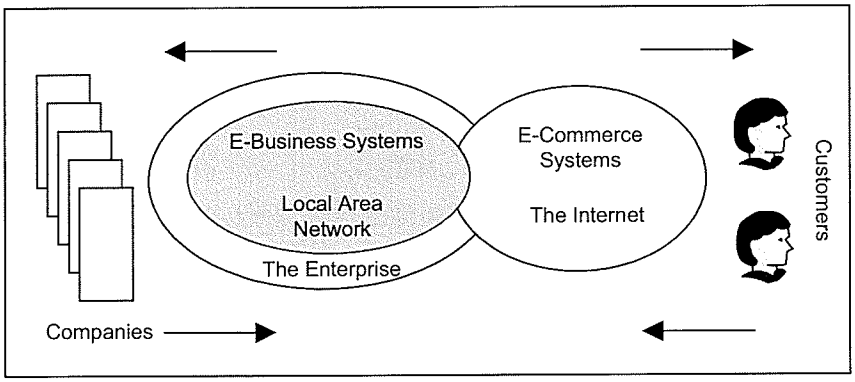

Figure 2.1: Conceptual Difference between E-Commerce and E-Business

Table 2.1: E-Commerce and E-Business Relationship Table

E-Commerce	E-Business
Commercial activities that involve exchange of values with the outside world.	Involves improvement and sharing of business strategies within the firm.
The aim of e-commerce is to improve the customer base of the firm, and hence maximize profit.	Aims to improve business strategies within the firm in order to withstand competition.
E-commerce infrastructure supports customers and other businesses.	E-business infrastructure supports the company's internal business systems and can also support e-commerce exchanges.

Here, the e-business module is within the firm's infrastructure and consists of the firm's internal business systems. The e-commerce module cuts across the firm's boundaries. Table 2.1 gives a summary of the differences and similarities between e-commerce and e-business.

2.1.2 E-Commerce Application Environments

E-commerce provides a Web-enabled application that supports business transactions between multiple parties through the Internet. E-commerce involves managing enormous volume of data (e.g., product catalogues) and, therefore, requires database support for its transactions. The Internet hosts e-commerce through one of its services called the World Wide Web (Web). The Web is a global hypertext system, providing access to documents through a hypertext transfer protocol (HTTP) interface [W3C99a]. Navigation is achieved by clicking the hyperlinks. Figure 2.2 depicts a typical Web-based e-commerce application environment using a multi-tier architecture. Data sources (database servers) form the lowest tier, while the highest tier consists of the client machines that provide the browser-based user interfaces. Collections of applications are in the middle-tier between the client and the database server(s). In an e-commerce environment, Web services are often deployed in application servers (that provide pre-packaged e-commerce solutions) that use the database server to provide services to the client. An e-commerce solution provides Web-based e-commerce software to facilitate online business transactions.

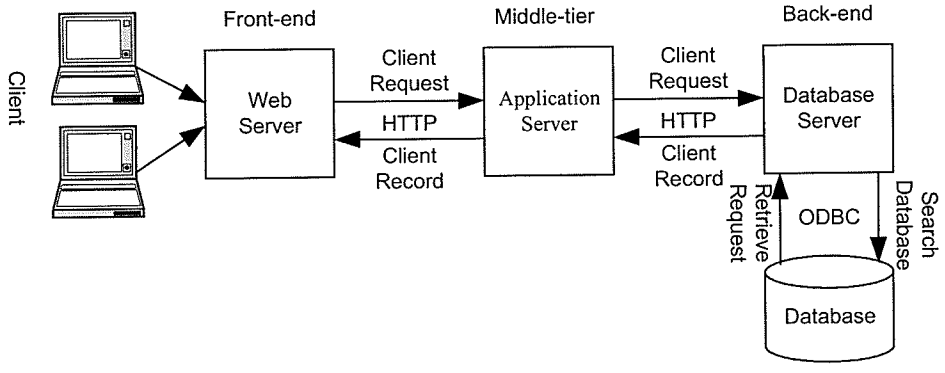

Figure 2.2: A Typical E-Commerce Application Environment

For example, in Figure 2.2, customers' requests are sent from the Web server to the application server. The application server does most of the work and it forwards the request to database server, and returns the result to the customer. E-commerce solutions manipulate such requests to provide services to the customer. It could be a request to locate a particular product, which triggers the database server to search the database for selected records and return them to the client. In an e-commerce environment, database servers must be able to serve many clients simultaneously, as some Web sites may receive thousands of hits per minute.

A Web server is a program that offers a service that can be reached over the Internet using HTTP. Web servers are common access gateways that enable users with browsers to access remote information content. Web servers interact with browsers using the HTTP protocol and also incorporate the necessary technologies to interface with an enterprise's backend services and applications. An application server is the middle-tier in the three-tier e-commerce architecture [Raj00]. It provides an interface between the Web server and the database server. An application server permits clients to use applications and databases that are managed by the server.

A database is an organized collection of logically related data, usually stored in an electronic format that can be searched by a computer. A database management system (DBMS) is the software that manages and controls access to the database. The DBMS enables users to access and maintain the database, as it provides an interface between user programs and the database. Databases support the expression of complex queries in a concise and simple fashion. Open database connectivity (ODBC) [Mic94] software enables different databases to interoperate. An application developer does not need to write a separate program to access different databases (e.g., Access, Foxpro, Oracle, etc.); instead a developer only needs to use the ODBC language (a combination of ODBC API function calls and the SQL language) to access any ODBC compliant database. The ODBC manager has a means of interacting with the target database regardless of its composition. Java Database Connectivity (JDBC) [Sun98], standardized a similar database interface for Java.

2.1.3 Characteristics of E-Commerce Systems

The attributes of e-commerce systems that have direct relationship with schema integration include:

- Support for multimedia content. E-commerce systems, unlike traditional applications, typically handle richer data formats and different multimedia artefacts, such as animated graphics, video, audio, still images, etc. Content management is a basic necessity for any reasonable e-commerce application because it offers merchants flexibility in controlling the information that users access. Good content management also influences the presentation of data to the user. Özsu and Valduriez [OV99] note that the popular relational database model lacks the needed support for multimedia objects, necessitating better ways of handling multimedia artefacts to provide rich content for e-commerce applications.
- Support for heterogeneous data sources. The global and distributed nature of e-commerce systems makes it necessary to access data across organizational boundaries. Such data emanate from independent data sources that are potentially heterogeneous and autonomous. Data heterogeneity arises because data sources are developed independently, and different designers are most likely to have different information needs. E-commerce solutions should, therefore, provide access to data from disparate data sources. Unfortunately, providing access to information from data sources that are autonomous and potentially heterogeneous has been a major challenge to the database research community [BCV99, GBMS99, GPQ+97, Lev01].
- Integration of different application components. E-commerce systems are complex in nature and comprise a combination of different components, which may be legacy, Internet-based, and other sub-systems. Many enterprises have substantial investments in their corporate databases that represent great assets both in terms of financial value and the amount of time and effort spent in developing such data repositories. The value of this data, mostly in legacy systems, is beyond measure. The growth of e-commerce has its attendant challenges on how to manage the transition of new technologies, while retaining the value of their investments in

legacy data. An enterprise can continue to leverage existing systems instead of porting one system to another. To be successful, therefore, Internet-based applications, including e-commerce solutions, must interoperate with legacy and other sub-systems.

- Support for assorted information appliances. Internet connectivity, supported by telecommunication networks, permits access to the Web through a range of methods, including portable devices. E-commerce systems, therefore, should support access to data from various information appliances including desktop computers, notebook computers, personal digital assistants (PDAs), and other devices. Assorted information appliances empower users to access e-commerce data from mobile or remote locations. E-commerce data must, therefore, be represented to ensure ease of display in any of the appliances. For example, PDA users now access their stock trading systems regardless of their locations and the trading software must support the limited screen "real-estate" provided by typical PDAs.
- Support for quick response to users' queries. E-commerce systems should provide quick response to users' queries. Therefore, data must be organized to enhance quick access. For example, the relational data model lacks the flexibility to handle vendor catalogues containing large volumes of stocks with different attributes because relational databases are inefficient in managing many tables with many nulls. To enable e-commerce solutions to respond to users' queries rapidly, a flexible data model that can manage large catalogues efficiently must be used to represent such data.
- Diverse functionality. Compared with traditional business systems, e-commerce systems enable more functionality. Indeed, e-commerce systems must diversify their functionality to be successful. This functionality should enable a wide range of activities including product location, negotiation, payment, delivery options, and after sales services in a convenient and user-friendly manner.
- Correctness of operations. E-commerce solutions must ensure the correctness of e-commerce transactions. In an e-commerce environment, business partners are not likely to meet physically to attest to their characters. Therefore, correctness of operations is crucial to retain customers' confidence and sustain their patronage.

A successful e-commerce solution is an amalgam of data from disparate data sources that supports rich information content and brings various application components with diverse functionality into one cohesive, coherent, and efficient e-commerce system.

2.1.4 Infrastructure for E-Commerce

In a traditional business setting, certain facilities (office space, warehouse, mall, telephone, power, etc) have to be in place for an organization to carryout its business transactions. Likewise, for e-commerce to succeed there has to be basic infrastructure in place. This infrastructure includes network infrastructure, payment infrastructure and legal infrastructure [Kos97, Raj00, SQS+00]. This thesis extends the list of e-commerce infrastructure to include data infrastructure, which is often neglected by many e-commerce systems' researchers.

1. Network infrastructure. Network infrastructure facilitates the creation of an Internet marketplace. Network infrastructure involves interconnectivity among telecommunications, cable, satellite, or other Internet facilities. Internet service providers (ISPs) connect market participants on the Internet to end-user devices such as PCs, TVs, mobile telephones, or other PDAs. The ISPs provide Internet access (physical connections) to organizations and individuals through various channels. Developments in the area of network infrastructure are yielding positive results, as the cost of access to data through high speed Internet connectivity is relatively low.
2. Payment infrastructure. Payment infrastructure links the traditional marketplace to the digital marketplace through electronic payment mechanisms, such as credit cards, debit cards and e-money, in transactions that are secured against intentional and unintentional abuses. Payment infrastructure facilitates and reconciles payments among parties, thus making it possible to exchange money for goods and services on the Internet. For business transactions in B2B systems, the Automated Clearing House (ACH) [Nel02] is used to transfer funds between accounts. In all such payment mechanisms, there has to be an interface between e-commerce enterprises and the various payment networks to enable automated processing of payments.

3. Regulatory infrastructure. Regulatory infrastructure includes the protocols, laws, and regulations that promote secure e-commerce transactions. This infrastructure affects the conduct of electronic commerce, as well as the relationships between businesses, consumers, and government. Examples include technical communications and interconnectivity standards; the legality and modality of digital signatures, certification, and encryption; and disclosure, privacy, and content regulations. Established protocols and legal frameworks offer a relatively secure environment to regulate transactions and protect participants from abuses. Many emerging technologies, including digital signatures [Kee03, KG03, SV03] are being designed because of legal and technological developments, along with strong market demand for secure transactions on the Internet. The US government has put the responsibility of addressing legal issues on the online sites, so that they can increase their security apparatus and improve on the level of customer trust. Samuelson *et al.* [SDG92] argue that those responsible for risk are the most driven to prevent it. This position tallies with the US government approach to the issue of the security of online sites.
4. Data infrastructure. Data infrastructure involves managing the large volume of e-commerce data that is transferred and exchanged among participants in e-commerce transactions to provide easy access to such data on the Web. Data infrastructure development has continued to lag behind other e-commerce infrastructure, and fails to provide the needed access and transparency to e-commerce related data on the Web. To this end, users are currently saddled with the responsibility of locating and interpreting the enormous amount of Web data. This research addresses the issue of data infrastructure development in e-commerce systems.

2.1.5 Challenges of E-Commerce Transactions

E-commerce has grown rapidly over the years with a lot of organizations deploying e-commerce enabled applications on the Web. Regardless of the increasing volumes of e-commerce sites, e-commerce transactions come with some challenges. Maamar [Maa03] identifies the following challenges that affect e-commerce operations: 1) The user has to identify the relevant Web sites to access catalogues; 2) It is necessary for the user to

understand how the Web sites operate; 3) The user has to specify his/her needs according to the characteristics or terminology of the Web sites; and 4) there are also security concerns in dealing with sensitive information, such as users' account information. Although, e-commerce systems bring relative comfort and convenience to businesses and customers, they also have their attendant risks. There is a very high level of risk, with relatively low trust management because such systems are fairly new and the openness of the Internet exposes e-commerce transactions to attack by users with criminal intentions. However, the issue of secure e-commerce infrastructure has been largely addressed in the literature [Evi03, Tyg96, Ver03].

Points 1 to 3 above fall in the domain of data infrastructure (identifying, interpreting and use of standard formats) for e-commerce data on the Web. Customers often face the problem of identifying e-commerce sites on the Web and data relevant to them. The vast quantity of information on the Web makes this task very demanding. For example, there may be huge sales in a particular area that a customer mostly desires, but the customer may not be able to take advantage of the sales because of lack of access to the relevant information. E-commerce systems also have rich media content (audio, sound, image, and text) and require clear descriptions of items to be represented in a manner easily accessible and understandable to the user. The problem of easy access to information on the Web is fundamental to the success of e-commerce systems since users need to locate and interpret data to carry out successful transactions. Web users and, in particular, e-commerce users desire an environment where data is easily accessible and visible to all the participants. Therefore, a model that makes e-commerce data on the Web visible to all users is desirable.

2.2 Schema and Data Integration

Information integration has the goal of providing an integrated and coherent interface to heterogeneous data sources. It is one of the major problems limiting access to data on the Web. The Web has not only provided people with unprecedented access to online information and services, but also motivated the creation of numerous new sources. Regrettably, however, meaningful access to the enormous amount of information on the

Web is a major source of concern to the community of Web users [PDEW97, PE95]. The broad spectrum of data available on the Web emanates from different information sources that are independent and stored using different formats. Vianu [Via01] examines the challenges and opportunities of the Web and concludes that the Web presents a shift in paradigm when compared with traditional databases. Silberschaltz and Zdonik [SZ96] as well as Vianu argue that the disparate[10] nature of Web data sources makes it difficult to query the Web as one huge database and suggest reengineering classical database theory to be relevant to the data management needs of Web data. Integrating data sources involves combining their concepts and knowledge into an integrated view that insulates users from the complexities of the underlying data sources. The goal of information integration is to access, relate and combine data from disparate information sources, to provide a simplified uniform interface to all the data sources available [ACM+99, LW00].

Information integration can be at the schema level or the data level. Schema-level integration (schema integration) involves combining two or more local database schemas into one coherent and cohesive global schema. Data-level integration (data integration) involves the integration of data at the data item level. The interest of this thesis is integration at the schema level. However, the terms data integration and schema integration will be used interchangeably. Data integration enables Web data to be consolidated, standardized and made readily available to the numerous users of Web information, including the research community, business partners, and customers for e-commerce decision support solutions. However, an integrated use of Web-based data demands that we have information about its meaning and the structure of its organization in clear terms. Integrating data from different sources involves combining their concepts and knowledge into a global view. Users can then query the global schema without having to know the underlying system organization of the individual data sources. That is, a mediated schema provides users with a uniform interface to a multitude of data sources, thus protecting the users from needing to know both the details of the sources and the particular mode of interaction with each source. Different methods exist in the

[10] Disparate data sources are information sources that are independently distributed and potentially heterogeneous [LBGR99].

literature [BCV99, GBMS99, GPQ+97, Lev01] to address the issue of data integration from disparate data sources, including the Web.

Database integration is a relatively recent problem that has appeared in the context of distributed databases, and has attracted a lot of attention from the database research community. Researchers have been studying the issue of schema integration since the early 80's and the dramatic explosion in data due to the advent of the Internet has made the problem even more complex [BLN86]. A distributed database refers to a collection of data that logically belong to the same system but are distributed over the nodes of a computer network. There are two broad classes of distributed databases: homogeneous databases, where the local databases have the same data model and identical Database Management System (DBMS) and heterogeneous databases, where there is diversity in data models and/or DBMSs. This research considers heterogeneous databases, which present greater challenges to data integration because of the diversity of the data sources.

2.2.1 Challenges of Integrating Heterogeneous Data Sources

The task of integrating data from heterogeneous data sources is difficult because of the differences in the underlying data sources. Differences arise between different data sources because of conflicting representations of two or more schemas. These differences arise because of the designers' perception, different information needs or using different tools to express such perceptions. Indeed, two designers modelling the same problem space, or even two overlapping spaces under consideration, may not necessarily describe the real world state (RWS) of objects[11] in the same way. Different authors [BLN86, Law01, LBGR99, SK93] have classified schema heterogeneity in different ways at varying degrees, which sometimes coincide, overlap or differ. We classify schema heterogeneity into two broad classes – structural and semantic heterogeneity.

Structural Heterogeneity

Structural heterogeneity refers to a situation where designers use different data models (including structured, semistructured and unstructured data models), formats, and

[11] Larson *et al.* [LNE89] defined the real world state of an object A, denoted RWS(A), as the set of instances of the object A at a given time.

constructs to represent the same real world objects. Subtypes of structural heterogeneity include:

- Behavioural heterogeneity, which refers to differences arising from the design decisions as a result of different policies adopted for identical concepts such as insertion, deletion or update. For example, in a products catalogue, products are organized according to categories. In one schema, deleting the last product in a given category leads to the deletion of that category, while in another schema, deleting the last product in a particular category does not necessarily affect the category.
- Data model heterogeneity, which refers to differences arising from data represented using different models. This kind of heterogeneity arises because differences exist in the semantics of data models. For example, the inheritance property of the object-oriented model is lacking in the relational model.
- Interschema properties (ISPs) refer to differences that arise in semantic relationships when schemas are combined. In particular, semantic relationships between the set of different concepts that are mutually related by some semantic properties. ISPs create additional constraints that hold in the integrated schema but are not specified in any of the local schemas. ISPs are semantic relationships between different sets of objects in two or more schemas.
- A terminal element may be defined as a key element in a given schema. The contents of a key element by definition must not be null and it must be unique across the documents in which it occurs. Key conflict arises when: two different elements are defined as keys; or two elements have the same name, where in one schema the element is a key, while in another, it is not a key.
- Data type heterogeneity arises when constructs with different representation capabilities are used to describe corresponding types. For example, in a relational data model an entity type may correspond to a relation type. In an object-oriented data model, an object class may correspond to an attribute type. In a semistructured data model, "Address" might be modeled as a terminal element, as in [Address] in one schema, and as a non-terminal element as in [Address [Street, Postal-Code]] in another schema.

- Data/model conflicts arise when the value of data in one source matches the type name (metadata) in another source. For example, the value of data in one source may correspond to the value of an attribute type or even the value of a relation type in another source. Schema A may have "Machine" as an attribute whose value is "Computer". Schema B may have "Computer" as a table with "Type" and "Model" as attributes. Notice that "Computer" in schema A is the value of an attribute, while in schema B "Computer" is a table with two attributes.

Semantic Heterogeneity

Semantic heterogeneity arises when designers perceive related sets of real world objects in different ways. Examples of semantic heterogeneity include, but are not limited to the following.

- Naming conflicts may be a problem including synonyms where different concepts have the same name, or homonyms where the same concept is given different names. Naming conflicts may occur in a situation where data in different systems are given different interpretations due to the designers' different perceptions of the same set of real world objects. For instance, a schema may have an "Employee" object class, while another schema may have a more restricted "SD-Employee" object class (i.e. a group of employees in the Sales Department).
- Format conflicts arise when different conventions are used to represent the same entity. For example, one representation of University of Manitoba, Canada could be "University of Manitoba", and another could be "U of M". Another example is the date format used, "dd/mm/yyyy" vs. "mm-dd-yyyy".
- Measurement conflicts also arise when different scales and units are used to represent equivalent entities. For example, temperature measured in degrees Centigrade, degrees Fahrenheit or Kelvin; speed measured in miles/hour or kilometres/hour.

Resolving these heterogeneities has been the focus of active research in the field of data integration [CGH+94, Coh00, GBMS99, FLM98]. In an ideal situation, an efficient integration system is expected to resolve all the different conflicts mentioned above. The

way and manner an integration system resolves conflicts greatly enhances its applicability. The result of an integration system should be able to provide a system that transparently presents data within the space of discourse. Transparency is a vital asset in determining a suitable integration system. An integration system that is transparent insulates the users from the different data structures and organizational characteristics of the individual data sources.

There are several integration approaches that are used to resolve some of the above heterogeneities. Each of these approaches follows a particular solution technique to resolve heterogeneities among different systems. Regardless of the particular technique chosen, the integration systems follow some or all of the steps given below.

- Pre-integration. The systems to be integrated are analyzed to make certain decisions that will direct future actions. The choice of schemas to be integrated and the platform to use is determined at this stage. Also, the integration approach to adopt and the assignment of certain preferences are also made. This phase is more of a preparatory phase in the integration process.
- Correspondence identification. The correspondence identification step is devoted to identification and description of inter-schema relationships. Here, relationships and conflicts among the schemas are identified, and views are modified to conform to one another. At this stage, input schemas are transformed to make them homogeneous (both syntactically and semantically) and the appropriate information sources are located. Creating a common understanding of existing data is a fundamental condition to successful database integration. In this regard, input schemas are usually transformed to make them as homogeneous as possible.
- Integration process. This is the final step where inter-schema conflicts are resolved, and corresponding items are unified into an integrated schema. The integrated data can then be defined as a view (global) of the sources (local views). A query interface to the global schema is constructed and queries posed against the integrated view must then be translated into queries against the sources.

2.3 Integration Approaches

The different approaches to integrate heterogeneous data sources include:

1. Hard-Wired Approach. In the hard-wired approach [HKR+00], the integration system is comprised of data sources that are hard-wired to allow some applications to communicate with several data sources. However, the hard-wired approach is inflexible because replacing one data source with another involves rewriting a portion of the application. Furthermore, queries to data sources are also hard-wired in the application. This inflexible process is not amenable to heterogeneous data sources that are dynamic.
2. Warehouse Approach. In the warehousing approach [Wid95], data from multiple data sources are used to populate a data warehouse. The data warehouse is later used to analyse queries. Compared with the hard-wired approach, the warehouse approach allows greater flexibility in retrieving and comparing data. However, the warehouse approach comes with the additional cost of maintenance and re-implementing or losing the specialized functions of the original data sources. Maintenance transactions in the warehouse isolate the users from read activity, thus limiting the availability and/or size of the warehouse. The advantage is that performance can be guaranteed at query time.
3. Database Middleware Approach. The use of database middleware systems [SL90] allow the creation of a global schema that combines data from multiple data sources and presents such data to the user without creating a physical warehouse. The global schema presents a virtual database that facilitates the query of inherently distributed multiple data sources.

The database middleware approach is popular within the research community. The systems described in this thesis follow the database middleware approach. The subsections that follow classify middleware integration approaches, hereinafter referred to simply as integration approaches. This classification may not be all-inclusive, and the classes are not mutually exclusive as some classes may overlap with one another.

2.3.1 Early Integration Methods

Most of the existing methods in schema integration are based on schema models that allow the description of data in an abstract and clear manner. Basically, the aim of an integration system is to translate the different views of data into a global view. The manipulation of the schema models to produce an abstract, global view of the models is often done with a substantial amount of manual effort. In manual integration systems, the integrated schema, which results from editing efforts of the database administrator (DBA), is built as a view over existing pre-determined local databases. A functional model improved with generalization is used to describe both the input schemas and the integrated schema also called the "superview" [MB81]. Queries posed on the superview are mapped into the individual databases, and the answers are recomposed to form the effective answer to the original query. This mapping, which is stored with the integrated schema as a virtual database, is derived from the editing process. In MERMAID [TBC+87], for example, the goal is to integrate relational databases using a mainly manual approach, although the DBA is assisted to some extent by the system. First, the DBA defines all underlying semantic domains to extend the existing schema. The aim of the extension is to allow the system to determine relations that have a common semantic domain. These relations are presented to the DBA who has the responsibility of selecting which relations and attributes to include in the integrated schema. The use of abstract data types to add meaning to relational schemas was suggested, and also, there have been suggestions to develop superview definition languages using an object-oriented approach (e.g. [MB81]). Early integration approaches were more interested in providing a framework and tools to assist humans in information processing and integration activities than in performing fully automated information integration without user participation. Unfortunately, the explosion of information sources, due mainly to advances in the Web, makes manual effort expensive, laborious and error prone, and therefore calls for some form of automation of the integration process.

Batini *et al.* [BLN86] was one of the earliest survey studies carried out in the area of schema integration and it is still very relevant despite advances in the research. Among the different algorithms that were considered in the early integration systems, the

following problems were identified: different users' perspectives, the use of different model constructs and naming conflicts. Integration problems were seen from two contexts, view integration and database integration. View integration focuses on database design and produces a global view of a proposed database; while database integration focuses on distributed database management and produces a global schema for a set of distributed databases. Ram and Ramesh [RR99] observe that view integration belongs to the domain of top-down database design because view integration aims to have a common view of several user schemas. View integration methodology, good as it was at the time, needed some extensions to support the then emerging distributed databases. View integration methodologies could also be extended to include databases with special properties (for example, statistical, scientific and medical databases). However, such extensions have many problems that may be difficult to handle using the early integration algorithms, which are no longer popular.

Batini *et al.* have broadly classified early integration approaches into two groups, relational and semantic models. The integration approach in the relational model is based on the universal relation assumption, which states that the names of all attributes are unique across the database [MVU84]. Due to this assumption, the proponents of the relational model ignored naming conflicts (an unrealistic assumption). Despite the inherent limitations of the expressive power of the relational model resulting in incompleteness of the global schema, it is still relevant to the research community. This is so because the model is widespread, simple and there is an associated powerful query formulation language [RR99].

Semantic models possess richer semantics than the relational model so they are able to resolve conflicts without the assumptions of the relational model. The quality of earlier algorithms were evaluated on the basis of completeness and correctness, which means that the global schema must correctly represent all the component schemas; minimality, which means that no concept in any of the component schemas must be represented more than once in the global schema; and understandability, which means that the global schema should be easily understood by the end users. Although, the earlier integration algorithms had some degree of success, they relied on significant user participation.

2.3.2 Mediator / Wrapper - Based Systems

A mediator [Wie92] is a program that collects information from multiple data sources to provide an integrated view of the data sources. These data sources may also include other mediators. According to the classification given by Hull [Hul97], a mediated schema is an integrated read-only view approach to data integration. In this approach, a mediator provides an interface for the data that resides in the databases, but does not update them. It is one of the earliest approaches to integration of heterogeneous data sources and remains the most pervasive. In general a mediator provides an integrated view over multiple data sources. A Wrapper is a program designed to extract content from a particular data source and deliver the content of interest in a self-describing representation to another data source [Lew91]. The wrapper acts as an interface between an application and a data source, converting queries into sub-queries understandable by the underlying data source and transforms the resulting data into a format understood by the application.

Garlic [HKWY97, HMN+99] is a wrapper/mediator-based system, designed to address access to large information systems. Garlic derives its strength from powerful wrappers and a robust query processing mechanism capable of handling complex operations. In Garlic, the component export schemas are tightly integrated within an object-oriented data model and the system optimizes and executes queries over diverse data sources in an object-extended version of SQL. Garlic ignores heterogeneity in schemas, but handles differences in query capabilities using a flexible query optimizer.

Infomaster [GKD97] provides integrated access to multiple distributed heterogeneous information sources giving the illusion of a centralized homogeneous information system. It is based on a global schema completely modeled from local data sources and a core system that dynamically determines an efficient plan to answer the user's queries by using the translation rules that harmonize heterogeneous data sources. Schema translation is an operation that involves mapping one schema into another when the two schemas are represented using different data models [MIR93, AT95]. Infomaster uses rules and constraints to describe information sources, and translations among heterogeneous data sources. Translation rules relate each source to the reference schema.

However, Infomaster lacks the necessary formalism to guarantee the correctness of such translations.

The Stanford-IBM Manager of Multiple Information Sources (TSIMMIS) [CGH+94] focuses on the development of tools to facilitate integration of heterogeneous information sources. The system propagates all schemas of the data sources' wrappers to a global view. TSIMMIS is a query-centric system that selects a set of queries and provides a procedure to answer each query from the set of queries using available data sources. Queries on the global view are mapped into views or queries on the data sources using query languages or logical rules.

The Mediator envirOnment for Multiple Information Sources (MOMIS) [BCBV01, BCV99] is a mediator - wrapper based framework for extracting structured and semistructured data. The MOMIS approach uses object patterns to support the integration of heterogeneous sources with structured databases. An object-oriented language, called Object Definition Language for Intelligent Integration of Information (ODL-I3), a derivative of the standard Object Data Management Group (ODMG) [CBB+02], is used for information extraction. The integration process uses a semantic approach based on a description logic component and a cluster generator module in a semi-automatic manner to explore the knowledge of Thesaurus[12] (defined by the framework) and descriptions of data source schemas defined in ODL-I3. The integration process generates a virtual integrated view of the local data sources (global schema) for which constraint mapping rules are specified to allow the resolution of conflicts.

Context Interchange Strategy (CIS) [GBMS99, LBGR99] is another mediator-based approach used in achieving semantic interoperability among heterogeneous sources and users of the sources. CIS focuses on the semantics of individual data items as opposed to conflicts at the schematic level. In this approach, semantic conflicts among heterogeneous systems are identified dynamically by a context mediator through comparison of context axioms corresponding to the systems engaged in data exchange.

Lee *et al.* [LBGR99] propose the Gestalt approach, which exploits the natural intentional heterogeneity (intentional heterogeneity refers to the differences in the

[12] A thesaurus for an integration system is a terminological relationships for schema class and attributes of different data sources.

information content of the different sources) of data sources available over the Internet. In the Gestalt approach, the responsibility of maintaining the "content" is distributed across various sources. Here, the integrator simply publishes a schema for the integrated view and the administrators of the relevant data sources are given the opportunity to independently register their sources into this view.

Application Manifold (AM) [EM01] is also a mediator-based data integration system that offers a declarative specification language for describing the integration/customization tasks. The system supports a modular approach where new applications can be easily added and integrated, thus ensuring extensibility.

The Information Manifold (IM) [KLSS95, LSK95] system integrates structured data sources on the Web. IM derives its strength from source descriptions and efficient query processing. Heterogeneous sources are hidden from the users using a mediator designed according to the information needs of the users. IM is a source-centric integration system that separates the declarative source description from the actual detail of interacting with an information source. Each concept of a component schema is mapped to the mediator schema using a declarative language. The system describes information sources independent of the queries that are subsequently posed on them. Given a user query, the system uses the descriptions of the data sources to identify relevant sources, executes the sub-queries on the data sources, and collates the results. Local data sources are autonomous and new sources are easily added to the integration system because the data sources are independent of the mediator and there is explicit description of the relationship between the data sources and the mediator. IM, in its present form, cannot handle unstructured information sources.

Tukwila [IHW01] is a query-centric integration system for processing network-based XML data sources. The Tukwila engine avoids the complex task of reading, parsing, and storing data before querying, but can return query results dynamically. Passi *et al.* [PMMB02] adopt a layered approach to integrate XML schema to facilitate e-commerce. This layered approach follows the traditional three-layer technique (pre-integration, comparison, and integration) [BLN86] to integration.

Most of the systems discussed [BCV99, CGH+94, GKD97, HKWY97, KLSS95] are academic prototypes that depend on heuristics for wrapper generation that lack sound

mathematical basis to prove the correctness and/or completeness of the transformations they perform. In particular, some of the systems [KLSS95, LSK95] are restricted to components based only on structured data models, and hence, are too inflexible to handle Web data integration. Another essential ingredient lacking in these approaches is the specification of the different layers (of the three-layer technique) to enhance understanding. In our approach, we formally specify the components of the integration systems to elucidate the integration process.

The systems described in this section fall under two categories – query-centric and source-centric systems. Query-centric systems [LRO96] (TSIMMIS, MOMIS, Tukwila, etc.) allow users to combine data from multiple data sources in a single query without creating a physical database. These systems choose a set of queries, and for each query they provide a procedure to answer the query using the underlying sources. Algorithms in the query-centric approach answer a new query by trying to relate it to existing queries. However, adding a new data source is always a problem because the new source may cause associated views of elements in the global schema to be redefined [Len02]. In source-centric systems (AM, IM, CIS, etc.), the content of each source is characterized in terms of a view over the global schema. Queries are posed to the system in terms of the global schema [LRO96]. In this approach, systems are not restricted by which queries they can answer. It is also easier to add or delete sources because query specific procedures are not modified to accommodate the changes. This research adopts the source-centric approach.

2.3.3 Industrial and Special Application Systems

Industrial systems [BCE+93, SK93] adopt a more *ad hoc* approach to integration. Sheth and Karabatis [SK93] describe a technique that is based on specifying dependencies among databases so that the system can guarantee consistency among databases in a multi-database system[13]. This approach is amenable to industrial needs because custom coding, although costly and laborious, is relatively simple. It is also relevant to the

[13] A multi-database system is a distributed database system where each participating site maintains complete autonomy.

transaction management problem as many global transactions are carried out by consistency enforcement.

Global consistency of data sources is also the driving force behind InterBase [BCE+93]. InterBase uses a flexible transaction manager that ensures global consistency of data sources to provide a framework for integrating different data sources. InterBase provides interfaces to the individual data sources for access to data. InterBase is an example of a practical system for industrial applications, despite the *ad hoc* integration approach that depends on substantial manual input. Other similar industrial projects that rely on significant user intervention include [ABC+95] and [PKGV95].

There are also domain specific efforts to standardize documents and messages for e-commerce applications based on eXtensible Markup Language (XML). An Electronic Business using the eXtensible Markup Language (ebXML) [W3C02] is a domain-independent standardization effort that aims at developing a domain and application-independent standard for exchanging business data. One problem of ebXML is that it provides only a high-level specification of documents and messages (i.e. only tags and terminology is defined) but does not define the structure of business documents. Such structure has to be defined by the business partners or by a particular industry.

BizTalk [Mic00] allows data to be exchanged between systems using standardized XML schemas. The Metadata Interchange Specification (MDIS) [Met97], developed by a consortium of database and software experts, is another emerging standard for specifying and exchanging metadata.

The Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT)[14] is a United Nations Economic Commission for Europe (UNECE) initiative to specify syntax rules for the preparation of messages to be exchanged between partners. EDIFACT has been used among big partners who set the standards for several years for B2B e-commerce. A major problem with all these approaches is that they define schemas for data exchange. In particular, Fensel [Fen01] notes that the EDIFACT standard is rather procedural and difficult, thus making the programming of business transactions expensive, error prone, and laborious to maintain.

[14] EDIFACT (Available at): http://www.unece.org/trade/untdid/welcome.htm

Industry efforts, including BizTalk, Standard Interchange Language (SIL) [UCC99], and Electronic Data Interchange (EDI) [SS91], which use standardized structures and formats as a vehicle for information exchange are not appealing because they lack the needed flexibility to accommodate legacy systems and the corporate databases of most enterprises are mainly in legacy systems. For example, EDI is faced with the following limitations which has encouraged a shift to Internet technologies [Cha01, Hal02, TIKL04]:

- EDI services can be expensive since the cost is based on the volume of the transactions.
- EDI mainly depends on third-party connectivity and proprietary technologies.
- EDI does not adequately support tight end-to-end process B2B relationships and it lacks integration with internal systems.
- EDI is complex, difficult and expensive for smaller companies to use.

A flexible framework that provides mechanisms for legacy systems to interoperate with other subsystems will be attractive to Web-based systems. However, such systems should be backed with sound mathematical formalism and a theoretical foundation that can guarantee correctness.

Omelanyanko and Fensel [OF01] and also Fensel *et al.* [FDO+01] propose a layered approach to the integration of product descriptions in B2B e-commerce systems where the integration task is decomposed into several subtasks using a divide-and-conquer approach. Omelanyanko and Fensel discuss the problems inherent in mapping from one product catalogue to another. These problems include heterogeneity in product, catalogue, and document description standards.

Stonebraker and Hellerstein [SH01] discuss the integration of catalogues in a B2B scenario pointing out some of the integration challenges as a result of the nature of e-commerce data. The paper adopts a materialized view approach to integration to support catalogue heterogeneity and transformations.

Marrón *et al.* [MLW03] propose an adaptive query evaluation strategy to query XML-based electronic catalogue without explicitly rewriting the queries. This approach allows the same query to be used on all local catalogues.

Agrawal *et al.* [ASX01] and Jhingran [Jhi00] discuss the use of database technology to support e-commerce. In particular, Agrawal *et al.* examine ways of supporting nameless querying of databases, efficient storage and a query mechanism for e-commerce data, as well as techniques to merge catalogues from different data sources.

DiscoveryLink [HKR+00] extends the Garlic integration system to create a virtual database for life sciences data from heterogeneous data sources. DiscoveryLink derives its strength from a flexible wrapper mechanism that adapts to the data sources and a query optimization technique capable of handling complex queries. Davidson *et al.* [DCB+01] also developed Kleisli, an integration system for biological data that allows the expression of complicated transformations across heterogeneous data sources.

Chen *et al.* [CKMS97] introduces the Object-Protocol Model (OPM) that leverages a flexible object model and a query optimization mechanism to achieve integration. The problems with these [CKMS97, DCB+01, HKR+00] systems include: 1) the existence of alternate sources semantics; 2) contents overlap because of multiple views of objects; and 3) requirement for substantial user participation in the integration process.

2.3.4 Artificial Intelligence Based Approaches

There is growing interest in applying Artificial Intelligence (AI) to research problems related to the Internet, especially data integration [LW00, BCV99]. We briefly highlight a few such efforts.

Bergamaschi *et al.* [BCV99] propose intelligent tool-supported techniques for information extraction and integration in both structured and heterogeneous data sources. Their approach uses object patterns[15] to support the integration of heterogeneous sources with structured databases.

The MOMIS project [BCV99] uses a semantic approach based on a description logic component and a cluster generator module. Information integration is performed through an intelligent extraction and analysis process, followed by a unification process. The unification process provides a common interface. Calvanese *et al.* [CDL+98] discuss the fundamental feature of a declarative approach to information integration based on

[15] An object pattern is representative of all different objects that describe the same concept in a given data source.

description logics[16]. The authors present a methodological framework to conceptual modeling that depends on fundamental reasoning services. These reasoning services include checking whether the domain model is consistent, checking whether a concept is satisfiable in the domain model and checking subsumptions between relations or concepts in the domain model.

Recently, systems have been developed based on the use of description logics [CCG+01, Lev01] and machine learning techniques [DDH01, DDH00, DP97, LC00]. All of the AI approaches are based on modern query decomposition, sending sub-queries to source databases, and merging the answers that come back. Recent systems based on description logics are focused primarily on conjunctive queries (i.e. those expressible using select, project, and join). Data held in the source databases is expressed as views over a global schema. Machine learning provides an attractive platform for finding semantic mappings because they are able to learn from previous experience.

There is also a growing interest in the AI research community to apply AI techniques (mostly knowledge representation and planning) to data integration problems. The nature and size of Web data sources often necessitates the use of approximate and heuristic techniques that form the core of AI solutions. A range of machine learning algorithms has been applied to schema integration problems (because of their ability to learn patterns) [DDH01, DDH00, DP97, LC00, LCL00, TKM96]. A program is said to learn if it has the ability to accumulate knowledge, draw conclusions from meaningful information, discover unknown knowledge from raw data, and improve its performance. These features put machine-learning approaches in a position to provide solutions for schema integration of heterogeneous data sources because they can learn semantic relationships between objects in different data sources and produce meaningful rules. Current research efforts are geared towards fully integrated support for Web data instead of using manually driven integration algorithms.

SEMINT (SEMantic INTegrator) [LC00, LCL00] is a neural network learner-based system used to develop a semi-automated semantic integration procedure that uses the metadata available in the database systems to identify attribute correspondences.

[16] Descriptions logics belong to first order logic which enable concepts to be expressed. They can be viewed as logical formulas built using unary and binary predicates.

SEMINT is a tool designed to assist in identifying correspondences in heterogeneous databases whose meanings in the real world do not differ. For instance, student grades in two databases will probably be similar in both design and values. On the other hand, student ages are different from their addresses. In its main approach, SEMINT uses neural networks to determine metadata available in databases for attribute correspondence identification that facilitates the integration of data sources. This approach is based on the assumption that attributes in different databases that represent the same real world concept will have similarities that will facilitate clustering[17] data objects. The use of a neural network learner is motivated by the fact that neural networks are known to be a powerful pattern recognition technique. The ability of neural networks to learn similarities between objects in different data sources distinguish them from traditional programs that are pre-programmed [LC00]. SEMINT is, however, limited in the attributes it can learn because it uses a single neural network learner, which may limit matching accuracy. In particular, SEMINT only identifies a match to attribute clusters and not individual attributes.

Large databases present different patterns that may be difficult for a single learner alone to learn, and use to make good predictions about relationships between objects in different data sources. To enhance the quality of its predictions in schema integration, Learning Source Descriptions (LSD) [DDH00, DDH01] uses multiple neural network learners (called multi-learners) to address this problem. Multi-learning strategies apply a set of learners, each of which looks at the problem from a different perspective, hence covering different types of domain knowledge. It is easy for multiple-learning strategies to incorporate new learners, which widens their scope of application and potentially improves matching accuracy. Multi-learning approaches are able to combine independent single learner systems. LSD uses machine-learning concepts to match schemas in a new data source against a previously determined global schema, producing a one-to-one mapping between elements of the two schemas (e.g., 'location' could be mapped to 'address'). That is, mapping one data item from the new source schema to another data item in the global schema. Mapping could be specified as a query that transforms instances of the source schema into instances of the global schema. LSD is a learner-

[17] Classes with semantically related objects in different sources are grouped together in clusters.

based approach, that learns from a small set of data sources provided manually. It gathers sufficient information from the data sources that have been manually mapped to the global schema and then proposes mappings for subsequent data sources with a very high success rate. LSD uses a name matcher in addition to other learners (matchers). It uses several instance-level learners[18] that are trained during a processing step.

In learner-based approaches, attributes' semantics are learned and not pre-programmed, thus making it possible for learner-based systems to adapt to changes with the growth of the information sources. With LSD, both SEMINT and ILA (Internet Learning Agents) [PE95] could be used as new base learners and their predictions would then be combined by the meta-learner [PDEW97]. Mappings produced by LSD could also be used as input to other systems, including rule based systems. Unfortunately, it takes time to process the multiple learning algorithms involved in the multi-learner approach [DDH01]. There is also an appreciable level of overlapping source schemas, which reduces matching accuracy and the use of user feedback in LSD could be time consuming.

2.3.5 Rule-Based Systems and Correspondence Assertions

Rule-based systems employ description logics. In description logics, complex hierarchical structures are modeled using a set of logics [GBMS99]. They provide a well-founded logical framework in which concepts are intentionally defined using descriptions built from a set of constructors. Matching is then performed based on both the names and structure of schema elements. Merging two or more schemas into a global schema demands the identification of common concepts and also the set of different concepts that are mutually related by some semantic properties. Identifying semantic relationships between a set of objects in one schema, S_1 and a different set of objects in another schema S_2 is crucial to the integration process. A commonly used technique is to map the objects of one schema into the other. A declarative statement is associated with each mapping to show how the elements of S_1 and S_2 are related. A statement that establishes the relationship between the semantics of objects in two different schemas is called a

[18] Instance –level learners are interested in contents of an element, instead of its tag (name).

correspondence assertion. Every assertion is analysed to generate a corresponding integration rule.

Spaccapietra and Parent [SP94] identified four kinds of correspondence assertions for declaring semantic relationships between objects to eliminate naming and semantic conflicts among component databases. The four correspondence assertions are equivalence assertion ($\equiv$), inclusion assertion ($\subseteq$), intersection assertion ($\cap$) and exclusion assertion ($\varnothing$). Chen [Che00] includes derivation assertion to accommodate 'more heterogeneity'. Derivation assertion is a more complex assertion that can capture simultaneous relationships between objects in different data sources.

Previous efforts [BLN86, SL90] could not establish the real semantics of objects from different schemas because of the lack of understanding of the different contexts, and therefore, could only capture limited semantics of attributes. Rahm and Bernstein [RB01] observe that existing approaches used in mapping elements of two different schemas are based on heuristics that cannot be formalized and thus, the mapping results are mathematically unsatisfying. For instance, the techniques used in [SK92, SP94] lack the necessary formalism for correspondence assertions and hence, it is difficult to prove their correctness. In addition, no analysis of assertions has been carried out to minimize redundant operations by using the characteristics of correspondence assertions. Earlier integration strategies [MB81, ME84, TBC+87] depend on significant manual input from the users and the ingenuity of the DBAs for identification and declaration of assertions. We argue that it would be exceedingly difficult for users/DBAs to identify, declare and specify assertions between objects, especially a mapping between two different attributes, for all attributes in the local databases. Furthermore, specifying mappings between all possible pairs of attributes is time consuming, tedious, and error prone (specifying mappings between all possible pairs of attributes is an $O(n^2)$ process, where n is the number of attributes [YSDK91]).

Larson *et al.* [LNE89] present definitions of attribute equivalence, which are used to identify and specify the relationship between a pair of attributes. Yu *et al.* [YSDK91] develop a scheme to semi-automatically determine attribute relationships. However, the tasks of identifying and specifying assertions are left to the user. Sheth and Gala [SG89] argue that automatic schema integration (especially automatically identifying

correspondence assertions) is impossible because of multiple views of the real world state. Spaccapietra and Parent [SP94] propose a path integration rule to integrate two semantically equivalent paths. The path integration rule, however, has no formal basis in the representation of the integration result. To add formalism to the path integration rule, Chen [Che00] developed a derivation assertion that could be formalized. This derivation assertion depends on the ingenuity of the DBA and a complex and inflexible data model for the integration process.

2.3.6 Ontology-Based Systems

An ontology provides an explicit specification of a concept that is shared. Ontologies have been proposed [FG97, UKMZ98, WSSK99] as an integration approach. (Additional details concerning ontologies are provided in Section 2.4.) For the moment, we only review related work on integration that is based on using ontologies.

The use of metadata in describing the structure and semantics of Web data has been a focus of much research. Singh [Sin98] suggests an efficient and cost effective method of data integration that involves the use of metadata specifications of the information models of the sources and user vocabulary. The system automatically deduces how to handle a request at runtime by examining the metadata specification that the machine can process. The distinctive feature of their proposed system is the expressiveness of the metadata specification language, which enables sophisticated coordination. Automated coordination based on metadata can reduce cost, increase efficiency, and enable a larger class of users to take advantage of the available information. The system develops a plan at runtime to handle a request.

In [Bor99], a representation model that enables the explicit description of the structure and semantics of semistructured data together with metadata that describes the model's organization and semantics was introduced. An ontology that provides a commonly agreed upon vocabulary was then used to provide a common view for the integration of data.

SIMS (an acronym for Search in Multiple Sources) [ACHK93] uses an ontology for integrating multiple information sources. The representation of the ontology in SIMS is based on description logic. The system leverages the expressiveness of description logic

to integrate data sources with different data models. SIMS also allows data source autonomy and the evolvability of the system as elements from each data source are mapped independently to the mediator ontology.

Barker and Lawrence [BL02] propose an XML-based strategy that combines a standard term dictionary and an *ad hoc* schema integration mechanism to achieve B2B integration. They adopt a two-phase approach to integration. First, local schemas are translated into a relational model using XML. Then, the integration algorithm involves simple matching of terms from the local schemas onto the integrated schema.

Although, the use of a common ontology in data integration systems resolves the problem of multiple views of objects, systems using such an approach still have to contend with other integration challenges. Central to these challenges are: 1) the need to formally specify components of the integration system to guarantee correctness; and 2) the use of a flexible data model to accommodate e-commerce data. These two requirements are lacking in the ontological systems discussed in this thesis.

2.4 Background on Ontologies

An interesting perspective in resolving differences among heterogeneous data sources is the use of ontologies. The use of an ontology for intelligent information representation and integration is popular in the AI research community [HH00a, HH00b, CJO02]. An ontology can be viewed as a set of concepts clearly specified in a given fashion to create a common vocabulary for exchanging information in a particular domain [Gru93]. An ontology can also be seen as a framework that is used to describe a set of concepts – objects, and their properties, and the relationships among them. These concepts depend on the particular application domain, which may be the working model for entities and interactions in that particular domain of knowledge. In designing an ontology, the aim of the designer is to provide common terminologies and their constraints that will be agreed upon across the participating systems. Clearly, specifying such concepts that could be used to share and reuse knowledge across applications is crucial to the successful design of an ontology.

Özsu and Valduriez [OV99] observe that an important aspect of semantic heterogeneity is naming conflicts, which may be synonyms or homonyms. Homonyms can simply be resolved by prefixing the terms with the local schema or model names, but synonyms cannot be resolved in a similar straightforward manner. In an ontology, entities are clearly specified in a given fashion to create a common vocabulary for exchanging information in that particular domain. Therefore, using a common ontology in a given domain will naturally resolve all naming conflicts since all the information sources agree to use a common vocabulary [CJO02, OV99]. In an ontology, there is a vocabulary of terms and the meaning of these terms are specified, thus leading to some form of interoperability, reuse and sharing. Interoperability among heterogeneous information systems demands that the meaning of the information that is to be interchanged be understood across the system. Providing this understanding is a major area of application of ontologies, since the people involved have to commit themselves to the ontology. Gruber [Gru93] notes that users of a common ontology commit to the ontology if they agree to use the common vocabulary in a coherent and consistent manner. The agreement by the users implies that they understand the common vocabulary. In that case, all the external actions of the users are consistent with the ontology.

According to Uschold and Grüninger [Usc96] ontologies are useful in communication, as they provide a unified framework that reduces conceptual and terminological differences within an organization, thus enhancing shared understanding and communication. The adoption of a common ontology ensures that the participating systems use shared terminology for all entities and their relations in a specific domain. Interoperability among distributed systems has been a major issue to the research community and most ontological systems address the issue of interoperability [Usc96, WVV+01]. Finally, a shared understanding of the entities, and the relations in a particular application domain could further enhance the specification of software systems. Ontologies can be useful in identifying the requirements of the system and understanding the relationships that exist. Since ontologies describe the terms in a particular domain, they could be used to provide a declarative specification for a software system. Ontologies can also be used as a global query model or for the verification of an integration description. In this case, ontologies can serve as a foundation to check the

design against the specification manually. One of the benefits derivable from the use of ontologies in software specification is their reusability. The concept of reusability encourages software modules to be imported and exported among different software systems.

2.4.1 Approaches

Many approaches have been adopted in the past to design ontologies for data integration. Wache *et al.* [WVV+01] identify three general approaches to ontology-based integration. These approaches are:

- Single ontology approaches: In this approach, all information sources relate to a shared vocabulary called a global ontology. This is a straightforward approach that provides a shared vocabulary for the specification of the semantics of the entities in that ontology. Information sources participating in the common ontology are expected to have nearly the same view of a domain. Gruber [Gru93] argues that it could be difficult to find an acceptable minimal ontology especially when information sources have different views on a domain. A major disadvantage of this approach is that changes in the information sources affect single ontology approaches, since such changes may lead to changes in the global ontology and in the relationships between the other information sources.
- Multiple Ontologies: The multiple ontologies approach was designed to overcome the shortcomings of the single ontology approach. In this approach, each information source has its own ontology that describes it. Ontologies are not shared and could be developed independently hence this approach supports heterogeneous views. Unfortunately, the fact that ontologies are not shared makes it rather difficult to compare the different ontologies. One of the ways to resolve this problem is to define a translation that provides inter-ontology mappings between the ontologies. However, the design of such a translation system is non-trivial.
- Hybrid Approaches: This approach is a combination of the single and multiple ontology approaches to produce a system that addresses both their shortcomings. Here, every information source is described by its own ontology, and a shared

global vocabulary provides a means of comparing the individual ontologies. In other words, every source ontology is a subset of the global ontology. The global ontology provides a common view through which the various local ontologies can be queried. This approach supports heterogeneous views, and permits new source ontologies to be added and changes in existing ones to be made. The hybrid approach has the potential to provide support for the integration of Web data sources. However, a major drawback of this approach is that it does not easily support the reuse of existing ontologies, since there must be a global vocabulary that all source ontologies refer to.

2.4.2 Methodologies

According to Uschold and Grüninger [Usc96], the design of an ontology should begin with the purpose and scope of the ontology, hence providing a description for the capabilities of the ontology. The second step is to build the ontology, which requires a means to capture, code and integrate existing ontologies. The identification of purpose and scope of the ontology to be constructed should provide a solid background for building the ontology. In the third step, evaluation of the ontology is considered. To make a proper evaluation of the ontology, there is need to also view critically the purpose of the ontology under construction as earlier stated. The fourth step is documentation. Skuce [Sku95] observes the importance of documentation of existing knowledge bases and ontologies and suggests that important assumptions should be documented. Such documentation is useful since ontologies link the executable systems and the real world they model. Finally, there has to be an initial guideline for designing ontologies. A clear methodology for developing ontologies should also include guidelines for ease of use. In particular, the ontological engineer needs such guidelines to make the appropriate choices at the various levels of development. The choice of a top-down or a bottom-up approach to ontological development for a particular application is a common example.

The methodology adopted in [GF95] agrees with [Usc96] in almost all aspects. In this methodology, the first step is to raise questions which the ontology to be developed must answer, thus defining an ontology's requirements using existing problems in that problem domain. After this stage, a definition of the concepts (i.e. objects, properties of

objects, and their relations) is given. Next, a first order logic specification of the definitions and constraints of the terminology is provided and implemented. The first order logics are axioms in the ontology, which specify clearly the terminology and their constraints without ambiguities. Lastly, an evaluation of the ontology is given with respect to the questions raised in the first stage. The interest of this research is to provide a formal framework for the design of a common ontology.

2.4.3 Representations of Ontologies

In information integration, the semantics of the different sources can be described explicitly by the use of ontologies. The contents of such information sources are made explicit and could also be represented with metadata. This involves clearly describing the inherent meaning of table and data field names [PM01, Gru93]. Most ontological systems use variants of description logics to represent ontologies. Some other systems use extensions of description logics, which include rule bases and classical frame-based representation languages. Calvanese *et al.* [CDL01a, CDL01b] use description logic with n-ary relations (DLR) for information integration in data warehouse quality (DWQ). DLR is a declarative (logic-based) approach that models data at the sources. It proposes a suitable language to construct a unified representation and makes reference to such a representation when querying the global database information system. In description logics, complex hierarchical structures are modeled using a set of logics that are based on a well-founded logical framework. However, the integration of description logics with rule-based reasoning promises to be more powerful in representing the expressive power of the terms in language [WVV+01].

Harmelen and Fensel [HF99] suggest two broad strategies for representing the semantics of information sources. First, a declarative approach (semantic markup languages) that enrich information sources with annotations that convey their semantics in a way that machines can understand. The second is a procedural approach that involves the use of programs (wrappers, filters, and extraction programs) that procedurally extract the semantics from the information sources. A more detailed survey of languages used in ontology systems is provided in [CG00].

2.4.4 Problems with Ontologies

One notable problem with the use of ontologies is the issue of a methodological framework for building ontologies. It was noted in [Usc96] that there are no standard methodologies for building ontologies and not much can be found in the literature in this area. At the moment there exist a substantial number of differences between ontologies that were constructed to address similar purposes. One obvious reason for these differences is that there are no standard methodologies for constructing ontologies. Jones *et al.* [JBV98] conclude that current practices in the development of ontologies depend on the skills of the ontology designers rather than on a clearly specified engineering process. Nonetheless, a methodology that is based on sound engineering process and experience from existing projects will leverage the potentials of ontologies [JBV98].

Wache *et al.* [WVV+01] review existing solutions to information integration, especially the use of ontologies in these systems and conclude that the design of a successful ontology must be driven by the purpose of the ontology. We need a framework for the design and evaluation of ontologies. Such a framework will provide a basis for a clear evaluation of the different proposals for an ontology.

Another problem area with the development of an ontology for integration purposes is how to resolve mappings between the local and global ontologies [CDL01a]. Klein [Kle01] examines different types of mismatches that can occur between ontologies and hence establishes a framework that could be used for comparison and solving this problems. In particular, it was observed that ontological systems like SHOE [HH00a, HH00b] could not map all elements. For example, functions might be needed to map meters to feet [Kle01]. The above shortcomings suggest the necessity to investigate mappings on both a theoretical and empirical basis since most approaches still use *ad hoc* or arbitrary mappings especially for the connection of different ontologies [WVV+01]. Current efforts in developing ontologies are carried out in an *ad hoc* manner that lacks any meaningful theoretical basis. This research specifies a common-term vocabulary, and provides a theoretical framework for its evolution and composition.

2.5 Semistructured Data and the XML Standard

Some of the data used in e-commerce activities reside in legacy systems, largely represented with rigid schema-based relational databases, while others are found on the Web, where the data has no rigid structure. Although, traditional data models (e.g. relational, object-oriented, object relational, etc.) are good candidates to represent data from legacy systems, such models are not good for representing Web data because of the dynamic nature of data on the Web. The structure of most data on the Web evolve with time. In addition, the large volume of data on the Web makes it difficult to represent data that supports Web-centric applications with rigid schemas. Abiteboul *et al.* [ABS00] observe that the Web is by orders of magnitude the largest database ever created. An efficient integration methodology must guarantee that an integration system allows a non-redundant, unified representation of all data managed by the participating organizations. The sections that follow examine flexible data models, especially, semistructured data models and XML.

2.5.1 Semistructured Data

Data without any element of structure is unstructured, (e.g., text documents, text files). Consider for example, a scenario where we manually (or automatically) generate text document reports on our research activities. This kind of document gives us free hand to add, delete or even omit information without violating any underlying structure because none exists. Such a document that lacks structure and places little or no restrictions on its designers is referred to as unstructured. Because of the seemingly unconstrained nature of unstructured data, it becomes increasingly difficult to support expressive queries over such data. The document designer lacks appropriate tools to model the semantics of the document's textual content. Over the years, there has been collaboration between research and industry to fashion tools to solve some of the problems inherent in dealing with unstructured data [BDHS96, BDFS97, Bun97, PGW95].

Data with some elements of structure but different from conventional structured data models (e.g., relational, object-oriented) is called semistructured data. Semistructured data lies between unstructured collections of textual documents and fully

structured data. That is, they are data that are not fully unstructured, but contain some structure that may not be explicitly stated, may change without notice or may be too complex to describe using traditional means. Semistructured data is referred to as "self-describing" because, although it has no a priori schema, schema information is contained within the data [Bun97] (i.e. there is no separate description of the type or structure of the data). For example, the semistructured data expression {name: "Chima", phone: "x-8694"} contains both the data and the schema information. The value of any data is associated with a corresponding label, which may convey the semantics of the data to the users. The obvious disadvantage of self-describing data is that it wastes space if not properly stored. However, space is traded-off for ease of data exchange, which is an important aspect of e-commerce transactions. Models for semistructured data have been based on labelled digraphs with nodes as objects that have identifiers [Via01, Bun97]. Complex objects are linked with other objects by arcs while atomic objects are associated with data values. There is no restriction on the set of arcs that emanate from a given node in a graph, or on the types of values of attributes [FLM98].

A semistructured schema is descriptive rather than prescriptive (the current state of the data is described, but violations of the schema are still tolerated). Also, semistructured data is commonly associated with relatively large schemas (with respect to the size of the data) that are constantly evolving. The aim of semistructured data is that schemas be represented in a consistent way, thus providing for flexible and powerful representation of data in a unified manner. The flexibility of semistructured data aids browsing and may be useful in dealing with structured data especially when structured data is viewed as being unstructured. Importantly, the flexibility of semistructured data makes it amenable to data exchange between disparate data sources. Semistructured data models provide the necessary framework for managing large volumes of data including XML data on the Web. Semistructured data, however, has no a priori schema, a price to be paid for flexibility.

No doubt, schemas are very helpful because they describe the data and support the querying (including optimization) of data. These beneficial features have made it necessary to recover schema information from semistructured data. If we represent semistructured data as edge-labelled (where information resides in the edges) graph

[BDHS96], a schema for semistructured data contains the sequences of labels along the paths that can be found in the edge-labelled graph.

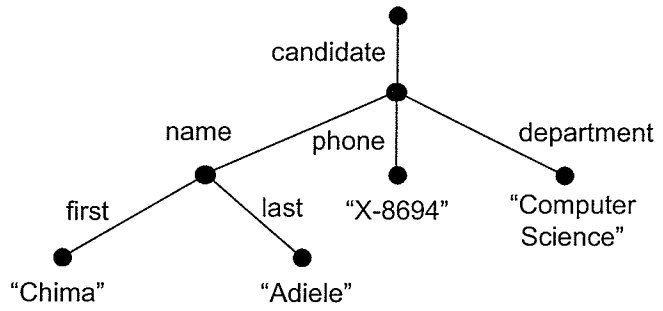

Figure 2.3: Graphical Representation of a Semistructured Data

Figure 2.3 is a semistructured data that represents the information in a student record. The values may be atomic as in [phone: "x-8694"] or complex as in [name: [first: "Chima", last: "Adiele"]]. If we denote the universe of all constants as (**Un** = Int Y; String Y; Bool Y ...) then every edge in the graph must contain a value in **Un** [BDFS97]. Notice that information only resides at the labels and labels must come from the universal set of all constants. Here, a node that represents the objects is connected by edges to the values. The above representation naturally extends relational or object oriented schemas to support semistructured data. A good feature of semistructured data that makes it amenable to data exchange is its ability to accommodate variations in structure, which means that we are not constrained to make the structures the same.

2.5.2 Variants of Semistructured Data

The use of semistructured data formats has been in existence for some time now for the interchange of data between applications. There are several types of semistructured data models. The first semistructured data model is the object exchange model (OEM) introduced as a means of integrating heterogeneous data sources [GMW99, Via01], although the use of the OEM model for exchange across organizational boundaries that are heterogeneous is more recent. The Lore [AQM+97, MAG+97] and UnQL [BDS95,

BDHS96] models followed after the development of OEM. In this section, we discuss OEM and the biological database system, AceDB (a Caenorhabditis elegans database) [TD92, TS99].

Object Exchange Model (OEM)

OEM [PGW95] was introduced within the TSIMMIS project as a platform for integrating heterogeneous sources. OEM retains the simplicity of relational models while allowing the flexibility of an object-oriented model. Objects in OEM have a very simple nature, yet they are powerful enough to facilitate integration because they offer a highly flexible data structure capable of capturing and representing most data types. A key feature of OEM is that it is self-describing, which eliminates the need for a regular structure or a predefined schema. An OEM object is a 4-tuple (label, oid, type, value) where

- label is a character string describing what the object represents.
- oid is a unique identifier for the object or Λ (for null).
- type is the data type of the object's value. Each type can either be atomic (integer, string, etc.) or complex (set, list).
- value is a variable-length value for the object.

Essentially, OEM data forms a graph similar to the edge-labelled graphs we described earlier, but the labels are attached to the nodes rather than the edges (though there are also some variants of OEM that attach labels to the edges instead of the nodes). OEM differs from other object oriented data models in several ways. In particular, OEM is an information exchange model that does not specify how objects are stored at the source; rather it specifies how objects are received at a client [PGW95]. Again, OEM is much simpler than OO models because it supports only object nesting and object identity, and does not support other features typically supported by OO models such as classes, methods, and inheritance. Finally, OEM uses labels in place of schemas.

AceDB

AceDB was one of the earliest genetic databases for a specific organism, although it is actually quite general. AceDB allows flexible representation of data similar to edge-

labelled trees. However, AceDB has a schema definition language that limits the type and number of edges stored in a database. AceDB only allows schemas that impose weak constraints on the database. Therefore, a class of AceDB schemas can be represented as a labelled tree with non-terminal[19] edges. The non-terminal edges are labelled by attribute names or base types and leaves are labelled by base types or by other class names. In AceDB, the query language allows selections of objects and pointer traversals but it cannot perform general projections or joins [BDHS96]. Biologists like AceDB for the following reasons [TD92]:

- Unlike the traditional "fixed schema" database models, the AceDB model fits the activities of biologists because of its support for the rapid evolution of experimental data techniques that requires constant adjustment of the schema.
- It accommodates missing data.
- It has a good user interface and some browsing tools.

2.5.3 The XML Standard

e**X**tensible **M**arkup **L**anguage (XML) [W3C98] is a set of technology standards used in defining, manipulating, displaying and managing data. XML was approved as a World Wide Web Consortium (W3C) standard in February 1998, to enhance the representation of data and interchange of structured documents over the Web. XML is widely used as a platform for interchange of data in Web-centric applications. XML documents exist on the Web and nearly every data management application today can export XML data [LSAF05]. The interest in integrating such data exported across applications and administrative boundaries is very high, thus making the integration of XML data across networks an exciting area of research.

A document type definition (DTD) [YA01] is a set of grammar rules for XML that defines the structure of an XML document. XML vocabularies are specified by a series of DTDs, which provide a means of parsing XML documents. An XML document is considered valid if its syntax satisfies a DTD grammar. A well-formed document in XML

[19] A non-terminal edge is an edge that has no child (this concept is described in Chapter 4).

has only one entry point (the root). XML documents are organized hierarchically beginning with the root.

The DTD can be seen as metadata for the corresponding XML document. The DTD can be contained within an XML document, which is referred to as internal subset. An external subset is when a DTD is provided outside the XML document in a separate file. The external approach is the default standard. DTDs can be shared by different systems to provide agreement on how XML documents should be structured [Sta01]. DTDs and XML documents provide a means for data exchange, especially in a B2B environment where multiple partners can exchange documents with each other based on a common set of DTDs.

Similar to DTDs, XML schemas [ABS00] can be used to specify the structure of a particular class of documents that must conform to XML syntax. The user is required to declare attributes and elements using an XML schema. Yee and Apte [YA01] identify the benefits of using XML schemas, which include but are not limited to the following:

- Extensibility. XML schemas allow the construction of a common schema document from multiple schemas using the fundamental object-oriented capabilities of aggregation and inheritance from base schemas. Such aggregation enhances reuse of schemas and reduces the probability of data management errors.
- Robust User Defined Types. XML schemas allow for robust user-defined types, while also providing base data types for numeric, date, time, and currency types.
- Access and Reference Mechanism. XML-based schemas can be traversed and accessed using mechanisms such as the Document Object Model (DOM)[20] [W3C04], thus paving the way to manage the schemas at a global enterprise level.
- Dynamism. XML schemas can be changed dynamically, a function that is significant to a B2B integration environment.

Furthermore, details of the presentation of the XML document can be specified using the eXtensible Stylesheet Language (XSL) [W3C99].

[20] DOM is a platform- and language-independent interface that allows dynamic access to programs and scripts, and updates the content, structure and style of documents.

2.5.4 Browsing Semistructured Data

The goal of a browsing mechanism is to provide a tool that is platform-independent for displaying and exploring the data that are returned as a result of a given query. It is important to offer users browsing mechanisms that allow them to navigate through the query answer space because of the nested nature of some objects in that space. In TSIMMIS [CGH+94], the heterogeneous information browser (HIB) provides a graphical user interface for submitting queries and exploring results. HIB gives the user a platform to query the TSIMMIS system and navigate through the object structure of answers returned from queries[21]. In Lore, DataGuides [GMW99] structures are used to replace the traditional database schemas. DataGuides facilitate the browsing of the structure of a Lore database and aid in formulating meaningful queries since they provide the needed structures.

Browsing unstructured data is easier because of its flexible schema structure that does not restrict the user. For instance, it is difficult in a relational schema to write a generic algebra expression to express a query that will search the entire database to determine which tables contain a particular term. On the contrary, a user could pose such a query on a semistructured data model because it is self-describing and easy to navigate through unknown objects. This is an example of a generic query that could be posed by a user who has no idea about a particular source as might occur on the Web. The user learns the meaning of each component as the labels are retrieved, and then formulates a more directed query. Therefore, for the purposes of browsing it is more convenient to ignore schemas even when they exist for the data.

2.5.5 Querying Semistructured Data

For a user to access semistructured data, the user has to query the system to request the information needed. Queries are posed based on the underlying syntax of a given language. Details of languages with varying capabilities for querying semistructured data models have been extensively discussed in [Abi97, ASB00, BC00, BDHS96, PGW95].

---

[21] See [PGW95] for details on the HIB

Abiteboul [Abi97] notes that most query languages for semistructured data are primarily focused on how to select and transform information from large data sources. These query languages take advantage of successive representation of edge-labels to reach arbitrary depths in the edge-labelled graph. Queries in both standard databases and semistructured data are conceptually the same, except for the fact that in standard databases, query languages exploit the relational nature of the schema (tables), while in semistructured data, query languages exploit a path expression[22]. Therefore, the standard SQL query format applicable to standard relational database systems could be extended to leverage path expressions in a semistructured data model.

The path knowledge in a semistructured data model is necessary because it enhances the quality of queries posed. It is easier to bind the path expressions and traverse the labelled graph if the path of a query is given explicitly. Semistructured query languages have the potential to reach arbitrary depths in the labelled graphs because of their ability to exploit the notion of path expressions. On the contrary, without path knowledge, the user learns the meaning of each component only as the labels are retrieved and then must formulate a more structured query. Unfortunately, this approach of posing a query is expensive to execute in terms of the time it will take.

2.5.6 The Future of XML

Widom [Wid99] reviews the research opportunities XML brings to the field of data management. In particular, Widom discusses how the semistructured data model in Lore [PGW95] could be used to support XML. Goldman *et al.* [GMW99] describe the essential changes made in migrating Lore's data model and query language to support XML. However, performance enhancement issues in the Lore-XML system with respect to indexing and data layout were not considered. Levy [Lev99] focuses on the role of XML in data integration and measures of performance. Levy suggests that measures of performance should be viewed from the number of XML sources/files and degree of

[22] A path expressions is a sequence of labels of the form $l_1, l_2, \ldots, l_n$. A simple path expression permits the access of a set of objects through a sequence of labels starting from a named object in a graph.

irregularity in the sources instead of the traditional means of measuring performance by the scale up in the size of the data and the size of the queries (query complexity).

Although, XML allows easy development of applications that exchange data over the Internet, it does not provide a complete solution for the problem of integrating e-commerce data. XML only simplifies the task of syntactically understanding the data and does nothing towards semantic interpretation of the data. There are still more challenges in using XML to provide interoperability for e-commerce data. These challenges include, but are not limited to the following [BF01, GMW99, SR01]:

- Using techniques from database systems and information retrieval to access XML data will be faced with the challenge of handling a mixture of traditional data elements with free text in XML. Traditional database tools will need to be reengineered to effectively handle XML data, especially on the Web.
- There is a need for a set of standardized domain specific tags and schemas since the names and meanings of XML tags are arbitrary. In addition, there may be different interpretations of the same DTD.
- XML does not provide semantic description for DTDs, elements and attributes. There has to be standards that provide shared vocabularies to describe the semantics of DTDs, elements and attributes.
- There is need for administrative tools that map heterogeneous attributes and schemas and resolve object identity and conflicting values.
- There is need to scale up everything about XML to the size of the Web. The growth rate of the Web is very high and any Web-enabled tool should be able to scale at the rate of the Web to retain its value.

2.6 Formal Techniques and Algebraic Signatures

The *ad hoc* informal methods of software development used by most existing integration systems are becoming increasingly inadequate to address the twin issue of software complexity on one hand and the high demand for reliable software on the other hand. In data integration, *ad hoc* integration approaches based on heuristics generate integration results that are mathematically unsatisfying [RB01]. Such mathematically unsatisfying

results cannot be tolerated in an e-commerce environment where accuracy is of the essence. Harry [Har96] notes that the use of software engineering and structured design methods is not enough to handle complex software artefacts, such as data integration systems. For example, the use of only software engineering and structured design methods are grossly inadequate for complex software, thus necessitating the use of a rigorous methodology for specifying and designing software.

In this subsection, a background survey on the use of mathematical notations to unambiguously define components of integration systems is provided. Mathematical notations offer distinct advantages that allow machine interpretation and manipulation because of their precise meaning. Mathematical notations are rigidly defined to allow incremental definition of new rules from specified ones in provable ways allowing such things as automatic simplification and proof analysis. The driving force behind formal methods is the elimination of ambiguities, vagueness, incompleteness, and contradictions inherent in informal systems' definitions, thus paving the way for a concise, unambiguous, and accurate formal systems' definition [Sch94].

Formal methods are techniques for describing system properties based on sound mathematical principles, in particular on discrete mathematics and logic. Such techniques provide a mathematical framework to systematically specify, develop, and verify systems, rather than adopting *ad hoc* approaches to systems specification [Win90]. The sound mathematical principles in formal methods provide a platform to precisely define notions such as consistency, correctness, and completeness of specification and the resulting implementation. Plat *et al.* [PKT92] state that formal methods are formal systems that provide tools to inspect the satisfiability of specifications, prove the correctness of a system's implementation, and prove system's properties without necessarily executing the system to determine its behaviour.

A formal method is required to specify a system in a complete, correct, and verifiable manner [Ehi97]. A specification is complete if it contains all essential characteristics and properties. Correctness ensures the specification accurately captures the designed system's properties in an unambiguous and consistent manner. Consistency in specification leads to the absence of contradictions. Verifiability shows that the

specification correctly specifies the system's properties. Such formal techniques are of great potential value in data integration systems.

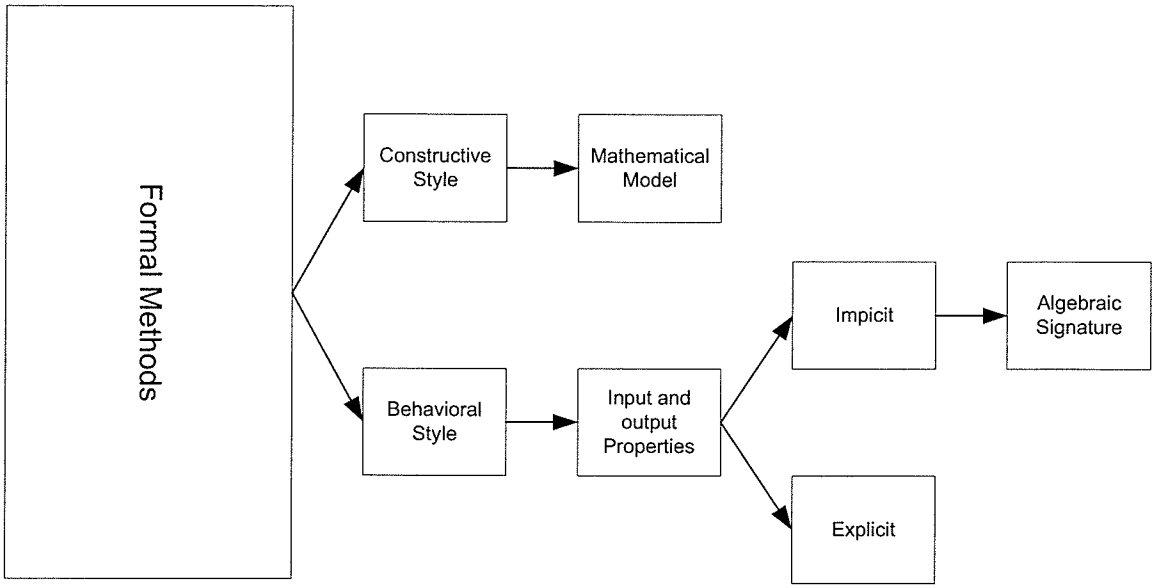

Figure 2.4: Classification of Formal Methods

Figure 2.4 shows a classification of formal methods. According to Harry, formal methods can be classified into two broad categories, namely constructive and behavioural styles. Using the constructive style, a mathematical model of the system is constructed, while using the behavioural style, the system is expressed in terms of its input and output properties. A formal specification may either be implicit or explicit. Implicit specifications focus on *what* a system's operations are required to accomplish without necessarily considering how the task is to be carried out. Explicit specifications focus on *how* a system's operations will be carried out. An algebraic signature is an implicit property-based formal framework that implicitly expresses operations by their algebraic equivalences. Implicit specifications are usually expressed in the form of abstract pre- and post-conditions, such that the function arguments satisfy the pre-conditions and the result of the function satisfies the post conditions [AP98]. Implicit specifications are more abstract in nature and normally shorter in description than explicit specifications. Implicitly specified terms of an algebra are representation independent and thus, offer the system's developer the freedom to choose appropriate implementation methodology

[BG77, Hoa77, Wir90]. Detailed explanation of algebraic signatures is provided in Chapter 4.

Algebraic signatures have been used in the past to describe database applications [GZC89, NLH+98, FWM97]. Algebraic signatures provide mathematical formalisms to the systems they describe. Güting *et al.* [GZC89] extended relational algebra to describe an algebra for structured office documents. Their algebra provides a formal model and language for the filing, retrieval, and construction of documents, particularly in office information systems. Ng *et al.* [NLH+98] extend data warehouse concepts to construct Web warehouses using Web algebra. This algebra consists of a Web data model and a set of operators that captures the topological structure of the Web within the schema. The Web algebra provides a formal foundation for data representation and manipulation for the Web warehouse. Fiebig *et al.* [FWM97] develop a relational algebra for the Web (RAW), an extension of relational algebra to evaluate queries against the Web. RAW introduces some Web-specific data types (HTTP-Address, HTTP-Address-Path, and HTML-Document-Path), and a set of operators including the map operator that maps input symbols to output symbols to deal with URLs, HTML documents and path expressions. This work is related to this research because both relational algebra and Web algebra are examples of algebraic signatures used to solve database problems, but they focus on different subjects. Fiebig *et al.* focus on query processing against the Web, while Ng *et al.* focus on developing a data model for the Web.

Wallace *et al.* [WCK91] observe that formal methods are useful at various levels of abstraction. Each level of abstraction provides different levels of assurance for corresponding software developed. According to Lustman [Lus94], formal methods are used in three ways: 1) to express the operations in a specification; 2) to verify that pre- and post-conditions of a specification are in agreement; and 3) to transform the input into some output. This research is limited to the use of formal methods in the first two ways.

Chapter 3

Characterization of E-Commerce Data Sets

This chapter is organized into three major sections. Section 3.1 presents a characterization of data into three orthogonal dimensions, and hence describes e-commerce data set [Adi05]. This characterization is necessary to determine the nature, constituents and quality and control requirements of e-commerce data to facilitate its integration. Section 3.2 examines the formal description and analysis of the constituents of e-commerce data, thus providing a framework to identify the building blocks of an e-commerce application [AE03c]. Section 3.3 presents a four-phase formal model of an e-commerce application and also provides a formal specification of the four-phase model [AE04e, AE04g]. This part is necessary to show the data management requirements of an e-commerce application.

3.1 E-Commerce Data Set

A piece of data is a concept that represents facts with specific meaning, which may be formatted in a special way. These facts may vary with respect to time because data exist in an evolving world. Different kinds of data exist in different environments with varying demands for storage and representation [Bra96]. The different kinds of data are described for clarity and ease of understanding. Figure 3.1 shows the three broad categories of data;

namely operational, historical, and archival data, which are explained below. It is not out of place to observe data as a life cycle because a particular piece of data may transit from one form to another, and can also be used in different ways, such as operation, analysis, forecasting, etc.

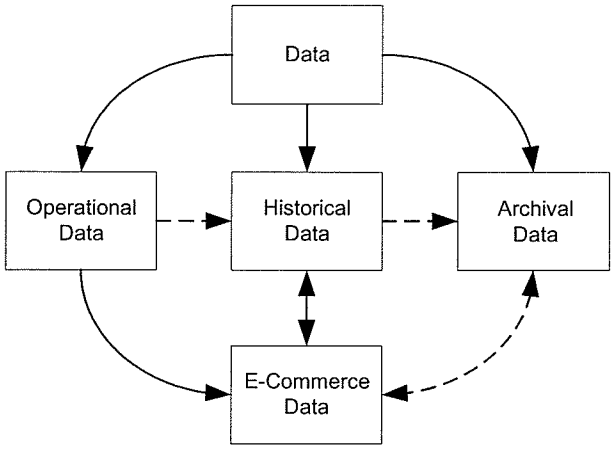

Figure 3.1: Classification of Data

- Operational Data. Operational data is used directly to support the daily operations of an organization. Operational data contains detailed data that are kept current to sustain online transaction processing. For example, the quantity of goods ordered by a merchant from a producer is operational data since it influences the production operations of the producer, and also certain operational decisions of the merchant.
- Historical Data. Historical data are naturally suitable to assess the behaviour of events (e. g. sales, stocks, etc.) and are often used for decision support processing. They are used to evaluate trends, make projections and provide other useful information that will aid decision-making. Historical data aggregate the detailed operational data with many levels of summarization [Bra96]. A typical example of a repository of historical data is a data warehouse. A data warehouse contains historical data that can be easily accessed and manipulated for decision support. As a result, historical data is mainly stored online to make it easily accessible for analytical processing. Historical data is also used in the e-commerce domain, especially in the area of data mining to study the behaviour of customers.

An example of historical data is when the value of a company's monthly sales may vary with respect to time and at the end of every month, the value for the preceding month is saved to a history file with the appropriate date. These kinds of data can be used to determine market trends and form a basis for managerial decision-making.

- Archival Data. Data that is no longer in operational use can be archived with a time function, and is then termed archived data. Archived data is usually not part of the operational data and are, therefore, stored offline or in a medium different from operational data. For example, people buy and/or sell cars at auction on a regular basis. The details of the most recent owner of a car are necessary for insurance and identification purposes and may be stored online. The information on former owner(s) may be stored offline in an archive file, possibly for police records. Archival data is also a form of historical data but one which is expected to be accessed less frequently.

3.1.1 Taxonomy of Data

E-commerce data is the set of data that represents information about activities or entities in the domain of e-commerce. E-commerce data is a subset of data in the domain of business data. Quix *et al.* [QSJ02] define business data as information about a business entity or activity that is stored on some media. Business entities include, but are not limited to companies, products, employees playing a certain role, contracts, money, rules and regulations. Business data is used, among other things, in operational systems that support business processes. A business process is an organizational activity that contributes to the overall business goals of the organization. Figure 3.2 classifies data along three orthogonal[23] dimensions, which include:

a) Temporal dimension. The temporal dimension captures the temporal properties of data. In this case, the temporal component of data varies from operational, which means that the data is current and in operational use; to archival, which is data that is neither current nor in operational use. The temporal property of data determines

[23] These dimensions are not in the continuous space and orthogonality in this context is different from the perspective geometry.

the mode of storage. Mode of storage can be online, offline (possibly in a primary storage), or archive (possibly in a secondary storage).

b) Representation dimension. The representation dimension shows how data is represented. Data can be represented using rigid schemas or a mixture of data models. For example, in legacy systems, e-commerce data is represented with structured schemas (relational, object-oriented, etc); while on the Web, e-commerce data is represented with different structures, which may be structured, semistructured (XML documents), or unstructured (text documents).

c) Usage dimension. The usage dimension shows the domain of use of the data. The use of data cuts across different domains and purposes, including statistical purposes (e.g., census data), geographical, medical, business information, etc. Data in any given domain may possess certain characteristics that uniquely differentiate it from data in other domains. The subsection that follows gives the characteristics of e-commerce data.

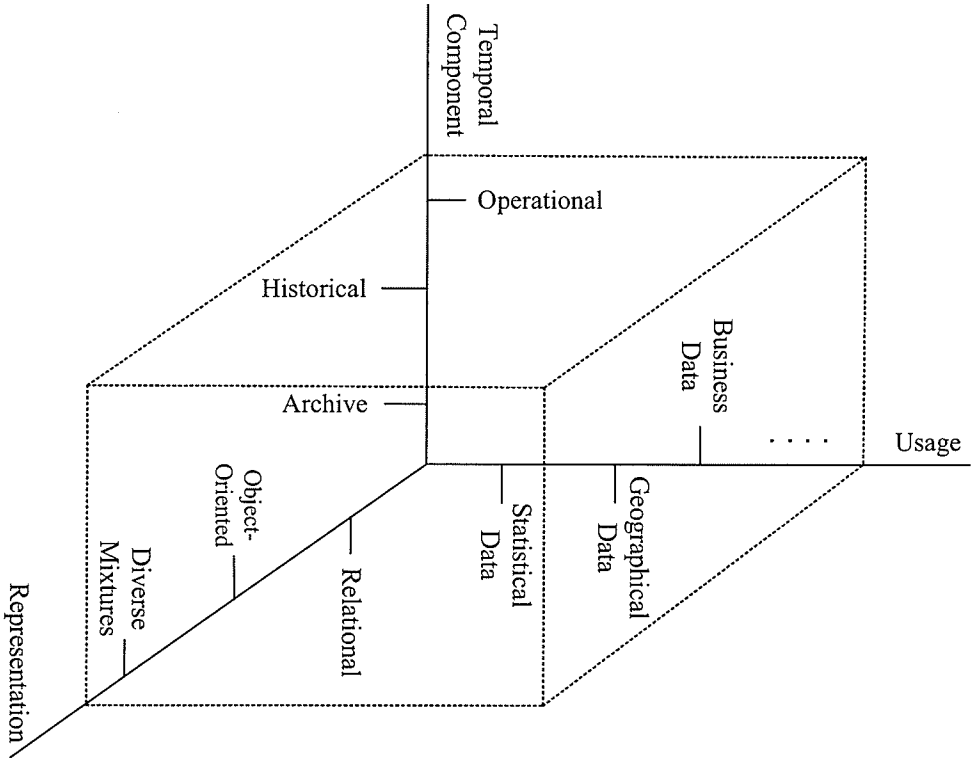

Figure 3.2: Fundamental Data Taxonomy

The taxonomy of Figure 3.2 represents data D as a triple $D_{\langle T, R, U\rangle}$ where T is the temporal component of the data set, R shows how the data is represented, and U gives its domain of use. In general, e-commerce data is the set of data in the domain of business data that are represented with a mixture of data models to capture data in both legacy systems and on the Web. E-commerce data could be in any part of the temporal dimension depending on the state of the data.

3.1.2 Nature of E-commerce Data Sets

The characteristics of e-commerce data that influence the design decisions of an integration system are:

- Heterogeneity. E-commerce data emanate from independent data sources that are inherently heterogeneous. These data sources may be structured, as in legacy systems that are mainly historical and archived data normally represented with relational data models; or semistructured, as in Internet-based systems that are mainly operational data presented as data on the Web. E-commerce data from the different environments need to interoperate to be fully useful.
- Size. E-commerce data is usually large in size to accommodate the large number of attributes that completely describes the data. These attributes describe the physical properties of data or how data is used. E-commerce also involves large volumes of transactions that depend on data from the e-commerce domain. For example, a typical catalogue may contain 100,000 stock keeping units (SKUs), and about 10,000 attributes [Jhi00]. It is difficult to represent such catalogues with relational databases, because relational databases are not flexible enough to manage so many attributes.
- Evolvability. E-commerce data is inherently dynamic because of the dynamic nature of e-commerce systems. Data sources and their contents evolve with time, as new sources may be added and some others may be removed. The high variability of e-commerce data makes it expensive and extremely difficult to represent with rigid schemas.

- Operational Use. E-commerce data is bi-directional as the application (middle-tier) receives data from the client (front-end) and also sends data from the database (back-end) to the client. Therefore, a suitable data model that efficiently represents data in the database and clearly presents data to the client in an understandable manner is necessary.
- Shareability. E-commerce data is inherently distributed over multiple sites and platforms because of performance considerations since data is shared by different applications. However, data distribution has its attendant difficulties, such as fragmentation (the division of a relation into sub-relations, called fragments) and the perpetuation of heterogeneous data, thus making it complex to integrate.

Characterizing e-commerce data sets aids the identification of data available to a business organization, and facilitates the creation of a common ontology. A common ontology elucidates the content and meaning of e-commerce data to support the information needs of an organization. Identifying and understanding the peculiar characteristics of the data in use in a given environment enhances the choice of a representation technique and integration mechanism as the meaning of data to be integrated is made explicit.

3.1.3 Typical Constituents of E-Commerce Data Sets

Figure 3.3 shows the sources, data and the different entities in an e-commerce system. This system could be a B2C, B2B or other type of e-commerce system. Every user has an identification embedded in a user profile that may serve as contact information in the system. User profile data may include name, organization, address, phone, user activities, etc. An allowable operation on such data could be the *Login-Method* to facilitate the identification and registration of a particular user. An e-commerce system keeps logs of user activities.

A company profile provides the contact data for companies, which may include name, identification, address, web page address, services, products identifiers and any other data that may be relevant. An allowable operation on such data could be the *Browse-Catalogue-Method.*

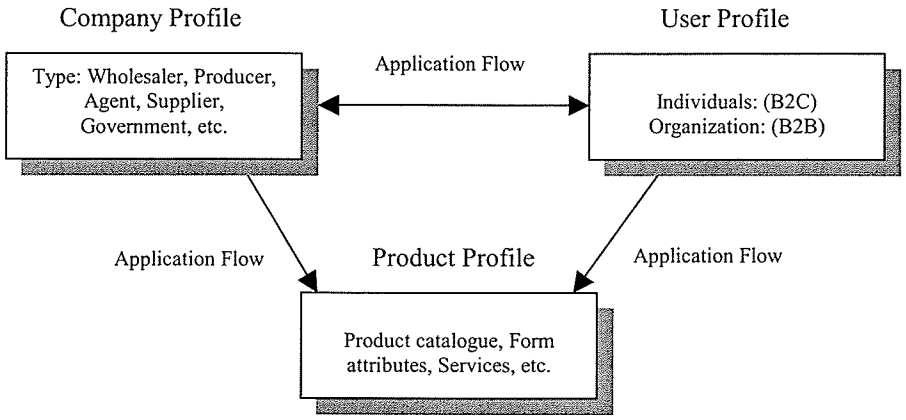

Figure 3.3: Data and Actors in an E-commerce Environment

Product profiles may include product catalogues and other descriptive information that a company would want consumers or business partners to see. An allowable operation on such data could be the *Search-Product-Method.* Different actors participating in an e-commerce application have different roles enabled by the associated data and methods. The sequence of roles of the various actors involved in a given application forms the application flow. Any meaningful e-commerce data integration should relate the data involved to application flow and the interactions between the various participating actors in an e-commerce application.

3.1.4 Control and Quality of E-Commerce Data

Organizations that own e-commerce data expect to have full control over the content of their data (update, add, or delete) since data is viewed as an invaluable asset. Thus, data integration techniques have to be re-engineered to accommodate data source autonomy. Autonomy of data sources implies that each data source can make choices with respect to its data model, naming concepts, universe of discourse, the structure and content of metadata, etc., regardless of the design choices of the other data sources. These independent choices of participating data sources naturally lead to heterogeneity.

E-commerce data must be of high quality to maintain the integrity of e-commerce applications, and hence attract and retain users' confidence. Data quality consists of data accuracy, data completeness, and data integrity [Bra96, WW96]. Data accuracy reflects the extent to which data represent the real world entities. Accuracy of e-commerce data is important since it contributes to the overall systems accuracy. Accuracy of e-commerce data is necessary at all times as data migrates from one form to another. Data completeness ensures that the entire set of data needed to meet the information requirements of businesses is available. Data integrity deals with the consistency and timeliness of data. The values obtained from data should not be contradictory or out of date. A data item d is a pair (v, t) where v is the value of the data item at time t, and $VAL[d, t]$ is a function that returns the value of any data item d at time t. The value of a particular data item accessible to different users at the same time should be the same. For example, if any two methods m_i and m_j perform a *read* operation on a data item d, at time t, from a source (say, DB), then $VAL[m_i(d), t] = VAL[m_j(d), t]$, otherwise ($d$, t) is said to be inconsistent. Also, it should be possible to trace (d, t) to a particular DB since the source of data contributes to its quality. If a company withdraws from business, then data from the company should be expunged from the system. The concept of timeliness shows the currency of the data. If $VAL[x_i, t_1]$ and $VAL[x_i, t_2]$ represent the values of x_i at times t_1 and t_2, respectively, $VAL[x_i, t_2]$ contains the more current value if and only if $t_2 > t_1$.

3.1.5 Components of E-Commerce Applications

An e-commerce application consists of three components that must cooperate to produce a successful e-commerce solution. These components can be represented in three dimensions, as shown in Figure 3.4. The data component represents all the data available to the various actors in an e-commerce activity.

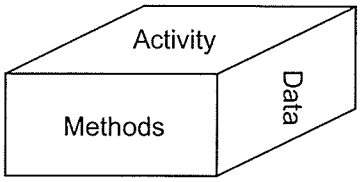

Figure 3.4: Components of E-Commerce Application

Data available may be from input or a by-product of the activities. The activity component represents e-commerce activities that invoke one or more methods to carry out a given task, while methods represent likely operations on a particular set of data. A typical activity could be the *Ordering-Activity* that facilitates the ordering of products. A task accomplished by an e-commerce activity is the result of execution of methods invoked by the activity.

3.2 Formal Description and Analysis

An e-commerce data set consists of a set of data objects that can be uniquely identified. An object is structurally composed of attributes, whose values define its state. An object identifier, *Oid*, uniquely identifies an object *O*. A label is a string of characters that uniquely describes the object. A data object *d* is an object whose attribute value is of a basic type (e.g. integer, real, string, e.tc.). The attribute value of a complex object may be a set of objects of the form $\{l_1: o_1,..., l_n: o_n\}$ (where l_i is the ith label that describes a corresponding object, o_i). A complex object may also consist of an arbitrary depth nesting of other objects called sub-objects. Operations that are possible on a particular set of data objects are represented as methods. A method *m* identifies and performs an action on object data. These operations can vary the object's state.

Let OBJECT be the basic type for objects, LABEL the basic type for labels, and OID be the basic type for object identifiers. To formally define an object, we give the signature of a function that maps an object to a label.

Object-Label: OBJECT $\rightarrow$ LABEL[24]

[24] A glossary of the notations used in this thesis is presented in Appendix A.

Definition 3.1 An *object O* is a triple (Ω, *Atr*, *M*) where:

1. Ω = (OID, *Object-Label*)

   $\forall O$: OBJECT • ($\exists_1 l$: LABEL • *Object-Label*(O) = l) $\wedge$

   ($\forall i, j$: OID | $i \neq j \Rightarrow O.i \neq O.j$)

2. $\forall a \in Atr$ • ($\exists i, j$: $\mathbb{N}_1$ | $i \neq j \Rightarrow a_i \neq a_j$)

3. $\forall m \in M$ • ($\exists i, j$: $\mathbb{N}_1$ | $i \neq j \Rightarrow m_i \neq m_j$)

Point 1 specifies that every object has a unique identifier and at most one corresponding label. Point 2 specifies the composition of the attributes that define the structure of the object. The attributes of an object are unique, which means no identical attributes can exist in an object. Point 3 specifies the set of allowable operations on the object that stipulate the behaviour of the object. The methods of an object are unique which means no two methods in the same object have the same name. ■

Let $m_i(d)$ denote the ith method on a data object d, and M the set of methods on data. Let $\prec_{pre}$ denote the precedes relation [Lam78], which is a binary relation that indicates the order in which methods are executed. The *precedes* relation is necessary because it captures the application flow[25] in an e-commerce application. Given two methods m_i and m_j, m_i *precedes* m_j (written $m_i \prec_{pre} m_j$) if the execution of m_i precedes m_j. Note that,

$$\left.\begin{array}{ll} m_i \prec_{pre} m_j \Rightarrow \neg (m_j \prec_{pre} m_i) & \text{(anti-reflexive)} \\ (\mathrm{m}_i \prec_{\mathrm{pre}} \mathrm{m}_j) \wedge (\mathrm{m}_j \prec_{\mathrm{pre}} \mathrm{m}_k) \Rightarrow (\mathrm{m}_i \prec_{\mathrm{pre}} \mathrm{m}_k) & \text{(transitivity)} \end{array}\right\} \quad (3.1)$$

Let $O_T \in$ {abort, commit} denote the termination operation for a method. This thesis defines atomicity of an operation in the spirit of [Ehi97, GR93][26]. The atomicity of a method stipulates that a method either aborts or commits and the read and write operations on a set of data preserve the domain of the data.

---

[25] An application flow shows the sequence of execution of a set of methods or activities on a set of data.

[26] The concept of atomicity [GR93] and the specification of transaction protocols are well documented in [Ehi97].

- *Commit* - A method commits if it terminates successfully and produces results. A committed method cannot be changed.

  $Commit(m_i) = \{\text{True} \mid \text{False}\}$

- *Abort* – A method aborts if it terminates unsuccessfully. The effects of the abnormal termination are removed.

  $Abort(m_i) = \{\text{True} \mid \text{False}\}$

Let DATA denote the attribute values of a data object. The definition of method follows.

Definition 3.2 A method m_i *on a data object* d *is a 5-tuple (*id, Op, P, O_T, R*) where:*

1. *id*: $\mathbb{P}_1\mathbb{N}$ uniquely identifies the method;
2. $Op = \{r(d_i), w(d_o)\}$;
3. $P = (d_i, d_o: \mathbb{P}\ \text{DATA})$;
4. $O_T \in \{Commit, Abort\}$
5. $R: \mathbb{P}\ \text{DATA} \mid Op \cup \{O_T\} = R \Leftrightarrow Commit(Op) = \text{True}$.

Point 1 states that *id* is an identifier that uniquely identifies a method. Point 2 states that *Op* is a set of read and write operations, where $r(d_i)$ is a read action performed by m_i when it reads some input data, d_i, from the set of data, and $w(d_o)$ is a write action performed by m_i when it writes some output data, d_o. Point 3 states that $\{d_i, d_o\}$ are input and output parameters, respectively. A method may not use an input or return an output, but generates results as side effects[27]. Point 4 states that a method either terminates successfully (commits) or unsuccessfully (aborts). Point 5 states that every successfully executed method produces results. The result generated by a method is a set of return values, which may be either an output data or side effects. ■

We formalize Point 5, and show that if a method, m_i precedes another method, m_j, then m_i, also commits before m_j.

$$Method \triangleq m_i \mid m_i \in M \bullet$$
$$\exists D, R: \mathbb{P}\text{DATA} \bullet m_i(D) = R \Leftrightarrow Commit(\text{m}_i) \wedge$$

[27] A method may not return a value as output but may trigger the execution of another method, which is referred to as a side effect.

$$(\forall m_i, m_j \in M \bullet \exists i, j: \mathbb{N} \mid i \neq j \bullet m_i \prec_{\text{pre}} m_j) \Rightarrow \quad (3.2)$$
$$Commit(m_i) \prec_{\text{pre}} Commit(m_j)) \wedge$$
$$\exists k: \mathbb{N} \mid j \neq k \bullet (\text{m}_j \prec_{\text{pre}} \text{m}_k) \Rightarrow (\text{m}_i \prec_{\text{pre}} \text{m}_k))$$

Let a denote an activity. The ith activity, a_i, invokes a set of methods, $M^n_{i=1}$, denoted as a_iM, to execute a particular task, such that each $m_j \in M$ operates on a given set of data D_j. The result of a committed activity a_i is the accomplished task T_i. T_i is the union of all the results R_j from the set of methods M executed by a_i. The execution of an activity a_i can cause the execution of another activity a_j. For example, a customer cannot successfully order a product without making full payment. Suppose, the task of activity a_i is to facilitate the ordering of products, and the task of activity a_j is to facilitate the payment of products ordered. To execute successfully, a_i triggers a_j, and a_i commits before a_j commits (written $a_i \prec_{pre} a_j$).

Definition 3.3 A *task* T_i is an instance of an activity which requires a set of methods M that operates on a set of data D to produce results R_j. The union of all R_j produced by M accomplishes the task of a given activity. Formally,

$$T_i = \sum_{j=1}^{n} R_j, \text{ such that } R_j \text{ is the set of results produced by } M^n_{i=1}. \quad (3.3)$$

■

Definition 3.4 An *activity* a_i is a 5-tuple (id, $M(D)$, O_T, T_i, $\prec_{pre}$) where:

1. id: $\mathbb{N}_1$ uniquely identifies the activity;
2. $M(D) \mid M = \bigcup_{j=1}^{n}(m_j) \wedge D$: $\mathbb{P}$ DATA;
3. $O_T \in \{Commit, Abort\} \bullet a_iM(D) \cup \{O_T\} = T_i \Leftrightarrow O_T = \{commit\}$;
4. T_i is the task accomplished by a_i; and
5. $\forall a_i \bullet (a_i \prec_{pre} T_i) \wedge (T_i \prec_{pre} O_T) \Rightarrow (a_i \prec_{pre} O_T)$

Point 2 states that $M(D)$ is a set of methods that depends on the set of data D when executed by a_i. Point 3 states that the execution of an activity accomplishes a task if and only if the activity commits. Point 4 states that T_i is the task accomplished. Point 5 states that the execution of an activity a_i precedes a given task T_i; and T_i is generated before the

activity commits. An activity commits if all the methods in the activity commit otherwise it aborts. ■

Let T denote a set of tasks and A, a set of activities. Formally,

$$\begin{aligned} &Activity \cong \{a_i \mid a_i \in A \bullet \\ &\quad (\forall m_j \in M \bullet (\exists D, R_j: \mathbb{P}\mathrm{DATA} \bullet m_j(D) = R_j) \Rightarrow \\ &\qquad (\exists T_i \in T \bullet T_i = \bigcup_{j=1}^{n} R_j \wedge Commit(a_i) \Rightarrow Commit(m_j)))\} \end{aligned} \tag{3.4}$$

An e-commerce activity is an activity that operates on a set of e-commerce data. A formal definition of e-commerce data follows.

Definition 3.5 An *e-commerce data set* (*ECD*) is a set of data objects D on which a method m may operate to generate a set of results R, where R represents information about an e-commerce activity. In addition, D satisfies the following conditions:

1. m is an allowable operation on D;
2. The state of D could be altered by m to ΔD;
3. The size of D could evolve from D_x to D_y, where x represents the number of attributes before operation by m, and y the number of attributes after operation by m, such that $0 \leq y \leq x \leq n \vee y > x$; and
4. $R \in \mathrm{dom}(ECD)$. ■

Recall, a method m_i, executed by e-commerce activity uses a set of e-commerce data as input and produces results when the method commits. These results may be a set of output data or side effects. The execution of a method m_i may trigger the execution of another method m_j that uses the output of m_i as input. Therefore, the two methods can be composed. The composition operator (°) is a binary operator that uses methods as operands. The definition of the composition operator follows:

$$\begin{aligned} &- \circ -: m \times m \rightarrow m \\ &\forall m_i, m_k \in M \bullet (\exists D: \mathbb{P}\mathrm{DATA} \bullet \\ &\quad m_i \circ m_j = m_i(m_j(D))) \end{aligned} \tag{3.5}$$

A set of methods $M = \{m_i \mid i: \mathrm{I}\}$ produce a set of results $R = \{R_j \mid j: \mathbb{N}\}$. I is an index since every $m_i \in M$ is unique (Definition 3.2). However, not all $R_j \in R$ are unique because two methods m_1 and m_2 can generate the same result. For example, m_1 could be a method

(using *ISBN* as its input data) to search for a book from an online catalogue. The method returns "True" if found or "False" otherwise. Similarly, m_2 could be a method (using *Customer-ID* as its input data) to check if a customer is a valued customer to qualify for some discount. The method returns "True" if found or "False" otherwise. Suppose m_1 operates on d_1 to return "True" and m_2 operates on d_2 to return "True", then, we have:

$$\{m_1(d_1), m_2(d_2)\} \rightarrow \{R_1, R_2\} = \{\text{True, True}\}$$

m_1 and m_2 are unique, but R_1 and R_2 are not necessarily unique.

To distinguish multiple instances of the various members of R, we define a family of methods and the graph of a family of methods.

Definition 3.6 A *family of methods FM* is a total function that maps unique methods, $m_i \in M$, into a set of results, $R_j \in R$. ■

$$FM: M \rightarrow R$$

The definition of the graph of a family of method follows.

Definition 3.7 The *graph of a family of methods* $G_R(FM) = \langle m_i, R_j \rangle$. Formally,

$$G_R(FM) \triangleq \{m_i, R_j \mid m_i \in M \wedge R_j \in R\}$$ ■

The various members of $G_R(FM) = \langle m_1, R_1 \rangle, \langle m_2, R_2 \rangle, \ldots, \langle m_n, R_n \rangle$ are distinct from one another, at least in their first component. This definition is important to trace the flow of execution of methods and their corresponding results, hence paving the way for us to show that the set of e-commerce data is closed (*ECD* is closed under $G_R(M)$).

Combining Definition 3.6 and the composition of methods (3.5) we can derive the required result:

$$\begin{aligned}
&\textit{E-Commerce Data} \triangleq \{ECD \mid ECD: \mathbb{P}\ \text{DATA} \bullet \\
&\quad (\forall m_1, m_2: \text{METHOD} \mid m_1, m_2 \in FM \bullet \\
&\qquad (\exists D, R_i, R_j: \mathbb{P}\text{DATA} \bullet (m_1(D) = R_i) \Leftrightarrow \\
&\qquad\quad D \in \text{dom}(ECD) \wedge (m_1, R_i) \in G_R(M)) \wedge \\
&\qquad\quad m_1(m_2(D)) = R_i \Leftrightarrow m_2(R_i) = R_j \wedge \\
&\qquad\qquad R_i \in \text{dom}(ECD) \Rightarrow R_j \in \text{dom}(ECD))\}
\end{aligned} \tag{3.6}$$

From (3.6), observe that data used in the family of methods belong to the domain of e-commerce data. Also, notice that data used when two methods are composed belongs to the domain of e-commerce data. Therefore, the results generated from the graph of a family of methods must be in the domain of e-commerce data. ■

The implication of Lemma 3.1 is that the set of e-commerce data is consistent with the characteristics in Section 3.1.2, since the set of e-commerce data is closed. Recall from Section 3.2 that e-commerce data is bi-directional because the client inputs data to the system and the application transfers back data to the client. E-commerce activities execute a set of methods that use e-commerce data to achieve a desired task. A method is defined in the domain of e-commerce only if it uses e-commerce data. Every result R must be in the set of e-commerce data since the results may also be used as input data to the methods.

3.3 The Four-Phase E-Commerce Application Model

Section 3.2 gives an analysis and specification of the building blocks of e-commerce applications (data, method, and activity). This section provides a synthesis of these components and thereby specifies the key characteristics of an e-commerce application. First, the section discusses a formal model of an e-commerce application with a view to identifying its components and how these components interact with one another to form the basic building blocks for an e-commerce application.

Figure 3.5 shows a four-phase model of an e-commerce application. This four-phase model [AE04b, AE04g] (*locate*, *negotiate*, *deliver*, and *post-sales* services) that shows different actors participating in the system, differs from the popular three-phase model such as [SH98, SKLQ01] because the three-phase model does not incorporate the post-sales phase. Each actor in the set of *actors* (customers, vendors, merchants, producers, etc.) participating in the application is involved in at least one activity. Activities depend on a set of methods that operate on a set of data to accomplish a given task. Application

flow[28] shows the sequence of execution of activities. The end of each phase marks the beginning of a succeeding phase.

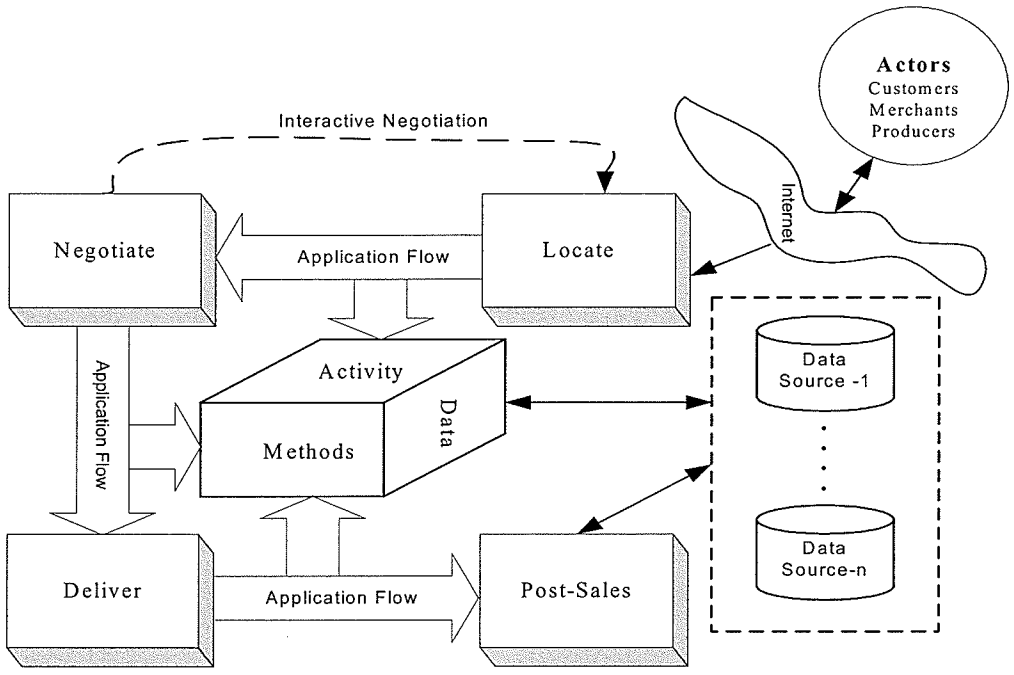

Figure 3.5: A Model of E-Commerce Application

Phase 1 The application starts with the *locate* phase. In this phase, actors are interested in how to find relevant information such as products, services, or new business partners. For example, the actor may be a bookstore employee searching for a publisher. A successful execution of this phase requires certain activities, such as *Login*, *Browse*, *Search*, etc., which depend on a set of methods that operate on a set of data needed by the methods. The search provided by this model transcends keyword-based search to semantic search based on context names. To support this kind of search, we will specify a common-term vocabulary in Chapter 4.

Phase 2 The successful execution of activities in the *locate* phase leads to the *negotiation* phase. Negotiation in a B2B scenario is a sequence of business-induced message exchanges that leads to a contract. Negotiation may involve business partners agreeing on a particular price for goods and services, mode of payment, date and method of shipment,

[28] This research captures application flow using the *precedes* relation discussed in Section 3.2.

etc. However, in a B2C scenario, negotiation may involve a customer accepting the advertised/marked price of a product and proceeding to add the product to his/her shopping cart. In this case, an activity in the negotiation phase might be *Add-to-Cart* that allows the customer to add products to their cart. This phase is designed to support interaction-based negotiation. This includes the ability to return to the *Locate* phase if multiple items are required or if an actor does not agree with the price/conditions of goods or services.

Phase 3 The successful execution of the *negotiation* phase leads to the *delivery* phase. In the delivery phase, contracts are completed as stipulated. The actors make commitments towards a business transaction. This commitment could be the payment for goods and the delivery of such goods. An example of an activity in the delivery phase is the *Payment-Activity*. The *Payment-Activity* facilitates the payment for goods and services by the customer to the merchant. The customer provides the necessary payment information, which forms input to the *Payment-Activity*. The system provides a record that the transaction committed, which may be in the form of receipt.

Phase 4 The *post-sales* services phase is often ignored in most e-commerce models. In this phase, a business organization initiates a follow up of a successfully delivered contract with a customer through post-sales service. Post-sales service is an informal interaction between organizations and their customers designed to meet the individual needs of the customers. A common technique used in the post-sales phase is the use of a Web-Based Community (WBC). A WBC is intentionally designed to promote human interaction and facilitates customer-driven e-commerce [AE04b]. Such a community enables the sharing of experiences, problems and solutions through established discussions and a sense of community among members [Bak98]. For example, Macromedia [Mic03] employs a WBC to facilitate customer participation in product development and maintenance. Similarly, Amazon is known for the book reviews contributed by customers in their online discussion forum.

The e-commerce application model of Figure 3.5 can also be modeled in terms of its states and certain constraints that regulate transitions from one state to another. Figure 3.6 abstracts away irrelevant portions, while presenting a high-level state diagram of the four-phase model of Figure 3.5.

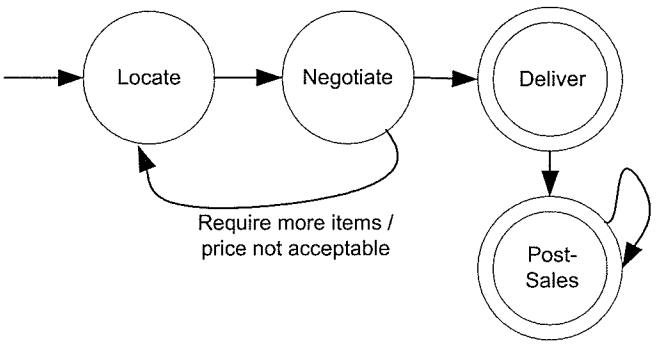

Figure 3.6: A State Diagram for an E-Commerce Transaction Activity

Let Ψ be a set of actors in an e-commerce application such that each *actor*, ACT_i, is involved in at least one activity, $a_i \in A$. Let Φ be a finite set of states that represent the phases p_i in Figure 3.6, where $p_0 \in \Phi$ (locate) is the start state, and p_3, $p_4 \in F$ is a set of final states, such that $F \subseteq \Phi$. Let $\sum$ be a set of input alphabets. The input alphabets are the set of activities A (such that $A \subseteq \sum$) that move an actor ACT_i from one state to another. There is a transition function given by δ: $\Phi \times \sum \rightarrow \Phi$ which takes $ACT_i \in \Psi$ from one state to another. For an *actor* ACT_i to change from one state to another, ACT_i has to execute an activity, a_i. The activity, a_i, depends on a set of methods, M_i, that operate on a set of data item D_i. Notice that D_i can be stored in one or more data sources with diverse structures and syntax. However, M_i needs to operate on D_i to successfully accomplish a given task regardless of the varying structures of D_i in the different data sources. M_i can access D_i in one of two ways: 1) write the same set of methods M_i with the syntax and semantics of D_i in the different data sources; or 2) write a generic M_i that is mapped to the underlying data sources. The former approach is typically manually driven, laborious, and prone to error. The latter approached could be addressed with an efficient data integration mechanism to ensure both logical and physical access transparency in the participating data sources. This thesis adopts the latter approach.

One major problem in the system described by this model is the vast amount of information available and our inability to make sense out of it. For example, how do we determine semantically equivalent data? That is, how does a consumer know that a supplier's product is what s(he) is looking for? How do we manage data generated in

these activities? Regardless of the type of e-commerce system involved, a set of activities needs to be successfully executed in each phase to progress to the next phase of the application. There is need to make these data sources interoperate to carry out a successful e-commerce transaction. Each of these phases requires efficient data management techniques. This research lays emphasis on data management in an e-commerce application to ensure both logical and physical access transparency to data from different data sources. The need is to have this set of data in a form where its syntax and semantics is understandable by the methods that operate on it.

Apart from the basic building blocks (activities, methods, data) discussed in this research, an e-commerce application model has other infrastructure such as communication protocols, message passing, language requirements, payment protocol, security protocol, etc. However, these requirements are outside the scope of this research.

3.3.1 Specifying E-Commerce Applications –An Example (Subset)

The previous section analyzed and specified components of e-commerce applications. In this Section, we formally define an e-commerce application and provide an example to simplify our discussion.

Definition 3.8 An *e-commerce application EA* provides a set of activities A, where each $a_i \in A$ executes a set of methods M to accomplish a given task T_i, such that A collectively accomplishes a set of tasks T implementing the required activities in A. ■

Formally,

$$\begin{aligned} EA \cong A \Rightarrow \{(\forall A_i \in A \bullet \\ \exists m_j \in M; D_j \in ECD; T_i \in T \bullet a_i(M) = T_i \Leftrightarrow \\ (\exists R_j \in ECD \bullet m_j(D_j) = R_j \Leftrightarrow Commit(m_j) \Rightarrow \neg(Abort(m_j)) \wedge \\ (Commit(a_i) \Rightarrow \neg(Abort(a_i)) \wedge T_i = \bigcup_{j=1}^{n} R_j))))\} \end{aligned} \tag{3.7}$$

Example 3.1 In a B2C e-commerce application, payment for goods ordered is a common activity. The *Payment-Activity* enables the clients to pay for goods supplied by the merchant.

Table 3.1: A Typical E-Commerce Activity

Activity	Methods	Data	Results	Tasks
Payment-Activity	*Make-Payment-Method*	Card-No, Amount, Expiry-Date, Card-Type	Balance = Balance - Amount	Client pays merchant for goods supplied and gets receipt.
	Validate-Account-Method	Amount, Balance	Valid / Invalid	
	Receive-Payment-Method	Merchant-ID, Account-No, Amount, Balance	Balance = Balance + Account	
	Print-Receipt-Method		Side effects	

To make payment, the e-commerce application prompts the client to supply account information (Card-No, Card-Type, Expiry-Date). To ease the exposition, we assume that payment is strictly by credit cards. The account information supplied by the client forms the input to the *Payment-Activity*, which depends on the successful execution of some methods, such as *Make-Payment-Method*, *Validate-Account-Method*, *Receive-Payment-Method*, and *Print-Receipt-Method*, to accomplish a task. Table 3.1 gives a summary of the *Payment-Activity*. *Make-Payment* receives the client's account information as an input parameter but does not commit until the client's account is validated. *Validate-Account-Method* is triggered by *Make-Payment-Method* to confirm that the client's account is valid (i.e., that the card is genuine and that the client has enough credit to pay for the goods). The output from *Validate-Account-Method* either initiates *Receive-Payment-Method* if the account is valid; or aborts and then signals *Make-Payment-Method* to display a message indicating that there is insufficient credit in the client's account. If the client's account is valid, the merchant's account is updated (i.e., Balance′ = Balance + Amount) and the client's account is also updated (i.e., Balance′ = Balance - Amount). Then, *Print-Receipt-Method* prints a receipt to show proof of the transaction. The application flow for this transaction if the client's account is valid is thus in the following order:

(*Validate-Account-Method* $\prec_{pre}$ *Make-Payment-Method* $\prec_{pre}$ *Receive-Payment-Method* $\prec_{pre}$ *Print-Receipt-Method*)

Notice that the *Payment-Activity* consists of a set of methods M, where each $m_j \in M$ operates on a set of data D_j and produces certain results R_j. A combination of all the results $\bigcup_{j=1}^{n} R_j$ produced by the set of methods accomplishes the task T_i of a given activity a_i. A typical e-commerce application EA involves a combination of different activities $\bigcup_{i=1}^{n} a_i$ to produce a successful e-commerce solution. Mathematically,

$$EA: A \rightarrow T \Rightarrow \sum_{i=1}^{n}\sum_{j=1}^{n}(a_i(M_j(D))) \rightarrow \bigcup_{i=1}^{n} T_i$$

but $A = \bigcup_{i=1}^{n} a_i$ (by Definition 3.8) (3.8)

$a_i: M \times D \rightarrow T_i$ (by Definition 3.4)

$M = \bigcup_{j=1}^{n} m_j$ (from (3.2))

We know that

$m_j(D_j) = R_j$ (by Definition 3.2)

$\Rightarrow \bigcup_{j=1}^{n} m_j(D_j) = \bigcup_{j=1}^{n} R_j$

$M \times D = \bigcup_{j=1}^{n} R_j$ but $a_i: M \times D \rightarrow T_i$

$\Rightarrow \bigcup_{j=1}^{n} R_j = T_i$ but $\bigcup_{i=1}^{n} a_i(M \times D) \rightarrow \bigcup_{i=1}^{n} T_i$;

$\bigcup_{i=1}^{n} a_i = A$ and $\bigcup_{i=1}^{n} T_i = T$

$\therefore EA: A \rightarrow T$

Example 3.1 abstractly models part of an e-commerce application as a set of activities. The data used in Example 3.1 could be distributed over multiple sites and shared by other activities to provide different functionalities. The transactions in this example depend on the fact that data is represented in a form that the methods understand, regardless of the fact that each supplier might use different structures and vocabularies to describe its products. It depicts a one-to-one relationship where the buyer could well get used to the

supplier's private terminology. The B2C application is for a specific merchant, where the activities, methods and sets of data interact with the same format and language. However, there may be a need for different e-commerce applications to communicate to provide greater functionality and better services. Suppose we have B2B marketplaces that enable communications between n suppliers and m merchants (n x m), where different organizations have different payment systems implementations. To effect payment between organizations, and reduce communications from n x m to $n + m$ mappings[29], the different implementation systems have to be integrated.

Providing a detailed description of e-commerce data sets and analysing the building blocks of e-commerce applications gives better understanding of the systems' components. The chapters that follow describe an algebraic data model that expresses conceptualizations of e-commerce data sources in a natural way, regardless of the complexity of the conceptualizations, a formal model for a common-term vocabulary to support enhanced search techniques and an efficient integration mechanism to integrate e-commerce data from diverse data sources.

[29] The integration system allows each of the n suppliers to access the m merchants without communicating directly with individual merchants.

Chapter 4

Semantic Description of E-Commerce Data Sets

This chapter consists of four major sections. Section 4.1 examines the compatibility of data models. This section is important because it facilitates the choice of a suitable data model for representing e-commerce data sets. Section 4.2 presents a syntactically simple and semantically rich semistructured data model, called *Del-G* [AE05]. *Del-G* is an edge-labelled graph that encapsulates, in it, structural information of the object it represents. *Del-G* is described using algebraic signatures and functions to make explicit both its structure and syntax in an unambiguous manner. Section 4.3 motivates the need for a common-term vocabulary (*CTV*), highlighting the shortcomings of existing approaches [AE04f, EA04]. We give a formal definition of the *CTV* and show that every term in the local data sources can be mapped to the *CTV*. Section 4.4 shows how to extract the semantics of data sources into a corresponding context schema [AE04h]. Central to this semantic description is a filter mechanism that recognizes a context schema and ensures that naming conflicts are eliminated prior to integration of the data sources [AE05]. A context schema, CS_i, for a particular local schema S_i is a schema where every context label, *cl*, in the context schema is also in the *CTV* and for all context labels $cl \in CS_i$ there exist corresponding labels $l \in S_i$.

4.1 Common Data Model for an Integration System

Schema integration requires local schemas in different models to be translated[30] into a common format for ease of representation and integration. It is advisable to adopt a common format that accommodates all local schemas without loss of information. For example, the relational model lacks sufficient semantic power to recognize name conflicts and contradictory specifications. Semantic models can deal with more conflicts than the relational model, but cannot represent complex object models without loss of information. In the object-oriented model (OOM), designers can define objects and the operations/methods associated with the objects. However, the OOM lacks the necessary flexibility to manage e-commerce data that is inherently dynamic. Semistructured data models (e.g. OEM) offer the designer the flexibility to represent data in a less restrictive manner. Table 4.1 highlights the strengths of different data models and areas of likely conflicts between data models used for legacy systems (most likely, systems compatible with the relational model), common commercial systems (relational model and object-oriented data model) and Web-based systems (semistructured data model).

Table 4.1: Compatibility of Data Models

Local Data Sources		**Integrated Schema**		
		Relational	Object-Oriented	Semistructured
Semantic Data Model		Fair	Good	Good
Structured Data Model	Relational	Good	Good	Good
	Object-Oriented	Fair	Good	Good
Semistructured Data Model		Bad	Bad	Good

A "Good" in any cell means the data model of the local schema can be translated to the integrated schema without loss of information. "Fair" means that translation may result in a possible loss of information and "Bad" means that there is some form of

[30] Schema translation is an operation that involves mapping one schema into another when the two schemas are represented using different data models.

incompatibility between the data models. Conflicts occur if the data model for the integrated schema cannot semantically represent all the concepts in any local schema without loss of information. To avoid such conflicts, the data model for the integrated schema should be semantically superior to the corresponding local schemas. Obviously, it is impossible to integrate local schemas represented using a semistructured data model, if the integrated schema is represented using the relational model.

A suitable data model for a common integration platform should express a concept in a natural way, regardless of the complexity of the concept. Differences arise in designers' perception of reality. An appropriate data model should store a single representation of the different perceptions without redundancies. The degree to which a data model accommodates these different perceptions of reality is a measure of its strength [SCG91]. The choice of a particular data model as an integration platform determines to a reasonable extent the kind of data sources that can be integrated in a given integration system [SL90].

4.2 Algebraic Data model

Most of the data used in e-commerce activities reside in some legacy systems, largely represented with rigid schema-based relational databases. E-commerce data are also found on the Web, where the data structure is irregular. Representing e-commerce data with a rigid schema-based relational database model would have been desirable, at least, due to the simplicity and wide use of relational databases in commercial environments. However, Ozsu and Valduriez [OV99] observe that the popular relational data model lacks the flexibility and structures to support complex data (e.g. multimedia objects). In addition, the dynamic nature of e-commerce data makes it increasingly difficult to represent with a model that has a rigid schema.

Data warehouses are able to store e-commerce data using a structured data model where business activities (say, order or sales activities) are associated with business entities (e.g., product, time, customer, merchant). However, the dynamic nature of e-commerce data makes it difficult to manage using the data warehouse concept and often results in the collapse (the warehouse is unable to keep pace with the autonomy and rapid

changes in data sources) of the warehouse. The dynamic and complex nature of Web data requires a flexible data model for its representation. To this end, we define a simple algebraic data model, called *Del-G*, which is a directed edge-labelled graph that encapsulates in it structural information about every object it represents and where each edge is labelled with at most one source label (name). *Del-G* is a syntactically simple and semantically rich semistructured data model that represents data as a set of nodes and edges, similar to existing semistructured data models [BDFS97, PGW95, SLLL97]. Simple data models have inherent advantages over complex models in data integration because the operations to transform and merge data are correspondingly simpler when using simple models [BLN86]. *Del-G* has the following features:

1. A unique singleton node, called the root node;
2. Labels are on the edges (labels carry the basic information);
3. The model contains at most one edge between two nodes; and
4. Represents unknown objects.

We leverage *Del-G*'s simplicity and the expressive power provided by its features to achieve data integration.

4.2.1 Formal Foundation

A directed edge-labelled graph (*Del-G*) for an object is an acyclic digraph that hierarchically represents information about its sub-objects. *Del-G*, as is common with most data models [Ull91], consists of two parts: 1) a notation that describes data using algebraic signatures; and 2) a set of operations used to manipulate the data. The general terminology applicable to trees / graphs applies in the usual way (in this thesis) to refer to relationships between objects in a schema or databases. In particular, we use the following informal definitions to describe tree concepts, where the elements u and v represent nodes, and e an edge in a tree.

- *Child*(u) returns the child elements (sub-elements) of u.
- *Parent*(u) returns the parent of the element u.
- *Ancestors*(e) returns the set of ancestors of e.

- *Descendants*(e) returns the set of descendants of e.
- *Siblings*(u, v) is a boolean function that returns true if u and v have the same direct parent, otherwise it returns false.
- *EdgeLabel*(e) returns the label on an edge e.
- *Source*(e) returns the source node of an edge e.
- *Target*(e) returns the target node of an edge e.

An edge in *Del-G* is a link between an ordered pair of nodes. Thus, $e_i = (u, v)$ represents an edge e_i where $Source(e_i) = u$ and $Target(e_i) = v$. The basic types [NODE, EDGE, LABEL] represent a node, edge, and label, respectively. Let N : $\mathbb{P}_1$ NODE; E: $\mathbb{P}$ EDGE; and L: $\mathbb{P}$ LABEL.

Figure 4.1 shows a generic representation of a *Del-G*. Nodes r_0, u_1, u_2, u_3, v, $w \in N$ represent objects, and the pairs of nodes (r_0, u_1), (r_0, u_2), (r_0, u_3), (u_2, v), and (u_2, w) $\in E$ are edges with the corresponding labels e_1, e_2, e_3, e_4, and $e_5 \in L$. Edges model the hierarchical relationship between ordered pairs of objects where, in each pair, one is called the source object and the other target the object.

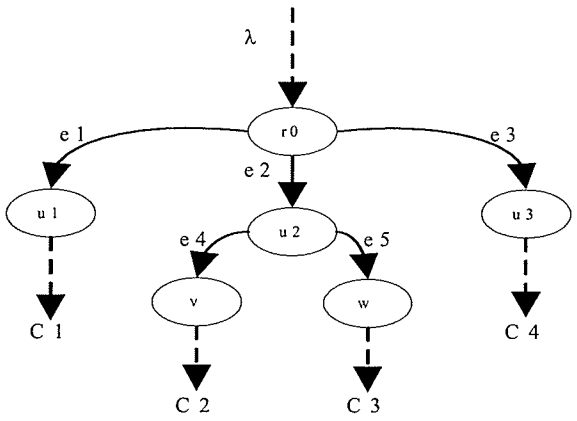

Figure 4.1: A Representation of *Del-G*

A label is a string of characters that describes the edges it represents. There is a unique singleton set called $Root \subseteq N$, whose only member is the unique node, called *RootNode* (r_0). A node $r_0 \in Root$ is a root node if it has no incoming edge thus matching $Parent(r_0) = \varnothing$. The root node is traversed before any other node n in a pre-order traversal. Edges with solid lines represent edges that show hierarchical relationships

between objects, while edges with broken lines are pseudo edges that associate either the *entry-point label* (λ) to r_0 or terminal nodes to constant values, CV ($C_i \in CV$, where $1 \leq i \leq n$). It is necessary to differentiate between the source object and target object of an ordered pair of nodes corresponding to an edge. This reasoning is motivated in part by the fact that an edge embodies a hierarchical order between objects. The definition of the functions *Source* and *Target* follows.

$$\begin{aligned} &Source, Target: \text{EDGE} \rightarrow \text{NODE} \\ &\forall e_i: \text{EDGE} \bullet (\exists u, v: \text{NODE} \mid u \neq v \wedge (u, v) = e_i \bullet \\ &\quad Source(e_i) = u \wedge Target(e_i) = v) \end{aligned} \tag{4.1}$$

For example, in Figure 4.1, $Source(e_4) = u_2$ and $Target(e_4) = v$. We define the function *Parent* to enable the hierarchical reference of objects. A node u is the parent of another node v, written $Parent(v) = u$, in *Del-G* if there exists an edge e_i such that u is the source node and v is the target node. The parent of a node is traversed before the child node in a pre-order traversal. The definition of the function *Parent* follows.

$$\begin{aligned} &Parent: \text{NODE} \rightarrow \text{NODE} \\ &\forall u, v: \text{NODE} \mid u \neq v \bullet \\ &\quad Parent(v) = u \Rightarrow \exists_1 e_i: \text{EDGE} \bullet \\ &\qquad Source(e_i) = u \wedge Target(e_i) = v \end{aligned} \tag{4.2}$$

For example, in Figure 4.1, node u_2 is the parent of node v ($Parent(v) = u_2$). Similarly, an edge e_i is the parent of another edge e_j if the target node of e_i is equal to the source node of e_j. The function *Edge-Parent* defines the parent of an edge.

$$\begin{aligned} &Edge\text{-}Parent: \text{EDGE} \rightarrow \text{EDGE} \\ &\forall e_i, e_j: \text{EDGE} \mid e_i \neq e_j \bullet \\ &\quad Edge\text{-}Parent(e_j) = e_i \Rightarrow \exists u: \text{NODE} \bullet \\ &\qquad Target(e_i) = u \wedge Source(e_j) = u \end{aligned} \tag{4.2a}$$

For example, in Figure 4.1, edge $e2$ is the edge parent of $e4$. Notice that $Target(e_2) = u2$ and $Source(e4) = u2$. Observe that $e2$ is the label that identifies the edge with source node $r0$ and target node $u2$. The function *Edge-Parent* accepts a label as an argument and produces a label as output.

A node v is the *Child* of another node u, written $Child(u) = v$ if $Parent(v) = u$. The definition of the function *Child* follows.

$$Child: \text{NODE} \rightarrow \mathbb{P}\text{NODE}$$

$$\forall u: \text{NODE} \bullet \quad Child(u) = \{v: \text{NODE} \mid Parent(v) = u\} \tag{4.3}$$

Similarly, the function *Edge-Child* defines the child edges of an edge.

$$Edge\text{-}Child: \text{EDGE} \rightarrow \text{EDGE}$$
$$\forall e_i: \text{EDGE} \bullet \quad Edge\text{-}Child(e_i) = \{e_j: \text{EDGE} \mid Edge\text{-}Parent(e_j) = e_i\} \tag{4.3a}$$

The set of non-terminal nodes are represented as *Complex-Nodes* and the set of terminal nodes as *Atomic-Nodes*. A node $u \in Complex\text{-}Nodes$ has at least one child, while a node $v \in Atomic\text{-}Nodes$ has no child.

Siblings(w, v) is a boolean function that returns true if w and v have the same direct parent, otherwise it returns false. Formally,

$$Siblings : \text{NODE} \times \text{NODE}$$
$$\forall u, v: \text{NODE} \mid u \neq v \bullet \quad (Siblings(u, v) = \text{TRUE} \Leftrightarrow \exists w: \text{NODE} \mid (w \neq u \wedge w \neq v) \bullet u, v \in Child(w)) \tag{4.4}$$

The function *Edge-Siblings* returns true if edges e_j and e_k have the same direct parent, otherwise it returns false.

$$Edge\text{-}Siblings : \text{EDGE} \rightarrow \text{EDGE}$$
$$\forall e_j, e_k: \text{EDGE} \mid e_j \neq e_k \bullet \quad (Edge\text{-}Siblings(e_j, e_k) = \text{TRUE} \Leftrightarrow \exists e_i: \text{EDGE} \mid (e_i \neq e_j \wedge e_i \neq e_k) \bullet e_j, e_k \in Edge\text{-}Child(e_i)) \tag{4.4a}$$

To facilitate the definition of ancestors and descendants of an element in *Del-G*, we first define a path.

A *path* P (from a source node to a target node) in *Del-G* is a sequence of edges (e_1, e_2, …, e_m) labelled with a corresponding sequence of labels (l_1, l_2, …, l_m) such that $Target(e_i) = Source(e_{i+1})$, $1 \leq i \leq$ m-1. P is a path from the source node of e_i to the target node of e_m. The length of the path is the number of edges in the path.

The ancestor set of a node v in *Del-G* consists of the nodes n on the unique path P from the root r_0 to v. From this definition, a node is an ancestor of itself. A proper ancestor set of v does not include v. The signature of the function, *AncP* that returns the set of proper ancestors of v follows.

$$AncP : Node \rightarrow \mathbb{P}(Node)$$

The following conditions hold for the ancestor set of a node:

1. $AncP(r_0) = \varnothing$, the root node has no ancestor;
2. $\forall v : Node \bullet v \notin AncP(v)$, no node can be its own ancestor; and
3. $\forall v : Node \bullet \exists r_0 \in Root \bullet (r_0 = v) \vee (r_0 \in AncP(v))$, there is exactly one root such that $r_0 = v$ or $r_0 \in AncP(v)$.

A node v is in the descendant set of the node u if and only if u is in the ancestor set of v. From this definition u is a descendant of itself. A proper descendant set of u does not include u. The definition of the function, *DesP* that returns the set of proper descendants of v follows.

$$\begin{aligned} &DesP : NODE \rightarrow \mathbb{P}\, Node \\ &\forall u : Node \bullet \\ &\quad DesP(u) = \{v : Node \mid u \in AncP(v)\} \end{aligned} \tag{4.5}$$

$AncP, DesP \subseteq NODE$. *AncP* can determine *DesP* and vice versa. Therefore, the two functions are symmetric. Additional functions and terminology are defined in the contexts in which they are needed.

4.2.2 The *Del-G* Model

In *Del-G*, every edge is associated with a label. The definition of a labelling function, *EdgeLabel* that maps every edge to a label follows.

$$\begin{aligned} &EdgeLabel: \text{EDGE} \rightarrow \text{LABEL} \\ &\forall e: Edge \bullet (\exists_1 l: \text{LABEL} \bullet \\ &\quad EdgeLabel(e) = l \end{aligned} \tag{4.6}$$

Definition 4.1 A directed edge-labelled graph, *Del-G*, of an object O is a 3-tuple (N, E, L) where:

1. N is a finite nonempty set of nodes.
2. E is a set of edges; and $\forall e: EDGE \mid e \in E \bullet \exists u, v: \text{NODE} \mid u \neq v \bullet$ $Source(e) = u \wedge Target(e) = v$.
3. L is a set of labels; and $\forall e: \text{EDGE} \bullet \exists_1 l: LABEL\ L \mid l \in L \bullet EdgeLabel(e) = l$.

Point 1 specifies that N is a finite nonempty set of nodes (that represent sub-objects of O). By the inheritance property of objects, a node inherits the structural information of the object it represents. Point 2 specifies E as a set of edges, and an edge as an ordered pair of

nodes (u, v), where u is the source and v is the target. Point 3 specifies L as a set of labels where every edge has at most one label. ■

The schema of an object is a set of nodes that describes the structure of the object using a *Del-G*. The *Del-G* model is defined with its unique syntax and its associated semantics. The syntax is derived from the graph structure of *Del-G*. The semantics of the model show the inherent relationships (including parent, child, ancestor, descendant, etc.) that exist between objects.

Let *CV* be the set of constant values. The function *Val* is a function that maps terminal nodes in *Atomic-Nodes* to *CV*. A formal definition of the schema of an object follows.

Definition 4.2 The *Schema* of an object $S = (\varphi, r_0, \lambda, Val)$ is a *Del-G* where:

1. $\varphi = \cup$ (*Root*, *Complex-Nodes*, *Atomic-Nodes*) $\subseteq N$ |
   $\cap$ (*Root*, *Complex-Nodes*, *Atomic-Nodes*) $= \varnothing$;
2. $r_0 \in$ *Root*;
3. λ: LABEL; and
4. *Val*: *Atomic-Nodes* $\rightarrow$ *CV*.

Point 1 specifies that φ is the union of all the nodes and that the intersection of the different sets of nodes (*Root*, *Complex-Nodes* and *Atomic-Nodes*) is empty. Point 2 specifies that r_0 is a special node, called the root node. Point 3 specifies that λ is a special label, called the entry point label. Point 4 specifies that *Val* is a function that maps elements from the set of *Atomic-Nodes* to the set of constant values (*CV*).

A *Schema* also satisfies the following conditions:

a. terminal nodes have no outgoing edges;
b. the root node has no incoming edge, but every schema has an entry point label that is associated with the root node; and
c. each node n is reachable from the root node r_0 and its label, λ. ■

Let SCHEMA be the basic type of a schema. A formal specification of a schema follows.

$$\forall S: \text{SCHEMA} \bullet S = (\varphi, r_0, \lambda, Val) \Leftrightarrow$$
$$\varphi = \cup (Root, Complex\text{-}Nodes, Atomic\text{-}Nodes) \mid$$
$$\cap (Root, Complex\text{-}Nodes, Atomic\text{-}Nodes) = \varnothing \wedge \qquad (4.7)$$

$$\exists r_0: Root \bullet NodeLabel(r_0) = \lambda \ \wedge$$
$$\forall e: \text{EDGE} \bullet (\exists u, v: \text{NODE} \mid u \neq v \wedge u, v \in \varphi \bullet$$
$$Source(e) = u \wedge Target(e) = v) \wedge$$
$$(\exists l: \text{LABEL} \bullet EdgeLabel(e) = l) \wedge$$
$$Val(Target(e)) \in CV \Leftrightarrow Target(e) \in Atomic\text{-}Nodes)))$$

Example 4.1 Figure 4.2 shows an example of a *Del-G* schema with information about authors of books.

Author [Name, Birthday, Book [Title, ISBN]]

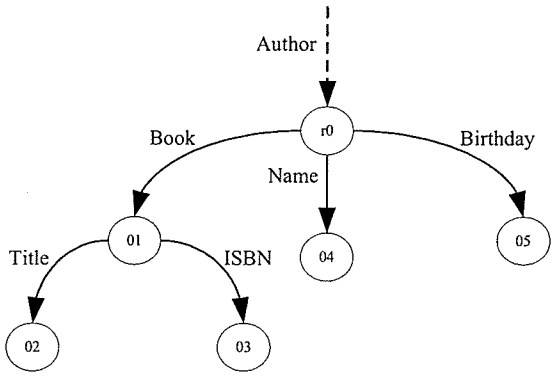

Figure 4.2: Graphical Representation of Semistructured Data in *Del-G*

In this representation, square brackets enclose children (sub-objects) of the object directly before the left square bracket and commas separate siblings of a given object. Author is the entry-point label associated with the object being described, and its value is a sequence of sub-objects enclosed in the square bracket. These sub-objects may include atomic objects as in [Name] or complex objects as in [Book [Title, ISBN]], and the values of atomic objects are drawn from the set of constant values *CV*. A semistructured data consists of the sequence of labels along the paths that can be found in the edge-graph. In the representation of Figure 4.2, the entry point label "Author" is associated with the root node r_0; (01) $\in$ *Complex-Nodes*, while (02, 03, 04, 05) $\in$ *Atomic-Nodes*.

In the *Del-G* model, as in the object exchange model (OEM), information only resides at the labels, and labels are used in place of a schema. However, unlike OEM, *Del-G* attaches labels on the edges instead of the nodes to be able to exploit path knowledge. Path knowledge is important to successful access of semistructured data because it identifies a unique singleton node (root node) that connects the entry-point

label to all other labels in the graph. This model, like OEM, is self-describing, and has no a priori schema. To avoid repeatedly traversing sub-graphs due to multiple edges between two nodes, *Del-G* contains at most one edge between two nodes. There is a natural recursive ancestral relationship that creates the notion of path knowledge, and hence enhances the quality of queries that may be posed. The *Del-G* model mimics the simplicity of OEM, which is flexible for semistructured data, yet powerful enough to represent other models and enhance integration. Therefore, the model is able to provide information exchange among organizations, since it can represent most existing models.

4.3 The Need for a Common-Term Vocabulary (*CTV*)

Classical integration techniques depend on significant manual effort to integrate data sources. For example, MRSDM [LA87] and VIP-MDBMS [KL88] require user intervention in identifying and analyzing correspondence assertions while Multibase [LR82] and Mermaid [TBC+87] demand an *a priori* identification and analysis of correspondence assertions by the DBA. Identifying and analyzing correspondence assertions manually is a difficult and labour-intensive task that requires sufficient support by the integration system. Wache *et al.* [WSSK99] and Sheth and Larson [SL90] argue that providing appropriate support for an integration methodology requires explicit semantic description of each data source in the integration system.

Ontologies provide a framework for participants in a given domain to speak a common language and hence understand one another. Gruber [Gru93] defines ontology as "an explicit specification of a shared conceptualization". Gruber's definition reflects the notion that an ontology captures consensual knowledge about an abstract model of some phenomena that is shared by a group. This real world phenomenon identifies the relevant concepts used in a given domain. The type of concepts used and the constraints on their use are explicitly defined without ambiguities [Fen01].

There have been several attempts to use ontologies in providing enterprise integration. For example, Uschold *et al.* [UKMZ98] examine strategies that leverage ontologies for information exchange in business transactions. TOronto Virtual Enterprise

(TOVE)[31] [FCF93, FG97] is a task and domain-specific ontology that supports enterprise integration by providing a sharable representation of knowledge. TOVE provides a shared terminology for the enterprise and the meaning of each term is defined in a precise and unambiguous manner.

Wache *et al.* identify two main problems associated with ontology-based integration. These problems are: 1) it is expensive and difficult to develop an ontology since ontological developments start from scratch; and 2) the fixed and static nature of an ontology leads to low scalability since the kind of information that will be integrated in the future is limited. For example, OBSERVER [MKSI96] assumes a predefined ontology for each information source. As a result, new information sources can easily be added and old ones removed. However, mapping terms between the heterogeneous ontologies leads to many homonym, synonym, etc. problems, because the ontologies use their own vocabularies. In addition, industrial standards and approaches that use common ontologies lack flexibility because they prescribe standard schemas. These problems limit the use of a common ontology to provide a scalable integration system, and hence, bring to focus the need for a more flexible and scalable integration approach.

To provide flexibility and scalability in an integration system, Goh [GBMS99] suggests semantic source description of each data source using a shared vocabulary. COIN [GBMS99] uses an attribute value vector to provide a "context"[32] that gives the local description of an information source. The terms for the context come from the shared vocabulary and the data itself. Milo and Zohar [MZ98] also define information sources as graphs and provide support for the comparison of objects. These approaches [GBMS99, MZ98] lack a clear strategy in guiding a user and the comparison of objects is reduced to syntax alone.

This research specifies a generic *CTV* that gives meaningful definitions of terms. The *CTV* contains primitive terms in a given domain. The *CTV* is used by all participating data sources. Contrary to industrial standards and approaches that use common ontologies, the *CTV* does not prescribe any standard schema, but provides a set of

---

[31] http://www.eil.utoronto.ca/tove/toveont.html

[32] Context is the set of facts in a statement that specifies the meaning of a word, or a passage in that particular statement.

primitives used to describe terms appearing in data sources. This flexibility permits data sources with different structures to participate in the *CTV*. The *CTV* is organized hierarchically using linguistic relations to show how terms relate to one another. To capture these linguistic relations, we let CONTEXT-LABEL be the basic type for context labels. The linguistic relations include:

1. Synonym - two words that can be interchanged in a given context (equivalent terms);
   $\forall cl_i, cl_j : \text{CONTEXT-LABEL} \mid cl_i \neq cl_j \bullet cl_i \equiv cl_j$
2. Hypernym - a generalization of a given term (superordinate);
   $\forall cl_i, cl_j : \text{CONTEXT-LABEL} \mid cl_i \neq cl_j \bullet cl_i \supseteq cl_j$
3. Hyponym - a more specialized term that belongs to a lower rank in the structure (commonly referred to as "is -a" relation);
   $\forall cl_i, cl_j : \text{CONTEXT-LABEL} \mid cl_i \neq cl_j \bullet cl_i \subseteq cl_j$
4. Meronymy - a semantic relationship between concepts that show how terms are related to each other where one is a part and the other a whole (commonly referred to as a "part-of" relation).
   $\forall cl_i, cl_j : \text{CONTEXT-LABEL} \mid cl_i \neq cl_j \bullet$
   $(\exists cl_k : \text{CONTEXT-LABEL} \mid cl_k \neq cl_i \wedge cl_k \neq cl_j \bullet$
   $cl_k = \cup (cl_i, cl_j))$

CTV is a power set of context labels and the linguistic relationship between pairs of context labels. To give a formal definition of a *CTV*, we first, define a *context label*, *cl*, as a primitive term (word) that has a unique meaning in the real world.

Let *LRI* = {*synon*, *hyper*, *hypon*, *meron*} be the set of linguistic relationship identifiers, where *synon*, *hyper*, *hypon* and *meron* are synonym, hypernym, hyponym, and meronymy, respectively. A formal definition of a linguistic relation follows.

Definition 4.3 A *Linguistic Relation*, $\Re$, is a partial function that maps context labels $cl \in CL$ from the set of all pairs of context labels to the set of linguistic relationship identifiers. ■

Formally,

$\Re$: CONTEXT-LABEL × CONTEXT-LABEL → *LRI*

Definition 4.4 A Common-Term Vocabulary (*CTV*) is a pair (*CL*, $\Re$), where *CL* is a set of context labels and $\Re$ is a linguistic relation which shows that given cl_i, $cl_j \in CL$, then the relationship between cl_i and cl_j is one of {synon, hyper, hypon, meron} (i.e., $\Re(cl_i, cl_j) \in LRI$). ■

Formally,

$$
\begin{aligned}
&\text{CTV} \triangleq \mathbb{P}_1 \text{ CONTEXT-LABEL} \bullet \\
&\quad \forall cl_i, cl_j : \text{CONTEXT-LABEL} \mid cl_i \neq cl_j \bullet \\
&\qquad (\Re(cl_i, cl_j) = \{synon\} \Rightarrow cl_i \equiv cl_j \vee \\
&\qquad\quad \Re(cl_i, cl_j) = \{hyper\} \Rightarrow cl_i \supseteq cl_j \vee \\
&\qquad\qquad \Re(cl_i, cl_j) = \{hypon\} \Rightarrow cl_i \subseteq cl_j \vee \\
&\qquad\qquad\quad \Re(cl_i, cl_j) = \{meron\} \Rightarrow \\
&\qquad\qquad\qquad (\exists cl_k : \text{CONTEXT-LABEL} \mid cl_k \neq cl_i \wedge cl_k \neq cl_j \bullet \\
&\qquad\qquad\qquad\quad cl_i, cl_j \subseteq cl_k))
\end{aligned}
\tag{4.8}
$$

4.3.1 *CTV* Design Principles

Basically, the goal of the *CTV* is to provide a common terminology for the enterprise that is understandable to all participating data sources. The *CTV* must define the meaning of each term in a precise and unambiguous manner and show the semantic relationship between terms. The interest of this research is to specify a *CTV* at the schema level and not instances of objects (at the data item level). The entities in the *CTV* are similar to the ones discussed in Section 3.1.3[33]. It is assumed that an independent group of domain experts familiar with typical tasks and problems in the given domain develops the *CTV*. The domain of e-commerce is large so it will be difficult for a single group of domain experts to develop a "super" *CTV* that captures all the terms in the e-commerce domain. The development of *CTVs* in the e-commerce domain will have to be a cooperative process involving different domain experts, possibly at different locations. Therefore, we propose area-centric *CTV* development. Area-centric *CTV* development is a phased incremental approach to developing a *CTV* with respect to a given area (e.g. automobile, tourism, publishing, etc.) within the e-commerce domain.

Since the aim of this thesis is to encourage scalability of the integration system, and hence allow participation of many data sources in a business transaction, there is a need

[33] Recall, the characterization of e-commerce data sets facilitates the creation of a suitable *CTV*.

to have large *CTVs* servicing different areas in the e-commerce domain. Essentially, most large *CTV* applications will need to compose other smaller *CTVs* in a given area. Composition of *CTVs* will encourage reuse of existing *CTVs* to build larger *CTVs* without undermining their local linguistic relations. Reusability is a conscious design principle to reduce the cost of constructing large *CTVs* and guarantee correctness. Since the *CTV* is designed for a Web-centric application, it has to be globally available and accessible to all participants. Global availability requires that the *CTVs* be distributed among different servers (hosts). A major challenge with distributed systems is the problem of consistency when systems evolve. In the sections that follow, we discuss composition of *CTVs* to support reusability and *CTV* evolution.

4.3.2 Composition of *CTV*s

Figure 4.3 shows a network of three service providers in the e-commerce domain. These service providers have their respective *CTVs* created by independent domain experts. *CTV-A* is defined at a server that provides services for the airline industry and *CTV-B* is defined at a server that provides services for the hotel industry. Some of the services provided by the tourism industry are also provided by both the airline and hotel industries. As a result, the tourism industry may choose to reuse the *CTVs* of both the airline and hotel industries instead of developing one from the scratch. *CTV-C* is defined at a server that provides services for the tourism industry. *CTV-A* and *CTV-B* may be replicated in the server of *CTV-C* to be reused by *CTV-C*.

Observe that these *CTVs* are developed independently and service the needs of the enterprises local to a given area. However, business transactions in different areas may overlap, thus motivating cooperation between different local *CTVs*. Informally, the composition of two *CTVs*, namely, *CTV-A* and *CTV-B*, is a third *CTV* called *CTV-Comp*. *CTV-Comp* is the union[34] of the two *CTVs* (e.g., *CTV-A* and *CTV-B*) where all the context labels of *CTV-A* and *CTV-B* are also contained in *CTV- Comp* such that no existing relationship $\Re_A$ in *CTV-A* and $\Re_B$ in *CTV-B* is compromised in *CTV- Comp*.

[34] *CTV-Comp* is a union of the composing *CTVs* since every term in *CTV-Comp* is unique. A particular term may occur in more than one composing *CTVs*, it can only be represented once in *CTV-Comp*.

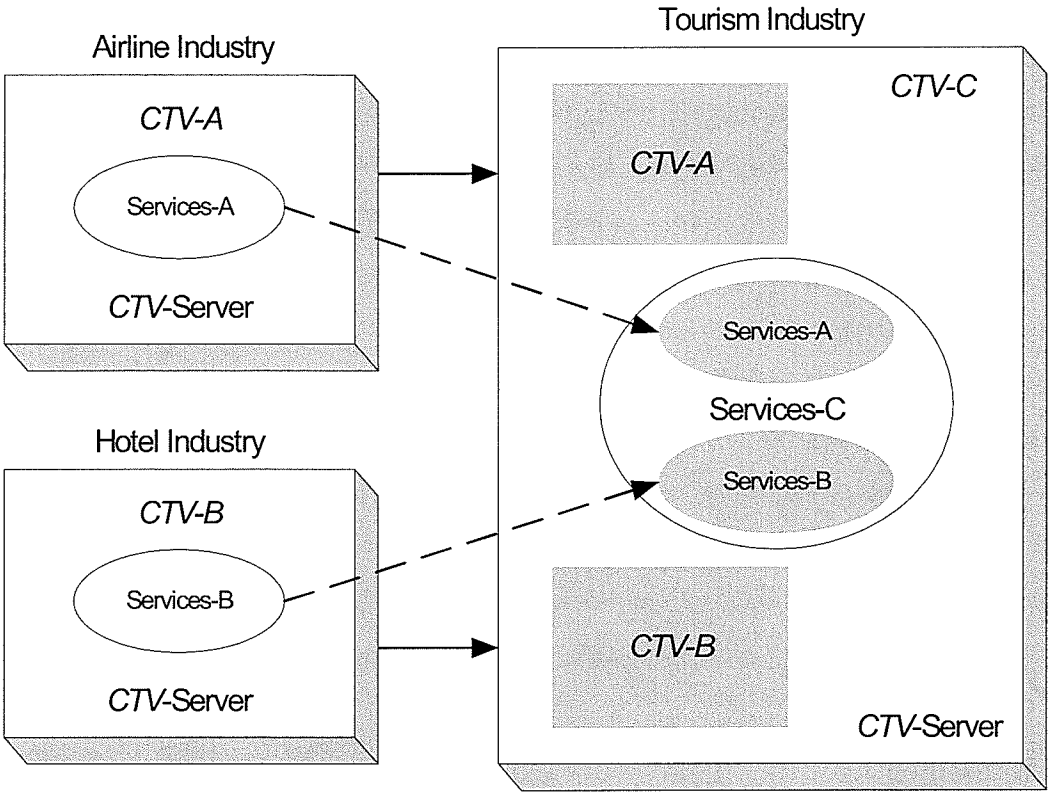

Figure 4.3: Composition of *CTVs* in a Distributed Environment

A formal definition of the function *Compose-CTV-AB* that returns *CTV-Comp* (a composition of *CTV-A* and *CTV-B*) follows.

Compose-*CTV-AB*: $CTV \times CTV \rightarrow CTV$

$$
\begin{aligned}
&\forall\ CTV\text{-}A, CTV\text{-}B, CTV\text{-}Comp: \mathbb{P}_1\ \text{CONTEXT-LABEL} \bullet \\
&((\exists cl\text{-}A_i, cl\text{-}A_j : \text{CONTEXT-LABEL} \mid (cl\text{-}A_i, cl\text{-}A_j \in CTV\text{-}A \wedge cl\text{-}A_i \neq cl\text{-}A_j) \bullet \\
&\quad (\Re_A(cl\text{-}A_i, cl\text{-}A_j) \in \{synon, hyper, hypon, meron\}) \wedge \\
&\quad\quad (\exists cl\text{-}B_k, cl\text{-}B_l : \text{CONTEXT-LABEL} \mid (cl\text{-}B_k, cl\text{-}B_l \in CTV\text{-}B \wedge cl\text{-}B_k \neq cl\text{-}B_l) \bullet \\
&\quad\quad\quad (\Re_B(cl\text{-}B_k, cl\text{-}B_l) \in \{synon, hyper, hypon, meron\}) \bullet \\
&\quad\quad\quad\quad (Compose\text{-}CTV\text{-}AB(CTV\text{-}A, CTV\text{-}B) = CTV\text{-}Comp \Leftrightarrow \\
&\quad\quad\quad\quad\quad CTV\text{-}A \cup CTV\text{-}B = CTV\text{-}Comp \bullet \\
&\quad\quad\quad\quad\quad\quad ((\forall cl\text{-}A_i, cl\text{-}A_j \in CTV\text{-}A \bullet \exists cl_i, cl_j : \text{CONTEXT-LABEL} \mid cl_i, cl_j \in CTV\text{-}Comp \bullet \\
&\quad\quad\quad\quad\quad\quad\quad cl\text{-}A_i = cl_i \wedge cl\text{-}A_j = cl_j) \wedge (\forall cl\text{-}B_k, cl\text{-}B_l \in CTV\text{-}B \bullet \\
&\quad\quad\quad\quad\quad\quad\quad\quad \exists\ cl_k, cl_l : \text{CONTEXT-LABEL} \mid cl_k, cl_l \in CTV\text{-}Comp \bullet \\
&\quad\quad\quad\quad\quad\quad\quad\quad\quad cl\text{-}B_k = cl_k \wedge cl\text{-}B_l = cl_l) \wedge (\forall cl_i, cl_j \in CTV\text{-}Comp \bullet \\
&\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \exists\ \Re_S(cl_i, cl_j) = \{synon, hyper, hypon, meron\} \bullet \Re_S = \Re_A) \wedge \\
&\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad (\forall cl_k, cl_l \in CTV\text{-}Comp \bullet \\
&\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \exists\ \Re_S(cl_k, cl_l) = \{synon, hyper, hypon, meron\} \bullet \Re_S = \Re_B))))
\end{aligned}
\tag{4.9}
$$

CTV composition allows reusing existing *CTVs* available within one node (server) in the system. The above formalism also allows the inclusion of *CTVs*.

4.3.3 Evolution of *CTVs* in a Web-Centric Application

The distribution of *CTVs* across several nodes (servers) requires timely adaptation of a *CTV* and consistent propagation of changes to dependent artefacts. Recall that the domain experts who develop the *CTVs* also manage the evolution of such *CTVs*. To capture such changes in the *CTV*, the function *Update-CTV* adds new context labels into the *CTV*, while maintaining existing relationships between context labels. Let MAX-CONTEXT-LABEL be the maximum number of terms in the *CTV*. A definition of the function *Update-CTV* follows.

$$
\begin{aligned}
&\textit{Update-CTV}: \text{CONTEXT-LABEL} \rightarrow \textit{CTV} \\
&\quad \forall cl_k : \text{CONTEXT-LABEL} \mid cl_k \notin \textit{CTV} \bullet \\
&\qquad (\exists cl_j : \text{CONTEXT-LABEL} \mid cl_j \in \textit{CTV} \bullet \\
&\qquad\quad (\text{MAX-CONTEXT-LABEL}' = \text{MAX-CONTEXT-LABEL} + 1 \wedge \\
&\qquad\qquad \textit{Update-CTV}\,(cl_k, \textit{CTV}) \Rightarrow \textit{CTV}' = \textit{CTV} \cup \{cl_k\} \wedge \\
&\qquad\qquad\quad \Re(cl_k, cl_j) = \{\textit{synon}, \textit{hyper}, \textit{hypon}, \textit{meron}\}))
\end{aligned}
\tag{4.10}
$$

CTV' is the new state of the *CTV* and MAX-CONTEXT-LABEL' is the new value of MAX-CONTEXT-LABEL after update. The interest of this thesis is neither in the exact definition nor the structure of the *CTV* since no global agreement is needed for its structure. However, the main interest is for the *CTV* to capture the semantics of every term from the participating data sources. The above specification captures the notion that every term, l_i, in any participating source schema, S_i, has exactly one context label, cl_i, in the *CTV* that semantically describes the term.

One way to achieve reuse of *CTVs* in Web-centric applications is to replicate distributed *CTV* models locally and to include them in other *CTV* models [MMS03]. Replication eliminates problems such as:

1. In a tightly coupled server (where servers share resources), the failure of one system will cause the failure of all servers that include the *CTV* model.

2. Overload of top-level[35] *CTVs* that are reused in many other *CTVs*.
3. Performance bottleneck arising from answering distributed queries with a centralized system.

However, replication introduces significant evolution and consistency problems (*CTV* evolution will be discussed shortly). The most important constraint is that replicated *CTVs* should never be modified directly. Instead, the modification should always be performed at the source and changes propagated to replicas using a distributed evolution process. To replicate a *CTV* requires physical access to the *CTV*. *CTVs* on the Web will be identified using uniform resource identifiers (URI), which can be used to access the *CTV* through appropriate protocols (e.g. Hypertext Transfer Protocol - HTTP). In distributed systems, consistency problems arise with replication. For example, the URI used to access the *CTV* and the URI under which the *CTV* is known originally may become different when the *CTV* is replicated. To address this consistency problem, Maedche *et al.* [MMS03] suggest that: 1) the logical URI for each *CTV* should be unique; and 2) the physical URI should unambiguously identify the location of the *CTV*.

When the *CTV* is updated, there is a propagation of new concepts into the *CTV'*. Bhide *et al.* [BDK+02] suggest two ways of propagating *CTVs*: 1) a Push-based approach, which requires instant propagation of changes from the changed *CTV* to dependent *CTVs*; and 2) a *Pull-based approach*, which requires an explicit request from dependent *CTVs* to propagate changes from the changed *CTV* to dependent *CTVs*. Maedche *et al.* propose the use of a hybrid synchronization strategy that combines favorable features of each approach. The pull approach is used for synchronizing original *CTVs* and replicas, while the push approach is used for maintaining consistency of *CTVs* within one node (server). In the pull approach, replication consistencies are enforced upon request. Information about included *CTVs* is stored in the dependent *CTV*, thus eliminating the need for central dependency management. Original *CTVs* are checked periodically to detect changes and collect deltas[36].

---

[35] These are standard *CTVs* that many other *CTVs* may depend on. Many other servers will often contact servers hosting these top-level CTVs.

[36] Deltas contain all changes that have been applied to the original *CTV* after the last update of the replica, as determined by the version numbers.

4.4 Semantic Description of Data Sources

A *CTV* contains typical terms used in some e-commerce application domain, and therefore, provides a natural source for finding context labels for the context schemas. A *context schema* is a collection of axioms which semantically describe a given data source. The *CTV* is used to annotate terms from the different data sources. Annotation in this context refers to the process of extracting the inherent concept hierarchy of a data source where each concept is described semantically using a context label from the *CTV*. To facilitate a formal definition of a context schema, and to provide a theoretical foundation for its acceptance by a finite state automaton, the definitions of the following concepts are required.

Definition 4.5 A *regular expression* R over the alphabet $\Sigma \mid \Sigma \subseteq CTV$ is defined as follows[37]:

i) ε is a regular expression;

ii) $a \mid a \in \Sigma$ is a regular expression;

iii) if R_1 and R_2 are regular expressions, then $(R_1 \cup R_2)$ is a regular expression;

iv) if R_1 and R_2 are regular expressions, then $(R_1.R_2)$ is a regular expression;

v) if R_1 is a regular expression, then (R_1^*) is a regular expression;

vi) no expression is a regular expression unless it is obtained from (i) – (v).

Let $L(R)$ be the language associated with a regular expression, R, over the alphabet, Σ. Then $L(R) \subseteq \Sigma^* \models L(R) \subseteq CTV$. A recursive definition of $L(R)$ follows:

a) $L(\varepsilon) = \{\varepsilon\}$;

b) if $a \in \Sigma$, then $L(a) = \{a\}$;

c) $L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$;

d) $L(R_1.R_2) = L(R_1).L(R_2)$;

e) $L(R_1^*) = L(R_1)^*$;

where L^* is the reflexive and transitive closure of the language L. ■

[37] The union of two regular expressions $(R_1 \cup R_2)$ is a regular expression; the concatenation of two regular expressions $(R_1.R_2)$ is also a regular expression; and the value of $\{0\}^*$ is the language consisting of all strings with any number of 0s.

Definition 4.6 A *Path-Label* (*pl*) in *Del-G* is a sequence of labels (λ. l_1. … . l_n) $\subseteq$ *CTV* on a path, each delimited by a period, such that λ is the entry point label, and the labels l_i correspond to edges e_i ($1 \leq i \leq n$), and *Target*(e_n) $\in$ *Atomic-Nodes*. The pseudo edge, e_0, is associated with the entry-point label λ. ■

Definition 4.7 A *Data-Path* (*dp*) in *Del-G* is an alternating sequence of labels on edges, and nodes delimited by periods of the form ($\lambda r_0.l_1o_1$. …. l_no_n), such that a path of *n* edges ($e_1.e_2$. …. e_n) could be traversed in a predefined order through (*n* +1) nodes ($r_0.o_1$. … o_n). In addition, every *dp* in *Del-G* satisfies the following conditions.

1. λ is the entry point label.
2. l_1 is the label of edge e_1 such that *Source*(e_1) = r_0 and *Target*(e_1) = o_1.
3. l_n is the label of edge e_n such that *Source*(e_n) = o_{n-1} and *Target*(e_n) = $o_n \wedge o_n \in$ *Atomic-Nodes*. ■

The basic types [PATH-LABEL, DATA-PATH] represent a path label and a data path, respectively. The following constraints hold for path labels and data paths.

$$\forall pl_i, pl_j: \text{PATH-LABEL} \bullet i \neq j \Rightarrow pl_i \neq pl_j$$
$$\forall dp_i, dp_j: \text{DATA-PATH} \bullet i \neq j \Rightarrow dp_i \neq dp_j$$

Example 4.2 To illustrate the definitions given above, consider the schema of a *Del-G* object *O* represented in Figure 4.2. (Author.r_0.Book.01.Title.02), for example, is a data path of the schema of Figure 4.2, where the edges labelled (Author, Book, Title) could be traversed through the nodes (r_0, 01, 02), and "Author" is the entry point label. "Book" is the label of edge e_1 such that *Source*(e_1) = r_0 and *Target*(e_1) = 01. "Title" is the label of edge e_n such that *Source*(e_n) = 01 and *Target*(e_n) = 03 and 03 $\in$ *Atomic-Nodes*. The *Del-G* of Figure 4.2 can be completely represented by the union of *Data-Paths*, $\bigcup_{i=1}^{n} dp_i$. Table 4.2 shows data paths and the corresponding path labels for Figure 4.2. A *k-dp Del-G* is a *Del-G* where the cardinality of the set of *Atomic-Nodes* is *k*. The *Del-G* for Figure 4.2 can be completely represented by a set of *k* data paths {dp_1, dp_2, …, dp_k} that are not prefixes of

each other. Notice that for every unique *Data-Path* dp_i there exists a corresponding unique *Path-Label* pl_i.

Table 4.2: Data Paths and Path Labels of Figure 4.2

Data paths	**Corresponding Path Labels**
dp_1 = Author.r_0.Book.01.Title.02	pl_1 = Author.Book.Title
dp_2 = Author.r_0.Book.01.ISBN.03	pl_2 = Author.Book.ISBN
dp_3 = Author.r_0.Name.04	pl_3 = Author.Name
dp_4 = Author.r_0.Birthday.05	pl_4 = Author.Birthday

Therefore, we can represent a *Del-G* structure with *Path-Labels*, which are regular expressions. Next, we define a well-formed schema in the context of regular expressions.

Definition 4.8 A *well-formed Schema*, S^{Wf}, for an object O is a 4-tuple (X_n, X_l, PL, DP) where

1. X_n is a finite set of nodes;
2. X_l is a set of labels;
3. PL is a set of path labels pl_i; and
4. DP is a set of data paths dp_i.

In addition, S^{Wf} must satisfy the following conditions.

a) for each regular expression ($pl_i \in PL$) there exists some edge labels in X_l (e.g. $pl_1 \in PL$ = Author.Book.Title, where Author, Book, Title $\in X_l$); a path label pl_i connects labelled edges to nodes in the form: $a \xrightarrow{pli} b$ where $a, b \in X_n$ and pl_i is a regular expression over X_l;

b) for each data path ($dp_i \in DP$), there exist some sequence(s) of nodes and edges in S^{Wf} such that each sequence is in the language $L(dp_i)$;

c) every node (excluding the root node) has at most one parent. $Parent(r_0) = \varnothing$; and

d) no two siblings in S^{Wf} are the same.

■

A formal definition of conditions c) and d) follows.

$$\begin{aligned}&\forall v\text{: NODE} \bullet (\exists S\text{: SCHEMA} \wedge \exists u, w\text{: NODE} \mid u, w \in S \bullet \\ &\quad (Parent(v) = u \wedge Parent(v) = w \Rightarrow u = w) \wedge \\ &\qquad \forall a, b\text{: NODE} \mid a \neq b \wedge Siblings(a, b) \bullet (\exists l_i, l_j : \text{LABEL} \bullet \\ &\qquad\quad NodeLabel(a) = l_i \wedge NodeLabel(b) = l_j \Rightarrow l_i \neq l_j)\end{aligned} \tag{4.11}$$

This thesis assumes that the DBA correctly maps every term in a source schema to the corresponding context label in the *CTV*. Let *LB* be a set of context labels in the context schema. Then, the partial function $\delta(l_j, cl_i) = lb_i$ that maps $l_i \in L$ and $cl_i \in CL$ to $lb_i \in LB$ correctly associates the terms. A *context schema*, CS_i, is a machine-readable form of a particular local schema, S_i, where every context label, *cl*, in the context schema is also in the *CTV*. A formal definition of a context schema follows.

Definition 4.9 A *Context Schema*, *CS*, for an object *O* in *Del-G* is a well-formed schema represented as a six-tuple (*CL*, *CTV*, *L*, S_i, *LB*, δ) where:

1. $CL \in CTV$ is the set of context labels from the *CTV*;
2. $L \in S_i$ is the set of labels from the local data source S_i;
3. $LB \subseteq CL$ is the set of context labels in the context schema; and
4. δ is a partial function that maps terms $l_i \in L$ and $cl_i \in CL$ to $lb_i \in LB$.

Formally, ∎

$$\begin{aligned}&\forall lb_i\text{: CONTEXT-LABEL} \mid lb_i \in LB \wedge LB \subseteq CL \bullet \\ &\quad ((\exists_1 cl_i\text{: CONTEXT-LABEL} \mid cl_i \in CL \wedge \\ &\qquad l_i\text{: LABEL} \mid l_i \in L) \bullet \delta(l_i, cl_i) = lb_i \wedge \\ &\qquad\quad (\forall l_j\text{: LABEL} \mid l_j \in L \bullet \delta(l_j, cl_i) = lb_i \Rightarrow l_j = l_i) \wedge \\ &\qquad\qquad (\forall lb_i, lb_j\text{: CONTEXT-LABEL} \mid lb_i, lb_j \in LB \bullet \\ &\qquad\qquad\quad \delta(l_j, cl_i) = lb_i \wedge \delta(l_j, cl_i) = lb_j \Rightarrow lb_i = lb_j) \wedge \\ &\qquad\qquad\quad \forall S_i\text{: SCHEMA} \bullet (\exists l_j \in L \Rightarrow \exists_1 cl_j \in CL \wedge lb_j \in LB \bullet \\ &\qquad\qquad\qquad \delta(l_j, cl_j) = lb_j \wedge cl_j = lb_j))\end{aligned} \tag{4.12}$$

Theorem 4.1 Context labels, *cl*, in the *CTV* uniquely provide the semantics of every term, *l*, in the set of data sources, S_{LS}.

Proof Idea. Suppose there exists a term $l_i \in S_i$ such that the semantic meaning of $l_i \in CTV$ could be either cl_i or cl_j. Figure 4.4 shows that l_i could either be interpreted as

cl_i or cl_j, which is mapped to the context schema, CS_i. It follows that the *CTV* did not provide a precise semantic description of l_i. Two designers modeling Figure 4.4 may interpret l_i as either cl_i or cl_j, which means that the *CTV* cannot resolve multiple views of objects.

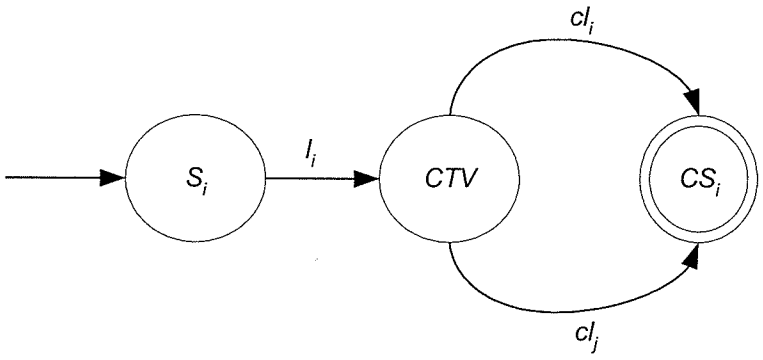

Figure 4.4: Transition from a Local Schema S_i to a Context Schema CS_i

Proof. Let $l_i \in S_i$, cl_i, $cl_j \in CTV$, and lb_i, $lb_j \in CS_i$ where l_i represents the ith term in the local schema, S_i; cl_i and cl_j respectively represent the ith and jth context labels in the common-term vocabulary, *CTV*; and lb_i and lb_j respectively represent the ith and jth context labels in the context schema, CS_i.

Suppose $\delta(l_i, cl_i) = lb_i \wedge \delta(l_i, cl_j) = lb_j$ (such that $lb_i \neq lb_j$) this implies that δ maps l_i to two different context labels. Recall,

$$\forall cl_i, cl_j: \text{CONTEX-LABEL} \mid (i, j: \mathbb{N} \wedge cl_i, cl_j \in CTV) \bullet cl_i = cl_j \Rightarrow i = j$$

(Context labels in the *CTV* are unique)

This means that the *CTV* provides a precise semantic description of terms, which means that δ can only map l_i to exactly one element in the *CTV*. Therefore, $cl_i = cl_j \vdash lb_i = lb_j$. Therefore, the function δ is injective since it can only map every term in the local schemas to precisely one term in the *CTV*. This contradicts the earlier assumption and establishes the proof. ∎

A central theme in extracting the semantics of data sources is a filter mechanism that recognizes a "context schema" so that all the participating schemas are guaranteed to be in the same format and without naming conflicts. We, therefore, provide a theoretical foundation to show that such a filter mechanism is capable of recognizing context

schemas in a finite number of steps. The following definitions are given to put the discussion in context.

Definition 4.10 A *filter mechanism* is an accepting device, which either accepts a local schema if it satisfies the accepting conditions, or rejects it otherwise. The accepting conditions are as follows:

1. Every local schema must have an entry point label that is associated with the root node.
2. Every element in the local schema (excluding the root) has at most one parent.
3. Every name (label) in the local schema is also in the *CTV*.
4. The entry point label of every object must be in the *CTV*. ■

If these conditions are satisfied, the filter mechanism accepts the local schema, and the accepted local schema then becomes a context schema. The filter mechanism parses the context schema with its associated metadata and mappings. Metadata information includes translation rules from local schema to context schema, mappings from source labels to context labels, and systems constraints. *Accept* is a boolean function that accepts a local schema as a context schema. The definition of the function *Accept* follows.

$$
\begin{aligned}
&\mathit{Accept}: \text{SCHEMA} \\
&\forall S: \text{SCHEMA} \bullet \exists\, \mathit{CTV} : \mathbb{P}\text{LABEL} \bullet \\
&\quad \mathit{Accept}(S) \Leftrightarrow (\exists r_0: \mathit{Root} \mid r_0 \in S \wedge \\
&\qquad \exists \lambda: \text{LABEL} \mid \lambda = \mathit{NodeLabel}(r_0) \wedge \lambda \in \mathit{CTV} \bullet \\
&\qquad\quad \forall v: \text{NODE} \mid v \in S \bullet (\exists_1 u: \text{NODE} \mid (u \neq v \wedge u \in S) \bullet \mathit{Parent}(v) = u) \wedge \\
&\qquad\qquad \forall l_i: \text{LABEL} \mid l_i \in S \bullet (\exists l_j : \text{LABEL} \mid l_j \in \mathit{CTV} \bullet l_i = l_j) \vee \\
&\qquad\qquad\quad \forall l_i \in S \mid (l_i \neq \lambda \wedge l_i \notin \mathit{CTV}) \bullet \mathit{Parent}(l_i) \in \mathit{CTV} \Rightarrow \\
&\qquad\qquad\qquad \mathit{Update\text{-}CTV}\,(cl_k, \mathit{CTV}) \bullet \delta(l_j, cl_i))
\end{aligned}
\tag{4.13}
$$

In the sections that follow, we design a two-phase integration model that isolates the input process from the integration process. The filter mechanism is used as an accepting device in the model.

4.4.1 A Two-Phase Integration Model

Unlike the techniques used in [Law01, GBMS99, Wac99], the approach in this thesis provides a formal framework to eliminate ambiguities in the informal definitions of the

semantic descriptions of the data sources. Figure 4.5 shows a two-phase model that leverages a syntactically simple and semantically rich semistructured data model, and the *CTV* to semantically describe each data source. In the model, data is represented with an edge-labelled graph, called *Del-G*, where information only resides in the labels. Annotation of terms from the data sources using the *CTV* then becomes a straightforward term lookup. Through comparison of the graphs, similarities between the data sources can be estimated. Phase 1 acquires the semantics of the data sources and Phase 2 dynamically identifies and analyses corresponding assertions to generate integration rules.

In Phase 1, the primary concern is to present the context schemas in a platform independent format and to eliminate multiple views of objects. The main goal of this phase is to support the autonomy of data sources so that local data sources can continue to support local transactions regardless of their participation in the integration system. The elimination of multiple views of objects in the input schemas is crucial to the integration process because it facilitates the dynamic identification of correspondence assertions. In particular, Sheth and Larson [SL90] observe that "unless the schemas are represented in the same model, analyzing and comparing their schema objects is extremely difficult." This thesis uses *Del-G* as a canonical data model [SL90] for this purpose because of its simplicity and its natural support for representing conceptualizations.

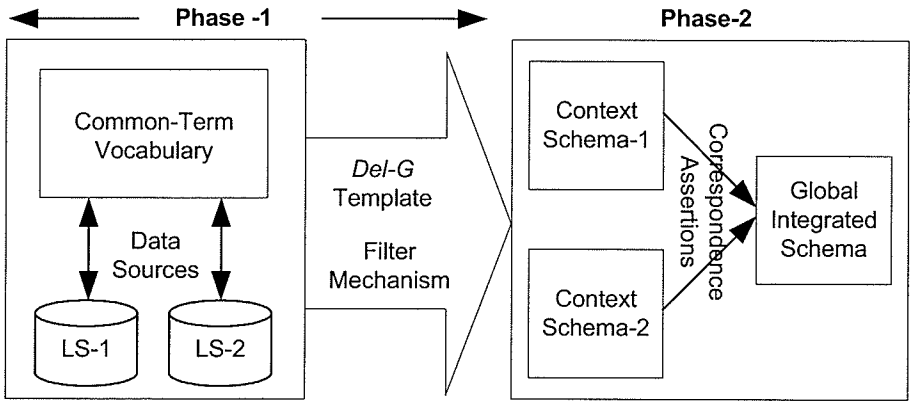

Figure 4.5: A Two-Phase Integration Model

The highlights of Phase 1 are:

a) A global terminology (the *CTV*) provides the primitive terms of the domain, which are combined with operators to build complex labels.
b) The concepts used in different sources are annotated with labels describing their semantics.
c) A semantic description of each participating data source is acquired (context schemas).

To acquire the semantics of the data sources (Phase 1) involves a semi-automatic process that begins with the annotation of terms in the data sources. The annotation is carried out in the form of a term lookup by the DBA of each participating data source who wants to provide better access to information in his/her data source. The DBA uses the *CTV* to associate each term in the local schema to a context label in the *CTV*. This is a one-time process. Context labels combine terms according to the structure and needs of the data sources to form a context schema. For every source label l_i in data source S_i, there exists a corresponding context label cl_i in context schema CS_i, where $cl_i \in CS_i \Rightarrow cl_i \in CTV$. Accordingly, if $l_i \in S_i$, $l_j \in S_j$, …, $l_n \in S_n$, where l_i, l_j, …, $l_n \in L$ (L is the set of source labels) and S_i, S_j, …, $S_n \in S_{\text{SET}}$ ($S_{\text{SET}} = \cup S_k$), then there exist corresponding $cl_i \in CS_i$, $cl_j \in CS_j$, …, $cl_n \in CS_n$, where cl_i, cl_j, …, $cl_n \in CL$ (CL is the set of context labels and $CL \subseteq CTV$) and CS_i, CS_j, …, $CS_n \in CS_{\text{SET}}$ ($CS_{\text{SET}} = \cup CS_k$).

There is assumed to be a software agent, called the *Del-G Template Creator* that provides a "template" to the data sources so that context labels corresponding to terms in each data source can be represented as context schemas. In other words, the *Del-G* template creator uses a filter mechanism to encode local schemas in a machine-readable format. A *filter mechanism*, which is a module in the *Del-G Template*, ensures the conformity of terms to the *Del-G* structure. Recall that a context schema is a well-formed schema. Both the *CTV* and *Del-G* template are made available to the data sources via the Internet. Observe that the process of acquiring data semantics from each data source is performed independently. Notice also that this model does not predefine context schemas but dynamically generates and maintains context schemas during the integration process. This dynamic nature of context schemas allows the revision and extension of context schemas with respect to newly added sources, and hence facilitates scalability.

4.4.2 Theoretical Foundations of the Semantic Description

Recall that a filter mechanism that accepts a context schema is an accepting device. This implies that it is possible to construct a finite state automaton that accepts a context schema since it is known [Sip97] that a finite state automaton can act as an accepting device under certain conditions. First, we define a finite state automaton for S^{Wf}.

Definition 4.11 A *finite state automaton* (FSA) for a well-formed schema S^{Wf} is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

1. $Q \mid Q$: NODE is a finite set of states;
2. $\Sigma \mid \Sigma$: LABEL $\wedge \Sigma \subseteq$ *CTV* is a finite set called the alphabet;
3. δ: $Q \times \Sigma \rightarrow Q$ is a transition function (which defines the transition rules);
4. $q_0 \in Q \mid q_0 \in$ *Root* is a start state
5. $F \subseteq Q \mid F \subseteq$ *Atomic-Nodes* is the set of final states (accepting states).

Juxtaposing Definition 4.11 with Definition 4.10, leads to the following deductions that facilitate the construction of an FSA from a well-formed schema.

a) Point 1 in Definition 4.10 defines the start state.
b) Point 2 in Definition 4.10 specifies that the finite state automaton is deterministic.
c) Point 3 in Definition 4.10 defines a finite set of alphabets, Σ.
d) Point 4 in Definition 4.10 states that $\lambda \in \Sigma$.

■

Theorem 4.2 Let $\Im$ be a finite state automaton. $\Im$ accepts a *well-formed Schema* S^{Wf}, if and only if there exists a set of path labels, *PL* that completely represents S^{Wf}.

Proof. To simplify the proof we assume, without loss of generality that $\Sigma \subseteq CTV = \{a, b\}$, and L is a regular language over Σ. Let R be a regular expression associated with L. A well-formed Schema, S^{Wf}, in *Del-G* is completely represented by a set of data paths *DP*, and for each data path dp_i, there is a corresponding *Path-Label*, pl_j, such that pl_j uniquely connects the root node, r_0, to a given node u, where $u \in$ *Atomic-Nodes*. The proof follows from the inductive definition of regular expressions.

Base Case

Suppose R is either ε, a, or b, then, L is either $\{\varepsilon\}$, $\{a\}$, or $\{b\}$. In this case, L is FSA-acceptable because it is finite. Observe that there can be a *Del-G* represented with only one node, r_0, and the entry-point label, λ (where $\lambda = \varepsilon$, a, or b), such that $dp_i = \lambda.r_0$ and $pl_j = \lambda$.

Inductive Case

Suppose R is of the form $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions, then both $L(R_1)$ and $L(R_2)$ are regular and are acceptable by an FSA. $L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$, so $L(R_1 \cup R_2)$ is acceptable by an FSA.

Suppose R is of the form $(R_1.R_2)$, where $R_1$and R_2 are regular expressions, then both $L(R_1)$ and $L(R_2)$ are regular languages and are acceptable by an FSA. $L(R_1.R_2) = L(R_1).L(R_2)$, so $L(R_1).L(R_2)$ is acceptable by an FSA.

Finally, suppose R is of the form (R_1^*), where R_1 is a regular expression, then $L(R_1)$ is regular by definition and are acceptable by an FSA. But, $L(R_1^*) = L^*R_1$, so $L(R_1^*)$ is acceptable by an FSA. ■

Example 4.3 To illustrate the proof of Theorem 4.2, we construct an FSA for the *Del-G* schema of Figure 4.2. Figure 4.6 shows the state diagram of the *Del-G* schema of Figure 4.2 and Table 4.3 represents the transition function (δ) for the FSA of Figure 4.6. It has six states each representing a node in the *Del-G* schema. When the automaton receives the *Del-G* represented as a set of data paths *DP*, it processes each data path dp_i and produces an output. The output is either "*accept*" or "*reject*".

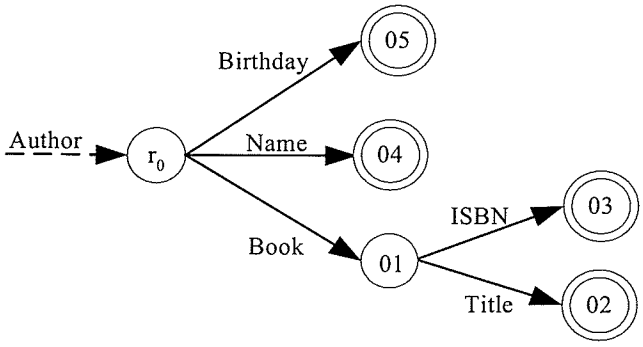

Figure 4.6: Finite Automaton of a *Del-G* Schema

Table 4.3: A Transition Table of Figure 4.6

	Book	Title	ISBN	Name	Birthday
r_0	01			04	05
01		02	03		
02					
03					
04					
05					

The automaton rejects the entire set of data paths if any data path $dp_i \in DP$ is rejected. Recall that there exists a pseudo edge, e_0, pointing from nowhere to the root node r_0 in a *Del-G* schema. This pseudo edge for Example 4.3 is associated with the *entry-point label*, "*Author*". Notice that a pseudo arrow pointing from nowhere to a state, r_0, in Figure 4.6 indicates the start state. The *entry-point label*, "*Author*" is attached to this pseudo arrow.

Let $M = (Q, \Sigma, \delta, r_0, F)$, where

1. $Q = \{r_0, 01, 02, 03, 04, 05\}$;
2. $\Sigma = \{$Book, Title, ISBN, Name, Birthday$\}$;
3. δ is described as shown in Table 4.3;
4. $r_0 \in Q$ is the start state; and
5. $F = \{02, 03, 04, 05\} \subseteq Q$.

If A is the set of all strings that machine M accepts, then, A is the language of machine M and is written as $L(M) = A$. M is said to accept A.

From Theorem 4.2, we deduce that the filter mechanism accepts a context schema. Another interesting task is to determine whether the filter mechanism accepts a context schema in a finite number of computational steps. Theorem 4.3 answers this question [AE05].

To answer this question, we define a Turing machine, M, in the context of a *Del-G* schema.

Definition 4.12 A *Turing machine* is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are finite sets, and

1. $Q \mid Q$: NODE is the set of states;
2. $\Sigma \mid \Sigma$: LABEL $\wedge \Sigma \subseteq CTV$ is the input alphabet not containing the special blank symbol $\sqcup$;
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subset \Gamma$,
4. δ: $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function,
5. $q_0 \mid q_0 \in Root \wedge Root \subseteq Q$ is a start state,
6. $q_{accept} \mid q_{accept} \in Atomic\text{-}Nodes \wedge Atomic\text{-}Nodes \subseteq Q$ is the accept state, and
7. $q_{reject} \mid q_{reject} \notin Atomic\text{-}Nodes$ is the reject state, where $q_{accept} \neq q_{reject}$. ■

A Turing machine gives a precise definition of an algorithm or a mechanical procedure. It is a simple machine that can decide if a computational problem is solvable in a finite number of steps.

Theorem 4.3 The process of recognizing a well-formed schema by the *filter mechanism* is decidable.

Proof Idea. Recall from Theorem 4.2 that an FSA accepts S^{Wf}. Also, from Definition 4.10, it can be deduced that a filter mechanism can be represented as a finite state automaton. The problem reduces to: Given an S^{Wf}, find a Turing machine, TM M, that accepts S^{Wf} in a finite number of computational steps. The need is to present a TM M that decides S^{Wf}.

M = On input $\langle B, w \rangle$, where B is an FSA and $w \mid w$: PATH-LABEL is a string:

1. Simulate B on input w.
2. If the simulation ends in an accept state $u \mid u \in Atomic\text{-}Nodes$, then accept; otherwise reject if it ends in a non-accepting state $u \mid u \notin Atomic\text{-}Nodes$.

Proof. First, we examine the input $\langle B, w \rangle$, which represents an FSA B together with a string w. B can be represented as a five-tuple $(Q, \Sigma, \delta, q_0, F)$. When M receives its input, it first checks whether the input properly represents an FSA B and a string w. If the input does not properly represent a FSA B and a string w, M rejects $\langle B, w \rangle$. Otherwise, M carries out the check in a direct way, keeping track of B's current state and B's current position in the input w by writing the information down on its

tape. Initially, *B*'s current state is $q_0 \in Root$ and *B*'s current input is the entry point label λ. The transition function δ updates states Q: NODE and the position of the input tape, *w*. When *M* finishes processing the last symbol of *w*, *M* accepts the input if *B* is in the accepting state; otherwise *M* rejects the input if *B* is in a non-accepting state.

■

Example 4.4 Consider a very simple example to illustrate how to obtain a context schema, CS_i, from a source schema, S_i. *Student-Books Inc.* and *Books-Warehouse Ltd.* are two online bookstores with different schemas.

- *Student-Books Inc.* represents its information in schema S_1 as follows:

  S_1 = *SB-Books* [*ISBN*, *Book-Title*, *Book-Author* [*Number*, *Name*], *Delivery* [*Method*, *Date*], *Price*]

- *Books-Warehouse Ltd.* represents its information in schema S_2 as follows:

  S_2 = *Books-BW* [*ISBN*, *Title*, *Author* [*No, Name*], *Publisher* [*Identity*, *Name*], *Qty*, *Price*]

In this representation, square brackets again enclose children (sub-objects) of the object directly before the left square bracket and comas separate siblings of a given object. For example, [*ISBN*, *Book-Title*, *Book-Author*, *Delivery*, *Price*] are children of *SB-Books* $\in S_1$. Recall, every term in the source schema, S_i, has a label *l* that identifies the element. However, source labels are not standardized, and could have different interpretations by different designers (multiple views). To solve the problem of multiple views of objects, and extract the semantics of terms in the source schemas, there exists a unique context label $cl \in CTV$ for every source label $l \in S_i$ (Theorem 4.1).

For every data source, there is a lookup by the DBA to map terms in the data sources to corresponding terms in the *CTV*. The terms lookup starts with the entry-point label, λ, and progresses to its descendants in a parent-child fashion until all terms in the source schema are mapped to the *CTV*. The parent-child approach of terms lookup ensures that existing relationships are maintained. Table 4.4 shows the corresponding context labels for source labels in source the schema S_1, while Table 4.5 shows the corresponding context labels for source labels in the source schema S_2.

Table 4.4: Substituting Context Labels for Source Labels in S_1

Source Schema	Source Labels (*l*)	Context Schema	Context Labels (*cl*)
S_1	SB-Books	CS_1	Books
S_1	SB-Books [ISBN]	CS_1	Books [ISBN]
S_1	SB-Books [Book-Title]	CS_1	Books [Title]
S_1	SB-Books [Book-Author]	CS_1	Books [Author]
S_1	SB-Books [Book-Author [Number]]	CS_1	Books [Author [ID]]
S_1	SB-Books [Book-Author [Name]]	CS_1	Books [Author [Name]]
S_1	SB-Books [Delivery]	CS_1	Books [Delivery]
S_1	SB-Books [Delivery [Method]]	CS_1	Books [Delivery [Mode]]
S_1	SB-Books [Delivery [Date]]	CS_1	Books [Delivery [Date]]
S_1	SB-Books [Price]	CS_1	Books [Price]

CS_1 is the corresponding context schema, for the source schema, S_1.

CS_1 = *Books* [*ISBN*, *Title*, *Author* [*ID*, *Name*], *Delivery* [*Mode*, *Date*], *Price*]]

Table 4.5: Substituting Context Labels for Source Labels in S_2

Source Schema	Source Labels (*l*)	Context Schema	Context Labels (*cl*)
S_2	Books-BW	CS_2	Books
S_2	Books-BW [ISBN]	CS_2	Books [ISBN]
S_2	Books-BW [Title]	CS_2	Books [Title]
S_2	Books-BW [Author]	CS_2	Books [Author]
S_2	Books-BW [Author [No]]	CS_2	Books [Author [ID]]
S_2	Books-BW [Author [Name]]	CS_2	Books [Author [Name]]
S_2	Books-BW [Publisher]	CS_2	Books [Publisher]
S_2	Books-BW [Publisher [Identity]]	CS_2	Books [Publisher [ID]]
S_2	Books-BW [Publisher [Name]]	CS_2	Books [Publisher [Name]]
S_2	Books-BW [Qty]	CS_2	Books [Quantity]
S_2	Books-BW [Price]	CS_2	Books [Price]

CS_2 is the corresponding context schema, for the source schema, S_2.

CS_2 = *Books* [*ISBN*, *Title*, *Author* [*ID*, *Name*], *Publisher* [*ID*, *Name*], *Quantity*, *Price*]]

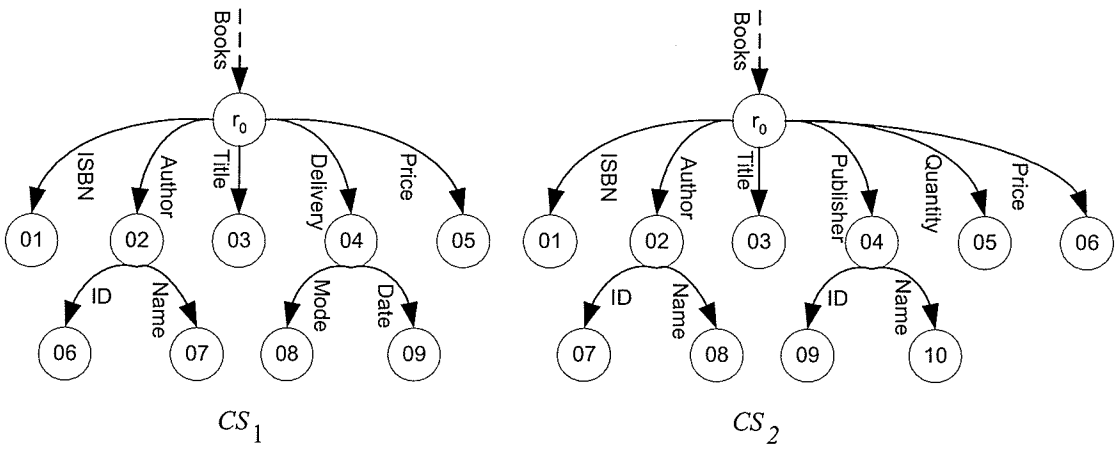

Figure 4.7: A *Del-G* Representation of Context Schemas

These two schemas S_1 and S_2 clearly represent the same universe of discourse, even though they are structurally different. Figure 4.7 shows the two context schemas, CS_1 and CS_2, corresponding to S_1 and S_2, respectively. The context schemas are represented using *Del-G*. Notice that the terms in the context schemas are drawn from the *CTV*. In the second phase of the integration model, the two context schemas are used as input to a graph-based integration algorithm, which dynamically identifies and analyses assertions from the schemas. A detailed description of Phase 2 processing is given in Chapters 5 and 6.

Chapter 5

Identification and Analysis of Correspondence Assertions

This chapter consists of three main sections. Section 5.1 specifies correspondence assertions, providing unambiguous definitions of the different assertions. This thesis identifies and specifies seven correspondence assertions [AE04c] used to integrate data sources. We make explicit the content of data [AE04h] and adopt a fine-grained comparison of objects[38] to identify more assertions than existing systems, such as [Che00, SP94]. This is more than the number identified by [Che00, SP94] because the semantics of data sources have been made explicit and the adoption of a fine-grained comparison of objects. Section 5.2 examines integration constraints [AE05]. This part provides the conditions under which data objects should be integrated. The discussion in this part paves the way to analyse correspondence assertions. Section 5.3 deals with the analysis of correspondence assertions [AE04d, EA04]. This part involves analysing assertions to generate integration rules that produce an integrated schema.

A correspondence assertion is a declarative statement that shows the relationship between objects in different data sources at various levels of perception. This definition differs from [LNE89] which only measures relationships at the same level of perception. This definition is relaxed to allow for the comparison of object types at different levels of

[38] Comparing objects at different levels of perceptions (e.g. object to object, sub-object to object, attribute to object, attribute to sub-object, sub-object to sub-object, etc.).

perception provided the elements have the same names (context labels). This relaxation allows us to identify more assertions. Identifying assertions involves mapping (*f*) elements from one data source, CS_i, to another data source, CS_j, using an intermediary data source, CS_k, such that CS_k relates to both CS_i and CS_j (f: $CS_i \times CS_j \rightarrow CS_k$). Mapping between heterogeneous data sources is key to a successful integration system.

Let L_1 be the context label of object O_1 from context schema CS_1 and let L_2 be the context label of object O_2 from context schema CS_2. O_1 has children O_{11}, O_{12}, ..., O_{1n} (sub-objects or attributes) with corresponding context labels ${L_1}^{L1}{}_1$, ${L_1}^{L1}{}_2$, ..., ${L_1}^{L1}{}_n$. O_2 has children O_{21}, O_{22}, ..., O_{2m} (sub-objects or attributes) with corresponding context labels ${L_2}^{L2}{}_1$, ${L_2}^{L2}{}_2$, ..., ${L_2}^{L2}{}_n$. A relationship exists between O_1 and O_2 if there exist context labels ${L_1}^{p}{}_i \in CS_1$ and ${L_2}^{q}{}_j \in CS_2$ such that ${L_1}^{p}{}_i = {L_2}^{q}{}_j$. For example, ${L_1}^{p}{}_i$ represents the *i*th element whose parent is $p \in CS_1$ (where p: LABEL and i: $\mathbb{N}_1$). L_1 represents an element without a parent in CS_1. In *Del-G*, there is only one distinguished element without a parent, the entry-point label.

Letting CS_{SET} be the set of context schemas, then a formal definition of dynamic correspondence assertion and specification of correspondence assertions follows.

Definition 5.1 A *dynamic correspondence assertion* (Ψ) is a declarative statement generated by the integration system that shows the relationship between ${L_1}^{p}{}_i \in CS_1$ and ${L_2}^{p}{}_j \in CS_2$, for all $CS_k \in CS_{SET}$ (where $i, j, k : \mathbb{N}_1$) at various levels of perception (e.g., objects and sub-objects, objects and attributes, and so on). ■

5.1 Assertions Specification

Recall, objects with the same context labels represent the same real world state, because every context label is drawn from a common-term vocabulary. Thus, erroneous choices regarding names that hitherto existed in earlier systems no longer exist. However, structural conflicts still exist since different designers may model a particular object differently due to variations in the designers' perceptions and modeling constructs. Therefore, as in [LNE89], corresponding objects must have the same names.

5.1.1 Identity Assertion

O_1 is said to be identical to O_2 if and only if $L_1 = L_2$ and $L_1{}^p{}_i = L_2{}^q{}_j$, ($1 \le i \le n$ and $1 \le j \le m$ such that $n = m$). Informally, O_1 and O_2 are precisely the same because the designers used the same constructs and perceptions, which means that ($L_1 = L_2 \wedge$ (*Edge-Child*(L_1) = *Edge-Child*(L_2))). The formal definition of the (binary) relation *Identity* corresponding to the identical assertion follows.

$$
\begin{aligned}
&\text{- } \textit{Identity} \text{ -: LABEL} \leftrightarrow \text{LABEL} \\
&\forall\, L_1, L_2\text{: LABEL} \bullet (L_1 \textit{ Identity } L_2 \Leftrightarrow \\
&\quad (\exists\, CS_1, CS_2\text{: SCHEMA} \mid L_1 \in CS_1 \wedge L_2 \in CS_2 \bullet L_1 = L_2 \wedge \\
&\qquad (\forall i\text{: } \mathbb{N}_1 \bullet \exists\, L_1{}^p{}_i \in \textit{Edge-Child}(L_1) \Rightarrow \\
&\qquad\quad (\exists j\text{: } \mathbb{N}_1 \bullet L_2{}^q{}_j = L_1{}^p{}_i \wedge L_2{}^q{}_j \in \textit{Edge-Child}(L_2))) \wedge \\
&\qquad\quad (\forall j\text{: } \mathbb{N}_1 \bullet L_2{}^q{}_j \in \textit{Edge-Child}(L_2) \Rightarrow \\
&\qquad\qquad (\exists i\text{: } \mathbb{N}_1 \bullet L_1{}^p{}_i = L_2{}^q{}_j \wedge L_1{}^p{}_i \in \textit{Edge-Child}(L_1))))
\end{aligned}
$$

5.1.2 Equivalence Assertion

O_1 is said to be equivalent to O_2 if $L_1 = L_2$ and $L_1{}^p{}_i = L_2{}^q{}_j$, ($1 \le i \le n$ and $1 \le j \le m$ such that $n \ne m$). Informally, O_1 and O_2 are not precisely the same because equivalent and inexact modeling constructs and perceptions have been used, which means that ($L_1 = L_2 \wedge$ ((*Edge-Child*(L_1) $\supseteq$ *Edge-Child*(L_2)) $\vee$ (*Edge-Child*(L_1) $\subseteq$ *Edge-Child*(L_2)))). Object L_1 consists of all the values of object L_2, or object L_2 consists of all the values of L_1. The formal definition of the (binary) relation *Equivalence* corresponding to the equivalent assertion follows:

$$
\begin{aligned}
&\text{- } \textit{Equivalence} \text{ -: LABEL} \leftrightarrow \text{LABEL} \\
&\forall L_1, L_2\text{: } \textit{Label} \bullet (L_1 \textit{ Equivalence } L_2 \Leftrightarrow \\
&\quad (\exists\, CS_1, CS_2\text{: SCHEMA} \mid L_1 \in CS_1 \wedge L_2 \in CS_2 \bullet L_1 = L_2 \wedge \\
&\qquad (\forall i\text{: } \mathbb{N}_1 \bullet L_1{}^p{}_i \in \textit{Edge-Child}(L_1) \Rightarrow \\
&\qquad\quad (\exists l\text{: } \mathbb{N}_1 \bullet L_1{}^p{}_i = L_2{}^q{}_j \wedge L_2{}^q{}_j \in \textit{Edge-Child}(L_2))) \vee \\
&\qquad\quad (\forall l\text{: } \mathbb{N}_1 \bullet L_2{}^q{}_j \in \textit{Edge-Child}(L_2) \Rightarrow \\
&\qquad\qquad (\exists i\text{: } \mathbb{N}_1 \bullet L_2{}^q{}_j = L_1{}^p{}_i \wedge L_1{}^p{}_i \in \textit{Edge-Child}(L_1))))
\end{aligned}
$$

5.1.3 Intersection Assertion

O_1 and O_2 have an intersection assertion if $L_1 = L_2$ and *Edge-Child*(L_1) $\cap$ *Edge-Child*(L_2) $\neq \varnothing$. Informally, O_1 and O_2 are perceived the same way but the modeling constructs differ slightly, which means that ($L_1 = L_2 \wedge$ (*Edge-Child*(L_1) $\cap$ *Edge-Child*(L_2) $\neq \varnothing$)). Object L_1 consists of some values of object L_2, and object L_2 consists of some values of L_1. The formal definition of the (binary) relation *Intersection* corresponding to the intersection assertion follows.

- *Intersection* -: LABEL $\leftrightarrow$ LABEL
$\forall L_1, L_2$: LABEL • (L_1 *Intersection* $L_2 \Leftrightarrow$
($\exists$ CS_1, CS_2: SCHEMA | $L_1 \in CS_1 \wedge L_2 \in CS_2$ • $L_1 = L_2 \wedge$
($\exists a, b$: *Edge-Child*(L_1) $\wedge$ $\exists b, c$: *Edge-Child*(L_2) •
$a \notin$ *Edge-Child*(L_2) $\wedge$ $c \notin$ *Edge-Child*(L_1)) $\Rightarrow$
Edge-Child(L_1) $\cap$ *Edge-Child*(L_2) $\neq \varnothing$)))

5.1.4 Compatibility Assertion

O_1 is said to be compatible with O_2 if $L_1 = L_2$, and *Edge-Child*(L_1) $\cap$ *Edge-Child*(L_2) $= \varnothing$. Informally, O_1 and O_2 are neither identical nor equivalent, but the modeling constructs are not contradictory. For the compatible assertion, given by $L_1 = L_2 \wedge ({L_1}^p{}_i \cap {L_2}^q{}_j = \varnothing)$ (where, $1 \leq i \leq n$ and $1 \leq j \leq m$), L_1 and L_2 represent the same object since they have the same name, but the attributes of ${L_1}^p{}_i$ in CS_1 are different from the attributes of ${L_2}^q{}_j$ in CS_2. A formal definition of the (binary) relation *Compatibility* corresponding to the compatible assertion follows:

- *Compatibility* -: LABEL $\leftrightarrow$ LABEL
$\forall L_1, L_2$: LABEL • (L_1 *Compatibility* $L_2 \Leftrightarrow$
($\exists$ CS_1, CS_2: SCHEMA | $L_1 \in CS_1 \wedge L_2 \in CS_2$ • $L_1 = L_2 \wedge$
($\forall a$: *Edge-Child*(L_1) | $a \notin$ *Edge-Child*(L_2) $\wedge$
$\forall b$: *Edge-Child*(L_2) | $b \notin$ *Edge-Child*(L_1) •
Edge-Child(L_1) $\cap$ *Edge-Child*(L_2) $= \varnothing$)))

5.1.5 Inclusion Assertion

O_1 and O_2 have an inclusion assertion, if $L_1 \neq L_2$ and ($L_1 \in$ *Descendant*(L_2) $\vee$ $L_2 \in$ *Descendant*(L_1)). Informally, the inclusion assertion states that O_1 and O_2 have different

perceptions but related modeling constructs. L_1 is either a sub-object or attribute of L_2, or L_2 is a sub-object or attribute of L_1. Suppose L_1 is a sub-object of L_2 ($L_1 \in$ *Descendant*(L_2)), and all the attributes of L_1 are also attributes of the corresponding sub-object of L_2, then L_2 properly includes L_1. A formal definition of the (binary) relation *Inclusion* for the inclusion assertion follows:

$$
\begin{aligned}
&\text{- } \textit{Inclusion} \text{ -: LABEL} \leftrightarrow \text{LABEL} \\
&\forall L_1, L_2: \text{LABEL} \bullet (L_1 \textit{ Inclusion } L_2 \Leftrightarrow \\
&\quad (\exists\, CS_1, CS_2: \text{SCHEMA} \mid L_1 \in CS_1 \wedge L_2 \in CS_2 \bullet L_1 \neq L_2 \wedge \\
&\qquad (\exists a: \textit{Descendant}(L_1) \bullet a = L_2 \wedge\ (\forall b: \text{LABEL} \mid b \in \textit{Edge-Child}(L_2) \bullet \\
&\qquad\quad (\exists i: \mathbb{N}_1 \mid a_i \in \textit{Edge-Child}(a)\ \bullet a_i = b) \Rightarrow \\
&\qquad\quad L_2 \in \textit{Descendant}(L_1) \wedge (\textit{Edge-Child}(a) \supseteq \textit{Edge-Child}(L_2))) \vee \\
&\qquad\quad (\exists b: \textit{Descendant}(L_2) \bullet b = L_1 \wedge \\
&\qquad\qquad (\forall a: \text{LABEL} \mid a \in \textit{Edge-Child}(L_1) \bullet \\
&\qquad\qquad\quad (\exists j: \mathbb{N}_1 \mid b_j \in \textit{Edge-Child}(b))\ \bullet b_j = a) \Rightarrow \\
&\qquad\qquad\quad L_1 \in \textit{Descendant}(L_2) \wedge \textit{Edge-Child}(L_1) \subseteq \textit{Child}(b)))))
\end{aligned}
$$

Included-Intersection Assertion

O_1 and O_2 have an included-intersection assertion if $L_1 \neq L_2 \wedge (\exists b$: *Descendant*($L_2$) | $L_1 = b \wedge$ *Edge-Child*(L_1) $\cap$ *Edge-Child*(b) $\neq \varnothing$) $\vee$ (a: *Descendant*(L_1) | $L_2 = a \wedge$ *Edge-Child*(L_2) $\cap$ *Edge-Child*(a) $\neq \varnothing$)). Informally, the included-intersection assertion states that O_1 and O_2 are perceived differently, and also the modeling constructs differ. L_1 is either a sub-object or attribute of L_2, or L_2 is a sub-object or attribute of L_1. Suppose L_1 is a sub-object of L_2, and the intersection of the attributes of L_1 and the attributes of the corresponding sub-object of L_2 is not empty, then there exists an included-intersection assertion. A formal definition of the (binary) relation *Included-Intersection* for the included-intersection assertion follows:

$$
\begin{aligned}
&\text{- } \textit{Included-Intersection} \text{ -: LABEL} \leftrightarrow \text{LABEL} \\
&\forall L_1, L_2: \text{LABEL} \bullet (L_1 \textit{ Included-Inclusion } L_2 \Leftrightarrow \\
&\quad (\exists\, CS_1, CS_2: \text{SCHEMA} \mid L_1 \in CS_1 \wedge L_2 \in CS_2 \bullet L_1 \neq L_2 \wedge \\
&\qquad (\exists a: \textit{Descendant}(L_1) \bullet a = L_2\ \wedge \\
&\qquad\quad (\forall b: \textit{Edge-Child}(L_2) \bullet (\exists i: \mathbb{N}_1 \mid a_i \in \textit{Edge-Child}(a) \bullet a_i = b) \Rightarrow \\
&\qquad\quad L_2 \in \textit{Descendant}(L_1) \wedge (\textit{Edge-Child}(a) \cap \textit{Edge-Child}(L_2) \neq \varnothing))) \vee \\
&\qquad\quad (\exists b: \textit{Descendant}(L_2) \bullet b = L_1\ \wedge \\
&\qquad\qquad (\forall a: \textit{Edge-Child}(L_1) \bullet (\exists j: \mathbb{N}_1 \mid b_j \in \textit{Edge-Child}(b) \bullet b_j = a) \Rightarrow \\
&\qquad\qquad\quad L_1 \in \textit{Descendant}(L_2) \wedge \textit{Edge-Child}(L_1) \cap \textit{Edge-Child}(b) \neq \varnothing))))
\end{aligned}
$$

Included-Compatibility Assertion

O_1 and O_2 have an included-compatibility assertion if $L_1 \neq L_2 \wedge (\exists b$: *Descendant*$(L_2) \mid L_1 = b \wedge$ *Edge-Child*$(L_1) \cap$ *Edge-Child*$(b) = \varnothing) \vee (a$: *Descendant*$(L_1) \mid L_2 = a \wedge$ *Edge-Child*$(L_2) \cap$ *Edge-Child*$(a) = \varnothing))$. Informally, the included-compatibility assertion states that O_1 and O_2 are neither perceived the same way nor have equivalent modeling constructs, but are not entirely unrelated. L_1 is either a sub-object or attribute of L_2, or L_2 is a sub-object or attribute of L_1. Suppose L_1 is a sub-object of L_2 and the intersection of the attributes of L_1 and the attributes of the corresponding sub-object of L_2 is empty, then there exists an included-compatibility assertion. A formal definition of the (binary) relation *Included-Compatibility* follows:

$$
\begin{aligned}
&\text{- } \textit{Included-Compatible} \text{ -: LABEL} \leftrightarrow \text{LABEL} \\
&\forall L_1, L_2\text{: LABEL} \bullet (L_1 \textit{ Included-Compatible } L_2 \Leftrightarrow \\
&\quad (\exists\, CS_1, CS_2\text{: SCHEMA} \mid L_1 \in CS_1 \wedge L_2 \in CS_2 \bullet L_1 \neq L_2 \wedge \\
&\quad (\exists a\text{: } \textit{Descendant}(L_1) \bullet a = L_2 \ \wedge \\
&\qquad (\forall b\text{: LABEL} \mid b \in \textit{Edge-Child}(L_2) \bullet b \notin \textit{Edge-Child}(a) \wedge \\
&\qquad\quad (\forall i\text{: } \mathbb{N}_1 \mid a_i \in \textit{Edge-Child}(a) \bullet a_i \notin \textit{Edge-Child}(L_2) \Rightarrow \\
&\qquad\qquad L_2 \in \textit{Descendant}(L_1) \wedge (\textit{Edge-Child}(a) \cap \textit{Edge-Child}(L_2) = \varnothing)))) \ \vee \\
&\qquad\quad (\exists b\text{: } \textit{Descendant}(L_2) \bullet b = L_1 \ \wedge \\
&\qquad\qquad (\forall a\text{: LABEL} \mid a \in \textit{Edge-Child}(L_1) \bullet a \notin \textit{Edge-Child}(L_2) \wedge \\
&\qquad\qquad\quad (\forall j\text{: } \mathbb{N}_1 \mid b_j \in \textit{Edge-Child}(b) \bullet b_j \notin \textit{Edge-Child}(L_1) \Rightarrow \\
&\qquad\qquad\qquad L_1 \in \textit{Descendant}(L_2) \wedge (\textit{Edge-Child}(b) \cap \textit{Edge-Child}(L_1) = \varnothing))))))
\end{aligned}
$$

5.1.6 Mutual-Inclusion Assertion

O_1 and O_2 have a mutual-inclusion assertion if $L_1 \neq L_2$ and ($L_1 \in$ *Descendant*$(L_2) \wedge L_2 \in$ *Descendant*(L_1)). Informally, the mutual-inclusion assertion states that O_1 and O_2 are not perceived the same way but have equivalent modeling constructs. L_1 is a descendant of L_2, and L_2 is also a descendant of L_1 (a detailed explanation of this assertion is provided in Section 5.3.4). A formal definition of the (binary) relation *Mutual-Inclusion* for mutual-inclusion assertion follows:

$$
\begin{aligned}
&\text{- } \textit{Mutual-Inclusion} \text{ -: LABEL} \leftrightarrow \text{LABEL} \\
&\forall L_1, L_2\text{: LABEL} \bullet (L_1 \textit{ Mutual-Inclusion } L_2 \Leftrightarrow \\
&\quad (\exists\, CS_1, CS_2\text{: SCHEMA} \mid L_1 \in CS_1 \wedge L_2 \in CS_2 \bullet L_1 \neq L_2 \wedge \\
&\quad L_1 \in \textit{Descendant}(L_2) \ \wedge L_2 \in \textit{Descendant}(L_1)))
\end{aligned}
$$

5.1.7 Exclusion Assertion

O_1 and O_2 have an exclusion assertion if $L_1 \neq L_2$ and ($L_1 \notin$ *Descendant*(L_2) $\wedge$ $L_2 \notin$ *Descendant*(L_1)). Informally, the exclusion assertion states that O_1 and O_2 neither agree in constructs, nor in the designers' perception. L_1 and L_2 are not the same, and they are not attributes of each other. However, L_1 and L_2 could still have a common ancestor. The formal definition of the (binary) relation *Exclusion* for the exclusion assertion follows:

$$
\begin{aligned}
&\text{- } \textit{Exclusion} \text{ -: LABEL} \leftrightarrow \text{LABEL} \\
&\forall L_1, L_2 \text{: LABEL} \bullet (L_1 \textit{ Exclusion } L_2 \Leftrightarrow \\
&\quad (\exists\, CS_1, CS_2 \text{: SCHEMA} \mid L_1 \in CS_1 \wedge L_2 \in CS_2 \bullet L_1 \neq L_2 \wedge \\
&\qquad (L_1 \notin \textit{Descendant}(L_2) \wedge L_2 \notin \textit{Descendant}(L_1)) \Rightarrow \\
&\qquad\quad L_1 \cap L_2 = \varnothing)))
\end{aligned}
$$

Notice that the exclusion assertion can only occur if the entry-point label of CS_1 is not related to any element in CS_2 and the entry-point label of CS_2 is not related to any element in CS_1. The sections that follow formalize the process of mapping elements of two context schemas and develop suitable algorithms to produce the mapping results. First, integration constraints are examined to put the discussion in context.

5.2 Integration Constraints

To simplify the correspondence rules generation process and to improve the quality of meta-information provided to the integration system, there is a need to explain certain constraints [AE05]. These constraints influence the analysis of correspondence assertions and consequently, the generation of correspondence rules by the integration algorithm (the integration algorithm will be discussed in Chapter Six).

To enable us to define these constraints, first, we define the function *Equal*, which is a Boolean function that shows if two context labels are the same. Let L_{EP} be a set of entry-point labels. A pair of elements ($L_1{}^p{}_i$, $L_2{}^q{}_j$) are equal, denoted *Equal*($L_1{}^p{}_i$, $L_2{}^q{}_j$), if: 1) at least one of the elements is an entry-point label and has the same context label as the other element which may not necessarily be an entry-point label; or 2) neither element is an entry-point label and both elements have the same context labels such that $p = q$. Formally,

$$
\begin{aligned}
&\textit{Equal}: \text{LABEL} \times \text{LABEL} \\
&\forall L_1{}^{p}{}_i, L_2{}^{q}{}_j: \text{LABEL} \bullet \\
&\quad (\exists\, CS_1, CS_2: \text{SCHEMA} \mid (L_1{}^{p}{}_i \in CS_1 \wedge L_2{}^{q}{}_j \in CS_2) \bullet \\
&\quad\quad (\textit{Equal}(L_1{}^{p}{}_i, L_2{}^{q}{}_j) = \text{TRUE}) \Leftrightarrow \\
&\quad\quad\quad (\exists \lambda_1, \lambda_2: \text{LABEL} \mid \lambda_1, \lambda_2 \in L_{EP} \bullet (\lambda_1 = L_1{}^{p}{}_i \wedge \lambda_2 = L_2{}^{q}{}_j \wedge L_1{}^{p}{}_i = L_2{}^{q}{}_j) \vee \\
&\quad\quad\quad\quad (\lambda_1 = L_1{}^{p}{}_i \wedge L_2{}^{q}{}_j \neq \lambda_2 \wedge L_1{}^{p}{}_i = L_2{}^{q}{}_j) \vee (\lambda_2 = L_2{}^{q}{}_j \wedge L_1{}^{p}{}_i \neq \lambda_1 \wedge L_1{}^{p}{}_i = L_2{}^{q}{}_j) \vee \\
&\quad\quad\quad\quad\quad L_1{}^{p}{}_i, L_2{}^{q}{}_j \notin L_{EP} \wedge p = q \wedge L_1{}^{p}{}_i = L_2{}^{q}{}_j))
\end{aligned}
$$

A *Terminal-Element* is an edge label where the target of the edge is in the set of atomic nodes. Formally,

$$
\begin{aligned}
&\textit{Terminal-Element} \triangleq l \mid l: \text{LABEL} \bullet \\
&\quad (\exists e: \text{EDGE} \bullet \textit{EdgeLabel}(e) = l \Leftrightarrow \textit{Target}(e) \in \textit{Atomic-Nodes})
\end{aligned}
$$

$$
\begin{aligned}
&\textit{NonTerminal-Element} \triangleq l \mid l: \text{LABEL} \bullet \\
&\quad (\exists e: \text{EDGE} \bullet \textit{EdgeLabel}(e) = l \Leftrightarrow \textit{Target}(e) \in \textit{Complex-Nodes})
\end{aligned}
$$

5.2.1 Element Constraints

1. Integrating a non-terminal element with a non-terminal element - Corresponding elements of the same modeling concepts will be integrated into a similar element. For example, if L_1 is an object in CS_1 and L_2 is an object in CS_2, and there is a correspondence assertion between both elements, then L_1 and L_2 are integrated into $L \in GIS$ (global integrated schema) as objects. Similarly, if L_1 and L_2 are attributes, they are integrated as attributes. Formally,

$$
\begin{aligned}
&\textit{NonTerminal-Element-Constraint}: \text{LABEL} \times \text{LABEL} \rightarrow \text{LABEL} \\
&\forall a, b: \text{LABEL} \mid a, b \in \textit{NonTerminal-Element} \bullet \\
&\quad \textit{NonTerminal-Element-Constraint}\,(a, b) = \\
&\quad\quad (c \mid c: \text{LABEL} \wedge c \in \textit{NonTerminal-Element}) \Leftrightarrow \\
&\quad\quad\quad (a = b \wedge a, b \in \textit{NonTerminal-Element}) \bullet a = c \vDash b = c
\end{aligned}
$$

2. Integrating a non-terminal element with a terminal element - To integrate a non-terminal element with a terminal element that have the same context label, we add both the terminal element and non-terminal element to the integrated schema as two different elements. The source name of the terminal element is appended to its context label so that it references its context schema. Formally,

Element-Constraint: LABEL × LABEL → LABEL
$\forall a, b$: LABEL | ($a \in$ *Terminal-Element* $\wedge$ $b \in$ *NonTerminal-Element*) •
Element-Constraint $(a, b) = (a', c$: LABEL |
($a' \in$ *Terminal-Element* $\wedge$ $c \in$ *NonTerminal-Element*) $\Leftrightarrow$
($a = a' \wedge b = c$) •
($\exists$ CS_1, CS_2: SCHEMA $\wedge$ $\exists L_1, L_2$: LABEL | ($L_1 \in CS_1 \wedge L_2 \in CS_2 \wedge$
Edge-Parent(a) = L_1 $\wedge$ *Edge-Parent*(b) = L_2) • $L_1 = L_2 \Rightarrow$
($\exists$ *GIS*: SCHEMA $\wedge$ $\exists$L: LABEL | ($L \in$ *NonTerminal-Element* $\wedge$ $L \in$ *GIS*) •
$L = L_1 \vdash L = L_2 \wedge$ *Edge-Parent*(a') = $L \wedge$ *Edge-Parent*(c) = L))

Note that $a' \in$ *GIS* is the same as $CS_1.a \in$ *GIS* which shows that $a' \in$ *GIS* is a terminal element that maps a corresponding terminal element in CS_1.

5.2.2 Inheritance Property of Integrated Elements

An object in the integrated schema inherits the descendants of any of its children from the context schema if no further assertion exists between that child and a corresponding object in another context schema that is involved in the integration process. For example, $L_1 \in CS_1 \wedge L_2 \in CS_2$ such that $L_1 = L_2 \Rightarrow \exists$L: LABEL such that $L \in$ *GIS*. Suppose ${L_1}^p{}_i \in$ *Edge-Child*(L_1) $\wedge$ $\neg(\exists$ ${L_2}^q$: LABEL | ${L_2}^q \in$ *Edge-Child*(L_2) • ${L_1}^p{}_i = {L_2}^q{}_j$)) then *Descendant*(${L_1}^p{}_i$) in the context schema will be added into the *GIS* as *Descendant*(${L}^q{}_k$) since no other assertion exists between ${L_1}^p{}_i$ and *Edge-Child*(L_2). This integration process leverages the inheritance property to ensure that objects integrated into the *GIS* maintain existing relationships with objects in the context schemas, such that the definitions in the local schemas are still valid.

Let L_{SET} be a set of context labels. A formal definition of *Inheritance* follows:

Inheritance: $CL \rightarrow CL$
$\forall L_1$: LABEL • $\exists$ CS_1: SCHEMA | $L_1 \in CS_1$ •
Inheritance(L_1) $\Leftrightarrow$ ($\exists$ L_2: LABEL; $\exists$ CS_2: SCHEMA | $L_2 \in CS_2$ • $L_2 = L_1 \wedge$
($\exists a$: LABEL • $a \in$ *Edge-Child*(L_1) $\wedge$
($\forall b$: LABEL | $b \in$ *Edge-Child*(L_2) • $b \neq a \Rightarrow$
($\exists$L: LABEL; $\exists$*GIS*: SCHEMA | $L \in$ *GIS* • $L = L_1$) $\wedge$
$a \in$ *Edge-Child*(L_1) $\Rightarrow \exists c$: *Edge-Child*(L) • $a = c \wedge$
($\forall a_i$: *Descendant*(a) • $\exists c_i$: *Descendant*(c) • $a_i = c_i$))))

Example 5.1 illustrates elements constraints assumption and the inheritance property.

Example 5.1 An interesting case occurs when there is an assertion between an object L_1 and an attribute *b* from CS_1 and CS_2, respectively. To simplify this exposition, let us consider a very simple example, the address of the author(s) of a book modeled differently in two schemas, CS_1 and CS_2.

CS_1 = Author [Name, Address [Number, Street, Code]]

CS_2 = Author [Id, Name, Address]

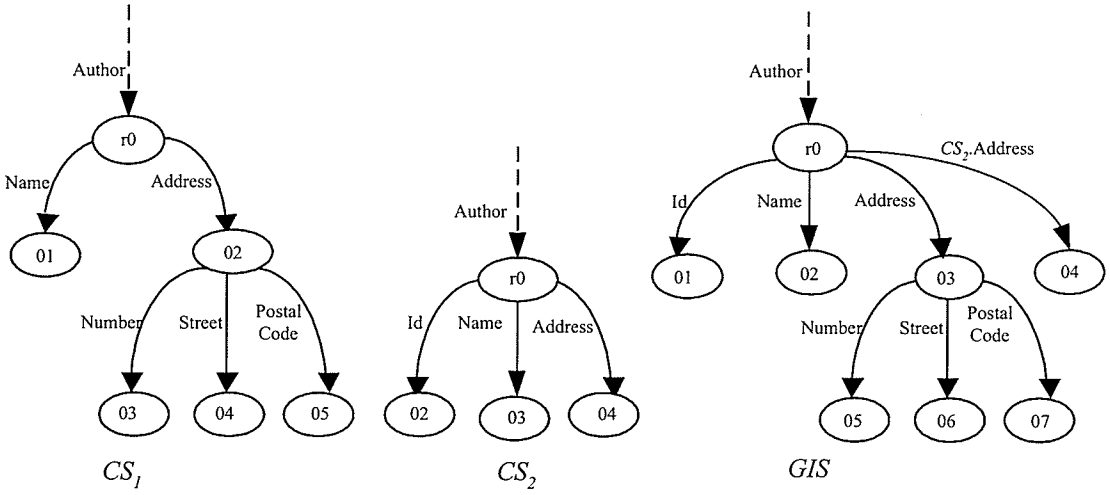

Figure 5.1: Attribute Integration Example

Observe from Figure 5.1 that "Address" in CS_1 is modeled as a non-terminal element with three child elements, while "Address" in CS_2 is modeled as a terminal element. An assertion exists between the non-terminal element, "Address" and the terminal element "Address" since their parents are related and both have the same name. Relying on the element constraints assumption permits the integration of the non-terminal element and its descendants as Address in the *GIS*. For the terminal element, the source name is appended to its context label in the *GIS* (CS_2.Address) so that it references its context schema.

By the inheritance property of integrated elements, child elements of "Address" in CS_1 are *inherited* into the *GIS* since they do not occur as child elements of "Address" in CS_2. If any of the child elements of "Address" in CS_1 (e.g. CS_1.Address.Street) had any descendant, such descendants would have been inherited into the *GIS* because they are guaranteed not to occur in CS_2.

5.2.3 Data Type Constraints

1. When types are compatible, the less restrictive of the two data types being integrated is chosen as the type for the *GIS*. For example, a decimal would be chosen when the types to be integrated are decimal and integer.
2. Data types that are not closely related are incompatible, e.g. string and integer. The union of the two local data types will form the data type for the *GIS* so that both may be stored.

5.2.4 Cardinality Constraints

Another thing to consider is how to resolve constraint conflicts among data sources. Cardinality constraint determines the number of possible relationships for each participating entity. This thesis adopts a strategy that considers a wider scope of constraints within the schemas involved in an integration process by allowing the least restrictive of the cardinality constraints. In this way, the cardinality constraints of the schemas are subsumed in the *GIS*. For example, if two objects L_1 and L_2 from context schemas CS_1 and CS_2 respectively have the following corresponding pairs of cardinality constraints (the resulting constraints is as indicated):

1. ([1: 1], [1: m]) = ([1: m]) (if the cardinality constraint of an entity in CS_1 is one-to-one and the cardinality constraint of the same entity in CS_2 is one-to-many, then the effective cardinality constraint of the entity in the *GIS* is the higher cardinality, which is one-to-many);
2. ([1: n], [m: 1]) = ([m: n]) (if the cardinality constraint of an entity in CS_1 is one-to-many and the cardinality constraint of the same entity in CS_2 is many-to-one, then the effective cardinality constraint of the entity in the *GIS* is the higher cardinality, which is many-to-many); and
3. ([1: n], [m: n]) = ([m: n]), (if the cardinality constraint of an entity in CS_1 is one-to-many and the cardinality constraint of the same entity in CS_2 is many-to-many, then the effective cardinality constraint of the entity in the *GIS* is the higher cardinality, which is many-to-many).

5.2.5 Terminal Element Constraints

Without prejudice to the element constraints described in Section 5.2.1, the assertion between terminal elements is dependent on the assertions between their parents. If there is an assertion between terminal elements that conflicts with an assertion between the parents of such terminal elements, to that extent, the terminal element assertion is invalid, and does not lead to a "well-formed" integrated schema. Suffice it to say that, no correspondence assertion exists between terminal elements unless there is an assertion between their respective parents. For example, a correspondence assertion exists between terminal element $a \in$ *Edge-Child*(L_1) from CS_1 and terminal element $b \in$ *Edge-Child*(L_2) from CS_2 if and only if *Equal*(a, b) and *Equal*(*Edge-Parent*(a), *Edge-Parent*(b)).

The definition of a terminal element assertion (*Terminal-Assertion*) follows.

$$
\begin{aligned}
&\textit{Terminal-Assertion}: \textit{Terminal-Element} \times \textit{Terminal-Element} \\
&\forall a, b: \textit{Terminal-Element} \bullet \\
&\quad \textit{Terminal-Assertion}\ (a, b) \Leftrightarrow (\exists\ CS_1, CS_2: \text{SCHEMA} \wedge \\
&\qquad \exists L_1, L_2: \text{LABEL} \mid L_1 \in CS_1 \wedge L_2 \in CS_2 \wedge \\
&\qquad\quad \textit{Edge-Parent}(a) = L_1 \wedge \textit{Edge-Parent}(b) = L_2) \bullet \\
&\qquad\qquad L_1 = L_2 \wedge \textit{Edge-Parent}(a) = \textit{Edge-Parent}(b)
\end{aligned}
$$

If the defined correspondence assertions exist, the terminal elements are then integrated into the *GIS* according to the assertion. Stated differently, the above formalism stipulates that no correspondence assertion exists between any two corresponding terminal elements if the parents of the terminal elements are unrelated. For example, if the parents of a and b are not related by any correspondence assertion, then a and b cannot be integrated into a corresponding terminal element in the *GIS*. Thus, a and b should instead be inherited by their respective parents in the *GIS*, using the inheritance property.

5.2.6 Key Constraints

Key constraints are resolved in the *GIS* by creating a new key that is the union of the two original key elements. Suppose $a \in CS_1$ is a terminal element and a key, and $b \in CS_2$ is also a terminal element and a key, and *Edge-Parent*(a) = *Edge-Parent*(b). This implies

that $a, b \in$ *Edge-Child*(L) $\in$ *GIS*. But a and b cannot be keys in *GIS* because a key element must be unique. The approach is to create a new key (say α) such that $\alpha = (a \cup b)$. Let KEY be the set of terminal elements that are key elements in a given schema. A formal definition of a key constraint follows.

Key-Constraint: *Terminal-Element* × *Terminal-Element*
$\forall a, b$: *Terminal-Element* | $a \neq b \wedge a, b \in KEY$ •
Key-Constraint$(a, b) \Rightarrow (\exists\ CS_1, CS_2$: SCHEMA; $\exists L_1, L_2$: LABEL |
$(L_1 \in CS_1 \wedge L_2 \in CS_2 \wedge$ *Edge-Parent*$(a) = L_1 \wedge$ *Edge-Parent*$(b) = L_2) \bullet L_1 = L_2 \Rightarrow$
$(\exists GIS$: SCHEMA; $\exists$L: LABEL | L $\in$ *NonTerminal-Element* $\wedge$ L $\in$ *GIS*) •
$a, b \in$ *Edge-Child*(L) $\wedge \exists \alpha \in$ *Key* • $\alpha = (a \cup b)$)

5.3 Analysis of Assertions and Generation of Integration Rules

The integration algorithm automatically performs matching of elements based on the context labels and structure[39] of schema elements to generate correspondence assertions [AE04b]. An analysis of each assertion is carried out to produce formal rules that state how to derive the constructs which are to be inserted into the *GIS*. These formal rules, also called correspondence rules ($\Re$), are specified automatically using algebraic signatures. Algebraic signatures are used to simplify the modeling of complex hierarchical structures using a set of logics and predicates because algebraic signatures remove ambiguity from the informal definitions. Formal rules present a well-founded logical framework in which concepts are defined using descriptions built from a set of constructors.

Definition 5.2 A *correspondence rule* $\Re$ is a formal rule that logically represents the derivations between the context schemas and the *GIS*. ■

Whereas correspondence assertions show the relationship between two context schemas, correspondence rules logically represent the derivations between the context schemas and the *GIS*. For each pair of elements of CS_1 and CS_2 that have common context labels there is an assertion that shows the relationship between those elements. For example, *Equal* $(L_1{}^p{}_i, L_2{}^q{}_j)$ implies that there is an assertion Ψ that shows the equivalence

[39] Semantics is built into the structure because of the hierarchical nature of *Del-G*.

relationship between $(L_1{}^p{}_i, L_2{}^q{}_j)$. Let ω be a set of elements of the pair $(L_1{}^p{}_i, L_2{}^q{}_j)$ such that $L_1{}^p{}_i \in CS_1$ and $L_2{}^q{}_j \in CS_2$ and $Equal(L_1{}^p{}_i, L_2{}^q{}_j)$. Let $\varpi = (L_1{}^p{}_i \cup L_2{}^q{}_j) \setminus \omega$ be a set of elements[40] from CS_1 and CS_2 such that for all elements in CS_1 and CS_2 the constraint $\neg Equal(L_1{}^p{}_i, L_2{}^q{}_j)$ holds. The union of ω and ϖ represents the set of local schemas CS_{SET}. By definition,

$$\omega = \bigcup_{i,j}^{n} (L_1{}^p{}_i, L_2{}^q{}_j) \bullet (\forall L_1{}^p{}_i \in CS_1 \bullet (\exists L_2{}^q{}_j \in CS_2 \bullet Equal(L_1{}^p{}_i, L_2{}^q{}_j))) \tag{5.1}$$

$$\varpi = \bigcup_{i,j=1}^{n} (L_1{}^p{}_i, L_2{}^q{}_j) \bullet (\forall L_1{}^p{}_i \in CS_1; L_2{}^q{}_j \in CS_2 \bullet \neg Equal(L_1{}^p{}_i, L_2{}^q{}_j)) \tag{5.2}$$

$$\omega \cup \varpi = CS_{SET} \tag{5.3}$$

Let Ψ be an assertion such that:

Ψ ::= Identity | Equivalence | Intersection | Compatibility | Inclusion |

Mutual-Inclusion | Exclusion

$\Psi(\omega)$ is a function that declares the relationship between the elements of ω. The assertion (Ψ) shows how elements of ω are related. Elements of ϖ maintain their local relationships with their ancestors or descendants, which are merged into the *GIS* by *inheritance*. $\Re$ is a set of production rules for generating the *GIS* from ω and ϖ. The *GIS* is generated such that elements of ω are merged according to their respective assertions, while elements of ϖ are merged into the *GIS* to maintain the existing local relationships between those elements and their parents. Accordingly, the signature of an integrated schema generation function (Υ_{GIS}) that generates the *GIS* from the context schemas using correspondence assertions and correspondence rules follows.

$$\Upsilon_{GIS}: \Psi(\omega) \times \Re \times \varpi \rightarrow GIS \quad \begin{cases} \omega = \varnothing \Rightarrow \bigcap_{i,j=1}^{n} (L_1{}^p{}_i, L_2{}^q{}_j) = \varnothing \\ \varpi = \varnothing \Rightarrow \bigcup_{i,j=1}^{n} (L_1{}^p{}_i, L_2{}^q{}_j) = \omega \\ (\omega \neq \varnothing) \wedge (\varpi \neq \varnothing) \Rightarrow \bigcap_{i,j=1}^{n} (L_1{}^p{}_i, L_2{}^q{}_j) \neq \varnothing \end{cases} \tag{5.4}$$

[40] The operator '\' is the set difference operator.

$\omega = \varnothing \Rightarrow \bigcap_{i,j}^{n} (L_1{}^p{}_i, L_2{}^q{}_j) = \varnothing$ implies that $\Psi(\omega)$ contains the *exclusion assertion* because there are no common context labels in CS_1 and CS_2 which are related, and hence no stated assertion between them. $\varpi = \varnothing \Rightarrow \bigcup_{i,j}^{n} (L_1{}^p{}_i, L_2{}^q{}_j) = \omega$ implies that $\Psi(\omega)$ contains the *identical assertion* since every element in CS_1 and CS_2 are related. $(\omega \neq \varnothing) \wedge (\varpi \neq \varnothing) \Rightarrow \bigcap_{i,j}^{n} (L_1{}^p{}_i, L_2{}^q{}_j) \neq \varnothing$ implies that $\Psi(\omega)$ contains any of the stated assertions except identical and exclusive assertions. Analysis of the assertions contained in $\Psi(\omega)$ yields the correspondence rules for the given assertion. The notion of inheritance ensures that definitions of elements in the context schemas are still valid after integration.

Recall,

$$\omega \cup \varpi = CS_{\text{SET}}$$

and

$$\begin{aligned} &\text{dom}(\Re) = \{L_i{}^p{}_j : \text{LABEL} \mid L_i{}^p{}_j \in CS_{\text{SET}}\} \Rightarrow \\ &\quad (\text{dom}(\Upsilon_{GIS}) = \{L_i{}^p{}_j : \text{LABEL} \mid L_i{}^p{}_j \in CS_{\text{SET}}\} \Rightarrow \\ &\qquad \Upsilon_{GIS} : CS_{\text{SET}} \rightarrow GIS) \end{aligned} \tag{5.5}$$

There is a need to show that the function, Υ_{GIS} maps all elements of CS_{SET} to *GIS* in a reasonable manner. To clarify this exposition, the following definitions of some basic set concepts are given.

Definition 5.2 Let X and Y be two sets;

1. A function F is functional if: $(F: X \rightarrow Y \bullet (\forall x: X \bullet \exists_1 y: Y \bullet F(x) = y))$.
2. A function F is total if: $(F: X \rightarrow Y \bullet \text{dom}(F) = X)$.
3. A function F is surjective if: $(F: X \rightarrow Y \bullet \text{ran}(F) = Y)$[41].

■

Given any two context schemas CS_1, $CS_2 \in CS_{\text{SET}}$, there exists an integration generation function Υ_{GIS} that takes elements of CS_1 and CS_2 as input and yields a *GIS* as output. Υ_{GIS} induces a functional, total, and surjective correspondence between elements of CS_1 and CS_2 and elements of *GIS*.

[41] Note that dom(F) is the domain of F and ran(F) is the range of F.

The need is to show that Υ_{GIS} is 1) functional; 2) total; and 3) surjective. Let CS_{SET} be the union of all the elements in the local schemas CS_i and *GIS* be the set of elements in the global integrated schema.

1. To show that Υ_{GIS} is functional is to show that for any two schemas CS_1, $CS_2 \in CS_{SET}$ such that there exist ${L_1}^p{}_i \in CS_1$ and ${L_2}^q{}_j \in CS_2$ and $Equal({L_1}^p{}_i, {L_2}^q{}_j)$ implies that there exists at most one ${L}^r{}_k \in GIS$.

$$\begin{aligned}&\Upsilon_{GIS}: \Psi(\omega) \times \Re \times \varpi \rightarrow GIS \Leftrightarrow \\ &\forall\, CS_1, CS_2: \text{SCHEMA} \bullet \\ &\quad (\exists {L_1}^p{}_i, {L_2}^q{}_j: \text{LABEL} \mid ({L_1}^p{}_i \in CS_1 \wedge {L_2}^q{}_j \in CS_2) \bullet \\ &\qquad (Equal({L_1}^p{}_i, {L_2}^q{}_j) \Rightarrow (\exists_1 {L}^r{}_k: \text{LABEL} \mid {L}^r{}_k \in GIS \bullet \\ &\qquad\quad Equal({L_1}^p{}_i, {L}^r{}_k) \vdash Equal({L_2}^q{}_j, {L}^r{}_k))))\end{aligned} \tag{5.6}$$

2. To show that Υ_{GIS} is total is to show that every element in CS_{SET} is also in *GIS*.

$$\begin{aligned}&\Upsilon_{GIS}: \Psi(\omega) \times \Re \times \varpi \rightarrow GIS \Leftrightarrow \\ &\quad \forall {L_i}^p{}_j: \text{LABEL} \mid {L_i}^p{}_j \in CS_{SET} \bullet \\ &\qquad (\exists {L}^r{}_k: \text{LABEL} \mid {L}^r{}_k \in GIS \bullet Equal({L_i}^p{}_j, {L}^r{}_k))\end{aligned} \tag{5.7}$$

3. To show that Υ_{GIS} is surjective is to show that every element in *GIS* corresponds to at least one element in CS_{SET}.

$$\begin{aligned}&\Upsilon_{GIS}: \Psi(\omega) \times \Re \times \varpi \rightarrow GIS \Leftrightarrow \\ &(\forall {L}^r{}_k: \text{LABEL} \mid {L}^r{}_k \in GIS \bullet \exists {L_i}^p{}_j \in CS_{SET} \bullet Equal({L_i}^p{}_j, {L}^r{}_k)) \wedge \\ &\quad (\forall {L_i}^p{}_j, {L_i}^q{}_m \in CS_{SET} \bullet Equal({L_i}^p{}_j, {L_i}^q{}_m) \Rightarrow \exists_1 {L}^r{}_k \in GIS \bullet \\ &\qquad (Equal({L_i}^p{}_j, {L}^r{}_k) \vdash Equal({L_i}^q{}_m, {L}^r{}_k)))\end{aligned} \tag{5.8}$$

We can show that the function Υ_{GIS} is total from the definition of the *GIS* since every element in CS_{SET} is also in the *GIS*. Also, that Υ_{GIS} is surjective means every element in the local schemas can be correctly traced to the *GIS*. The functional correspondence of Υ_{GIS} can be deduced from the definition of the *GIS* since $(\forall {L}^r{}_k, {L}^r{}_j \in GIS \mid k \neq j \Rightarrow {L}^r{}_k \neq {L}^r{}_j)$ and established in (5.6), (5.7) and (5.8). A function $(F(x) = y)$ is functional if for every x in the domain of F, there exists exactly one y in the range of F. No two elements in the range can map to the same element in the domain of F therefore, functionality suggests minimality and hence leads to the following consequence.

The global integrated schema (*GIS*) minimally represents the set of all elements in the local schemas CS_{SET}. That is the function Υ_{GIS} is a consistent and complete mapping from CS_{SET} to *GIS*.

The logic of (5.6), (5.7) and (5.8) is sound if $(L_i^{p}{}_j \in CS_{SET} \vdash L^{r}_k \in GIS \Rightarrow L_i^{p}{}_j \models L^{r}_k)$. Thus,

$$
\begin{aligned}
&CS_{SET} = \mathrm{dom}(\Upsilon_{GIS}) \Rightarrow \\
&\quad (\forall L_i^{p}{}_j : \mathrm{LABEL} \mid L_i^{p}{}_j \in CS_{SET} \bullet \\
&\qquad (\exists L^{r}_k : \mathrm{LABEL} \mid L^{r}_k \in GIS \bullet (L_i^{p}{}_j \in CS_{SET} \vdash L^{r}_k \in GIS)) \wedge \\
&\qquad\quad (GIS = \mathrm{ran}(\Upsilon_{GIS}) \Rightarrow \\
&\qquad\qquad (\forall L^{r}_k \in GIS \bullet \exists L_i^{p}{}_j \in CS_{SET} \mid Equal(L_i^{p}{}_j, L^{r}_k) \bullet \\
&\qquad\qquad\quad (L^{r}_k \in GIS \vdash L_i^{p}{}_j \in CS_{SET})))
\end{aligned}
$$

The logic of (5.6), (5.7) and (5.8) is complete if $(L_i^{p}{}_j \in CS_{SET} \models L^{r}_k \in GIS \Rightarrow L_i^{p}{}_j \vdash L^{r}_k)$. Thus,

$$
\begin{aligned}
&GIS = (\mathrm{ran}(\Upsilon_{GIS}) \Rightarrow \\
&\quad (\forall L_i^{p}{}_j : \mathrm{LABEL} \mid L_i^{p}{}_j \in CS_{SET} \bullet \\
&\qquad (\exists L^{r}_k \in GIS \mid Equal(L_i^{p}{}_j, L^{r}_k) \bullet L_i^{p}{}_j \in CS_{SET} \vdash L^{r}_k \in GIS)) \wedge \\
&\qquad\quad (CS_{SET} = \mathrm{dom}(\Upsilon_{GIS}) \Rightarrow \\
&\qquad\qquad (\forall L_i^{p}{}_j \in CS_{SET} \bullet (\exists L^{r}_k \in GIS \bullet \\
&\qquad\qquad\quad (L_i^{p}{}_j \in CS_{SET} \vdash L^{r}_k \in GIS)))
\end{aligned}
$$

The soundness of the logic of (5.6), (5.7) and (5.8) is established because Υ_{GIS} is a total surjective function from CS_{SET} to *GIS*, whose range is the whole of *GIS*. Completeness of the logic of (5.6), (5.7) and (5.8) is established because the function Υ_{GIS} is surjective and total. This implies that $GIS = \mathrm{ran}(\Upsilon_{GIS})$ and $CS_{SET} = \mathrm{dom}(\Upsilon_{GIS})$. Thus, *GIS* represents the views and CS_{SET} represents the database.

The main aim of the approach adopted in this thesis is to remove the burden of identifying correspondence assertions from the DBA and the user. Instead the technique proposed in this thesis dynamically identifies similarities based on the matching of names and descriptions, and hierarchical semantic relationships. For each pair of elements of CS_1 and CS_2 that have common names there is an assertion that shows the relationship between those elements. An analysis of each correspondence assertion is given to generate the correspondence rules that map elements from the context schemas to the the

GIS. Let L denote the element in *GIS* resulting from merging L_1 and L_2, and $L^r_k \in GIS$ which results from merging $L_1{}^p{}_i$ and $L_2{}^q{}_j$. The analysis of assertions and the corresponding integration rules generated follows.

5.3.1 Integration Rule for Identity Assertions

An analysis of identity assertions ($L_1 = L_2 \wedge$ *Edge-Child*(L_1) = *Edge-Child*(L_2))) to generate the corresponding integration rules is now given. Identity assertion applies to element relationships at the same level of perception (for example, non-terminal elements to non-terminal elements, or terminal elements to terminal elements). For identity assertion, L is structurally the same with L_1 and L_2, and L^r_k is structurally the same with $L_1{}^p{}_i$ and $L_2{}^q{}_j$. $L_1 = L_2$ and for every *Edge-Child*(L_1), $L_1{}^p{}_i$ there exist a corresponding *Edge-Child*(L_2), $L_2{}^q{}_j$ such that $L_1{}^p{}_i = L_2{}^q{}_j$. The integration rule for identity assertions follows.

begin

Merge L_1 and L_2 into $L \in GIS$, such that $L = L_1 = L_2$ (Rule 1)
for each *Edge-Child*(L_1) and *Edge-Child*(L_2) do
  Merge $L_1{}^p{}_i$ and $L_2{}^q{}_j$ into L^r_k such that $L^r_k = L_1{}^p{}_i = L_2{}^r{}_k$ (Rule 2)

end

5.3.2 Integration Rule for Equivalence Assertions

For equivalence assertions, $L_1 = L_2 \wedge$ (*Edge-Child*(L_1) $\supseteq$ *Edge-Child*(L_2) $\vee$ *Edge-Child*(L_1) $\subseteq$ *Edge-Child*(L_2)). This implies that L_1 and L_2 will be merged into $L \in GIS$, and any *Edge-Child*(L_1) $\notin$ *Edge-Child*(L_2) or *Edge-Child*(L_2) $\notin$ *Edge-Child*(L_1) will be "inherited" by its parent in the *GIS*. Finally, any *Edge-Child*(L_1) = *Edge-Child*(L_2) is merged into L^r_k. The set of rules for equivalence assertions is given below.

begin

Merge L_1 and L_2 into $L \in GIS$, such that $L = L_1 = L_2$ (Rule 1)
for each *Edge-Child*(L_1) and *Edge-Child*(L_2) where
  Edge-Child(L_1) $\supseteq$ *Edge-Child*(L_2) do
  Inheritance($L_1{}^p{}_i$) into $L^r_k \in GIS$ such that
  $L^r_k = L_1{}^p{}_i \wedge$ *Descendant*($L_1{}^p{}_i$) = *Descendant*(L^r_k) (Rule 3)
for each *Edge-Child*(L_1) and *Edge-Child*(L_2) where
  Edge-Child(L_1) $\subseteq$ *Edge-Child*(L_2) do
  Inheritance($L_2{}^q{}_j$) into $L^r_k \in GIS$ such that

$L^r_k = L_2{}^q_j \wedge$ *Descendant*$(L_1{}^p_i)$ = *Descendant*(L^r_k) (Rule 4)
for each *Edge-Child*(L_1) and *Edge-Child*(L_2) where
Edge-Child(L_1) = *Edge-Child*(L_2) do
Merge $L_1{}^p_i$ and $L_2{}^q_j$ into L^r_k such that $L^r_k = L_1{}^p_i = L_2{}^q_j$ (Rule 2)
end

Example 5.2 Let us consider an example to illustrate how the integration rules might be generated. Information about the personal data of the author(s) of books is represented in two different context schemas, CS_1 and CS_2 shown below. Although CS_1 and CS_2, represent the same universe of discourse, they still differ because of the differences in the designers' perceptions.

Context Schema 1 (CS_1)	Context Schema 2 (CS_2)
Author [Name, Address [Number, Street, Code]]	Author [Id, Name, Address]

"Author" in CS_1 is modelled with two attributes while in CS_2 it is modelled with three attributes, and all the attributes of CS_1 are also in CS_2, which implies that $L_1 = L_2 \wedge$ (*Edge-Child*$(L_1) \supseteq$ *Edge-Child*$(L_2) \vee$ *Edge-Child*$(L_1) \subseteq$ *Edge-Child*(L_2)). An analysis of the equivalence assertion in the above example is given below.

Identification of Assertions	Mappings
$L_1 = L_2$ = Author $L_1{}^{L1}{}_1 = L_2{}^{L2}{}_2$ = Name $L_1{}^{L1}{}_2$ = Address $L_2{}^{L2}{}_3$ = Address $L_2{}^{L2}{}_1$ = Id Id $\notin$ *Edge-Child*$(L_1) \Rightarrow$ (*Edge-Child*$(L_1) \subseteq$ *Edge-Child* (L_2))	$L = L_1 = L_2$ $L^L{}_1 = L_1{}^{L1}{}_1 = L_2{}^{L2}{}_2$ $L^L{}_2 = L_2{}^{L2}{}_2$ $L^L{}_2 = L_1{}^{L1}{}_2 = L_2{}^{L2}{}_3$ $L^L{}_3 = L_2{}^{L2}{}_1$

According to the element constraints rule, corresponding elements of the same modeling concepts are integrated into a similar element. In this case, $L_1 = L_2$, $L_1{}^{L1}{}_1 = L_2{}^{L2}{}_2$, and $L_1{}^{L1}{}_2 = L_2{}^{L2}{}_3$ are integrated into similar elements in the integrated schema. Then, *Edge-Child*$(L_1{}^p_i)$ and *Edge-Child*$(L_2{}^q_j)$ are compared for likely assertions. $L_2{}^{L2}{}_1 \notin$ *Edge-*

Child(L_1) is inherited by $L \in GIS$. Both CS_1 and CS_2 represent the same object, "Author", but only some of the possible attributes of CS_2 are of interest to CS_1.

For intersection assertion, ($L_1 = L_2 \wedge$ (*Edge-Child*(L_1) $\cap$ *Edge-Child*(L_2) $\neq \varnothing$)), both CS_1 and CS_2 represent the same object, but each schema is modelled so that it has a partial record of the object reflecting what each user values. However, there are some elements that are of interest to users in both schemas, hence *Edge-Child*(L_1) $\cap$ *Edge-Child*(L_2) $\neq \varnothing$. An analysis of this assertion shows that since $L_1 = L_2$, rule 1 of Section 5.3.2 applies. The intersection of the child elements is not empty, which means that in the set of child elements: 1) there are some elements that are common to both CS_1 and CS_2, then rule 2 of Section 5.3.2 is applied; 2) there are some elements in (*Edge-Child*(L_1) that are not in *Edge-Child*(L_2), we apply rule 3 (inheritance of *Edge-Child*(L_1) $\notin L_2$); and finally 3) there are some elements that are in L_2 that are not in L_1, and rule 4 is applied, (inheritance of *Edge-Child*(L_2) $\notin L_1$).

For compatibility assertions, ($L_1 = L_2 \wedge$ (*Edge-Child*(L_1) $\cap$ *Edge-Child*(L_2) $= \varnothing$)). In this case, CS_1 and CS_2 emphasize different elements, though they represent the same object. A careful examination of this assertion shows that since $L_1 = L_2$, rule 1 applies. There are only two possible scenarios for the child elements that are handled using rules 3 and 4. The set of rules for intersection and compatibility assertions is omitted since they are similar to those for equivalence assertion.

5.3.3 Integration Rule for Inclusion Assertions

An analysis of inclusion assertions ($L_1 \neq L_2 \wedge$ (*Descendant*(L_1) $\supseteq L_2 \vee L_1 \subseteq$ *Descendant*(L_2))) to generate the corresponding integration rules is given. An inclusion assertion applies to element relationships at different levels of perception (both non-terminal elements and terminal elements). In the case of (L_1 *Inclusion* L_2), $L_1 \neq L_2$ but *Descendant*(L_1) $\supseteq L_2$. L_2 is a specialization of L_1 (conversely, L_1 is a generalization of L_2). Elements of CS_1 are merged into the *GIS* until an ${L_1}^p{}_i = L_2$ is encountered. At this point, every element of CS_1 whose parent is p (except ${L_1}^p{}_i = L_2$) is merged into the *GIS* which inherits its descendants. Then, (${L_1}^p{}_i$, L_2) (where ${L_1}^p{}_i = L_2$) is merged into the *GIS*.

begin

Merge L_1 into $L \in GIS$, such that $L = L_1$ (Rule 5)
for each *Edge-Child*(L_1) where ${L_1}^p{}_i \neq L_2$ do
  Merge ${L_1}^p{}_i$ into ${L^r}_k \in GIS$ such that ${L^r}_k = {L_1}^p{}_i$ (Rule 6)
  if (${L_1}^p{}_i = L_2$)
    Merge ${L_1}^p{}_i$ into ${L^r}_k \in GIS$ such that ${L^r}_k = {L_1}^p{}_i = L_2$ (Rule 7)
    for each *Edge-Child*(L_1) and *Edge-Child*(L_2) where
      Edge-Child(L_1) $\supseteq$ *Edge-Child*(L_2) do
      Inheritance(${L_1}^p{}_i$) into ${L^r}_k \in GIS$ such that
      ${L^r}_k = {L_1}^p{}_i \wedge$ *Descendant*(${L_1}^p{}_i$) = *Descendant*(${L^r}_k$) (Rule 3)

end

Similarly, for (L_1 *Inclusion* L_2), $L_1 \neq L_2$ but *Descendant*(L_1) $\subseteq L_2$. L_1 can be seen as a specialization of L_2 (conversely, L_2 is a generalization of L_1). The set of rules for (L_1 *Inclusion* L_2) is given below.

begin

Merge L_2 into $L \in GIS$, such that $L = L_2$ (Rule 8)
for each *Edge-Child*(L_1) where ${L_2}^q{}_j \neq L_1$ do
  Merge ${L_2}^q{}_j$ into ${L^r}_k \in GIS$ such that ${L^r}_k = {L_2}^q{}_j$ (Rule 9)
  if (${L_2}^q{}_j = L_1$)
    Merge ${L_2}^q{}_j$ into ${L^r}_k \in GIS$ such that ${L^r}_k = L_1 = {L_2}^q{}_j$ (Rule 10)
    for all elements at level k do (except ${L_2}^q{}_j = L_1$)
      Inheritance(${L_2}^q{}_j$) into ${L^r}_k \in GIS$ such that
      ${L^r}_k = {L_2}^q{}_j \wedge$ *Descendant*(${L_1}^p{}_i$) = *Descendant*(${L^r}_k$) (Rule 4)

end

The analysis of the other variants of inclusion assertion generates similar rules, and therefore, is omitted.

Example 5.3 Consider the information about some publications made by faculty members, for which different designers have defined the context schemas, CS_1 and CS_2 below.

Context Schema 1 (CS_1)	Context Schema 2 (CS_2)
Publications [Journals, Conferences, Books [Title, ISBN] Author [Birthday, Name] Address [Street, Number]]	Books [Title, ISBN, Author [Birthday, Name] Address [City, Street, Number]]

CS_1 presents the world of interest as publications, which can be journals, conferences or books. CS_2 is only interested in books written by faculty members, which is a part of what CS_1 describes. Obviously, the two context schemas represent the same universe of discourse. In CS_1 "publications" could be any of "Journals, Conferences, or Books". Schema CS_2 represents "Books" which is part of publications.

Let Publications = $L_1 \in CS_1$ and Books = $L_2 \in CS_2$. $L_1 \neq L_2$ but *Descendant*(L_1) $\supseteq$ L_2. Clearly, L_2 is a specialization of L_1 (conversely, L_1 is a generalization of L_2). Merge the elements of "Publications" into the *GIS* until an ${L_1}^{p}{}_i = L_2$ = Book is encountered. At this point, every element in CS_1 at level l (except ${L_1}^{p}{}_i = L_2$) merged into the *GIS* inherits its descendants, if any exist. That means ${L^{L}}_1 = {L_1}^{L1}{}_1$ = "Journals" and $L_{12} = {L_1}^{L1}{}_2$ = "Conferences" will inherit their descendants. But ${L_1}^{L1}{}_1$ and ${L_1}^{L1}{}_2$ are terminal elements that have no descendants. Then, $({L_1}^{L1}{}_3, L_2)$ (where ${L_1}^{L1}{}_3 = L_2$) is merged into the *GIS*.

5.3.4 Integration Rule for Mutual-Inclusion Assertions

An analysis of mutual-inclusion assertion ($L_1 \neq L_2 \wedge$ (*Descendant*(L_1) $\supseteq L_2 \wedge L_1 \subseteq$ *Descendant*(L_2))) to generate the corresponding integration rules is given. A mutual-inclusion assertion applies to element relationships at different levels of perception (both non-terminal elements and terminal elements). In the case of (L_1 *Mutual-Inclusion* L_2), $L_1 \neq L_2$ but $L_2 \in$ *Descendant*(L_1) and $L_1 \in$ *Descendant*(L_2). L_1 and L_2 are both specialization and generalization of each other.

Figure 5.2 represents two different context schemas of a library information system modelled from two different perspectives. CS_1 models books with attributes title, ISBN, and author(s). CS_2 models the authors with attributes name, birthday, and books. These schemas actually represent the same universe of discourse, but seen from different ways by different designers. Suppose in CS_1, L_1 = Books, with direct children ${L_1}^{L1}{}_1$ = Title, ${L_1}^{L1}{}_2$ = ISBN, and ${L_1}^{L1}{}_3$ = Authors. In CS_2, L_2 = Authors, with direct children ${L_2}^{L2}{}_1$ = Name, ${L_2}^{L2}{}_2$ = Birthday, and ${L_2}^{L2}{}_3$ = Books. $L_1 \neq L_2$, but $L_1 = {L_2}^{L2}{}_3$, $L_2 = {L_1}^{L1}{}_3$. If this case is treated as a true inclusion, some elements may occur more than once in the *GIS*, which would create redundancy and incorrect representation of the context schemas. So,

descendants of L_1, excluding *Descendant*($L_1{}^{L1}{}_3$) since $L_1{}^{L1}{}_3 = L_2$ (Books [Title, ISBN]) and *Descendant*($L_2{}^{L2}{}_3$) (Books [Title, ISBN]) must be integrated.

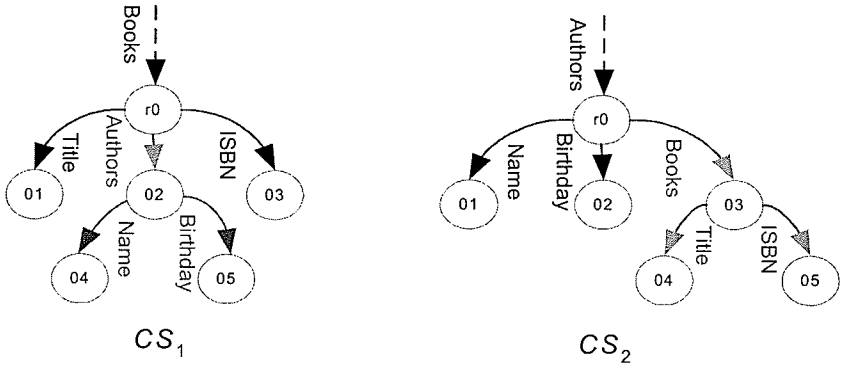

Figure 5.2: Sample Schemas to Illustrate Mutual-Inclusion Assertion

Also, consider L_2, excluding *Descendant*($L_2{}^{L2}{}_3$) since $L_2{}^{L2}{}_3 = L_1$ had already been considered (i.e., Author [Name, birthday]) and *Descendant*($L_1{}^{L1}{}_3$) (Author [Name, birthday]). Accordingly, the following rules will be generated for mutual-inclusion assertion.

begin

Remove *Descendant*($L_1{}^p{}_i$) from CS_1 (where $L_1{}^p{}_i = L_2$)
Merge $L_1{}^p{}_i$ into $L^r{}_k \in GIS$ such that $L^r{}_k = L_1{}^p{}_i = L_2{}^q{}_j$ (Rule 7)
Remove *Descendant*($L_2{}^q{}_j$) from CS_2 (where $L_2{}^q{}_j = L_1$)
Merge L_2 into $L^r{}_k \in GIS$ such that $L^r{}_k = L_2 = L_1{}^p{}_i$ (Rule 10)

end

5.3.5 Integration Rule for Exclusion Assertions

An analysis of the exclusion assertions ($L_1 \neq L_2 \wedge (L_1 \notin$ *Descendant*(L_2) $\wedge$ $L_2 \notin$ *Descendant*(L_1))) to generate the corresponding integration rules is now given. If $L_1 \neq L_2$, and $L_1 \notin$ *Descendant*(L_2) and $L_2 \notin$ *Descendant*(L_1) then there is no relationship between elements of L_1 and L_2. The empty string (ε) in the *GIS* is introduced such that *Edge-Parent*(L_1) = ε and *Edge-Parent*(L_2) = ε. L_1 and L_2 in the *GIS* inherit their descendants from CS_1 and CS_2, respectively. The empty string (ε) is in the set of labels, and may occur

as entry point label. Consider the schemas of Figure 5.3. CS_1 is the schema for books, while CS_2 is the schema for journals available in two online bookstores.

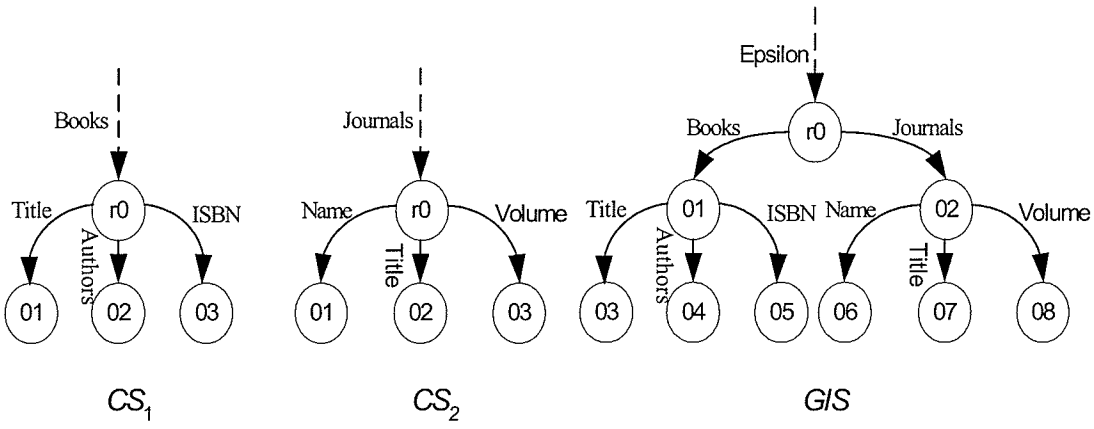

Figure 5.3. A Diagrammatic Representation of Exclusion Assertion

Although, both CS_1 and CS_2 represent different data objects, they belong to the same universe of discourse, publication. Observe from *Exclusion Assertion* that there is no relationship between elements of CS_1 and CS_2. Therefore, $L_1 \in CS_1$ and $L_2 \in CS_2$ will be merged into the *GIS* as children of $\varepsilon \in GIS$. The set of rules for exclusive assertion follows.

Begin
Let $\varepsilon \in GIS \mid \varepsilon = \lambda$ (where λ is the entry-point label for *GIS*) (Rule 11)[42]
Insert L_1 into $L^r_k \in GIS$ such that *Edge-Parent* $(L_1) = \varepsilon$ and $L^r_k = L_1$ (Rule 12)
Insert L_2 into $L^r_k \in GIS$ such that *Edge-Parent* $(L_2) = \varepsilon$ and $L^r_k = L_2$ (Rule 13)
end

Mapping in most integration systems is provided manually using heuristics or machine learning approaches that rely on feedback from the users. For example, Noy and Musen [NM00] use heuristics that identify structural and naming similarities between data sources. Machine learning approaches used to learn mappings between data sources include [DDH01, LH01, BM02, DMDH02]. These systems require significant user participation as the user is expected to provide feedback to the system to further refine a proposed mapping. Madhavan *et al.* [MBDH02] provide a tool for representing and

[42] Note that the empty string (ε) is in the set of labels, and occurs as the entry point label.

reasoning about mappings between domain models, but fail to dynamically generate mappings. The approach in this thesis goes beyond representing and reasoning about mapping [MBDH02] to dynamically generating mappings of attribute semantics between objects in different data sources. The integration algorithm dynamically identifies assertions [AE04e]; and analyses the assertions to generate integration rules [EA04]. To the best of our knowledge, this is the first work to formally specify a system that dynamically generates mappings from heterogeneous data sources.

Chapter 6

The Architecture of WISE

This chapter presents a high-level schema integration architecture, hereinafter referred to as WISE (an acronym for Web Integration System for E-commerce), and describes its basic components that make it "wise" for the integration of e-commerce systems [AE04a]. WISE consists of four processes, namely the input process, integration process, output process, and query process. Chapter Six presents these processes in different sections. Section 6.1 examines the input process with a view to extracting object semantics from the local data sources. Section 6.2 examines the integration process. In the integration process, a graph-based integration algorithm uses one or more context schemas as input to produce a global integrated schema (*GIS*). A detailed discussion of the algorithm, including its correctness and efficiency is also presented. Section 6.3 discusses the output process and the *GIS*. Here, the need is to show that the *GIS*: 1) is a correct representation of the local schemas; 2) completely represents the local schemas as all the elements in the local schemas are also in the global schema; 3) minimally represents the local schemas as each unique element in the local schema is defined no more than once in the global schema; and 4) is understandable because it can be represented using easily readable and simple XML format. In Section 6.4, the query process is described [AE05]. The query process decomposes users' queries against the *GIS*; maps the decomposed queries to the appropriate data sources; and integrates the query results from the data sources to form the effective result of the query. Finally, Section 6.5 considers some integration examples to illustrate how the integration algorithm works.

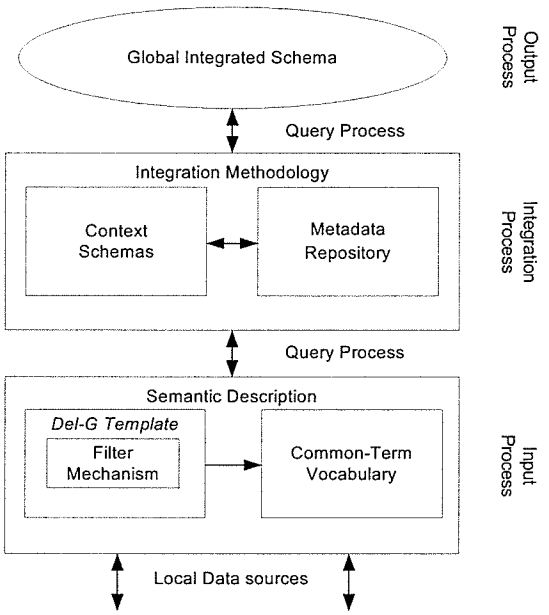

Figure 6.1: WISE Architecture

WISE is middleware that provides interoperability between applications through a client Web browser. The client Web browser provides a user-friendly graphical interface for submitting queries and exploring results. In WISE, the integration algorithm dynamically integrates context schemas and metadata from local data sources to produce the *GIS*. Figure 6.1 shows the four main component processes of WISE: the input process, integration process, output process, and query process. WISE adopts a two-phase "input-and-process" integration strategy (as discussed in Chapter 4) that separates the input process from the integration process. This strategy facilitates scalability as it permits new data sources to be added and old ones to be removed without necessarily altering the existing integration rules. The first phase involves extracting the semantics of data objects from the local data sources to form context schemas, where every context label in the context schema is also in the *CTV*. In the second phase, the integration algorithm uses context schemas and their corresponding metadata as input. The integration algorithm dynamically identifies assertions, determines attribute relationships, and generates correspondence rules that map the context schemas into the global integrated schema (*GIS*) (the sequence of activities in WISE is shown in the activity diagram of Figure 6.2). The user poses his/her query against the *GIS* after the *GIS* have been generated. This architecture does not rely on the DBA to specify the interschema

correspondence assertions, as it dynamically identifies assertions, and analyzes such assertions to produce integration rules that generate the *GIS*.

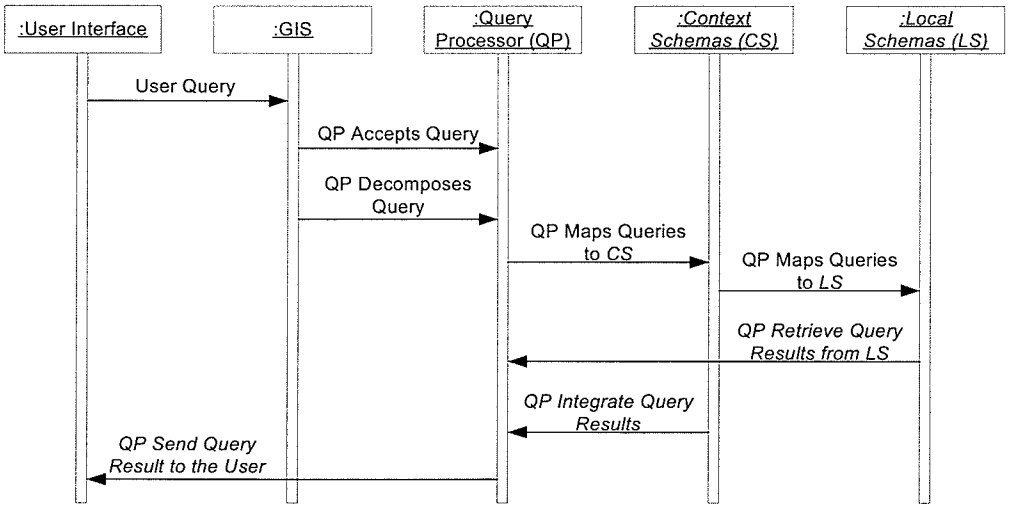

Figure 6.2: A Sequence Diagram for WISE (User Query Perspective)

Figure 6.2 shows a sequence diagram for the user query perspective of the WISE architecture of Figure 6.1. The user poses his/her query on the *GIS* through the query interface provided by the output process. The query processor accepts the query and decomposes it into sub-queries. These sub-queries are mapped to the local schemas through the context schemas. The query processor retrieves the results of the sub-queries, integrates these results to form the effective result of the original query. The query processor then sends the result to the user. A detailed description of WISE follows.

6.1 The Input Process

The aim of the input process is to support a system that ensures the autonomy of the data sources, while providing suitable input to the integration process. The input process extracts objects' semantics from data sources using the *CTV* and filter mechanism. The corresponding context schemas make the semantics of objects in the data sources explicit to facilitate dynamic identification of correspondence assertions by the integration algorithm. The input process forms the first phase of the two-phase integration process.

Details of the extraction process, including the *CTV* and the two-phase integration process were explained in Chapter 4.

6.2 The Integration Process

In the integration process, an integration algorithm dynamically identifies assertions, determines attribute relationships, and generates correspondence rules without costly user and inconvenient intervention. Therefore, the major task in this algorithm is to dynamically resolve heterogeneities among local schemas with a view to producing a *GIS* that guarantees data transparency across data sources. This research discusses the integration of two context schemas at a time. The integration of more than two context schemas involves an accumulation strategy where all participating local schemas are first integrated in pairs. This process is repeated with the integrated schemas until a global integrated schema is generated.

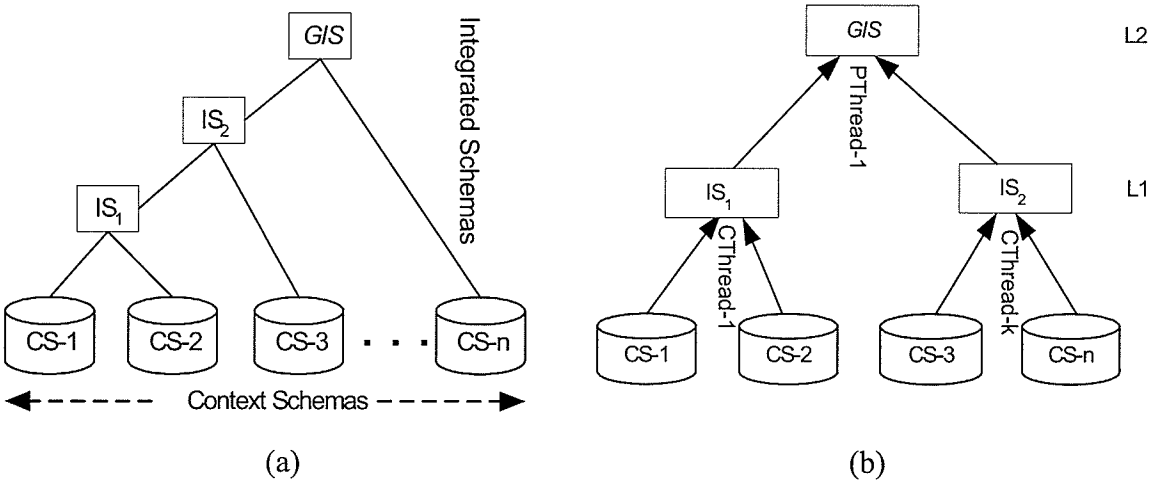

Figure 6.3: Schema Integration Trees

Figure 6.3(a) shows a schema integration tree (serial processing) where context schemas CS_1, CS_2 ... CS_n corresponding to local data sources S_1, S_2 ... S_n are integrated in pairs. First, two context schemas, CS_1 and CS_2 are integrated to produce an integrated schema, IS_1. Then context schema CS_3 and IS_1 are integrated to produce IS_2. This process continues until CS_n and IS_{n-1} are integrated to produce the final *GIS*.

Our integration approach can also benefit from parallelism where different pairs of participating context schemas are integrated in parallel. For example, in Figure 6.3(b) we have k threads (or processes) for the n context schemas at Level 1 (where each thread integrates two context schemas). These threads operate in a child-parent fashion, where two child threads (CThreads) generate a pair of integrated schemas (IS_1 and IS_2) for its parent. At Level 2, the parent threads (PThreads) integrate pairs of the integrated schemas to generate the *GIS*. There are possibilities of more parallelism in this approach (e.g. using different threads to identify specific assertions). However, the overheads incurred in communication might out weigh any likely benefits.

In Phase 2 of the two-phase integration process, context schemas are used as input to a graph-based integration algorithm that dynamically identifies correspondence assertions, and generates integration rules after analyzing the assertions. The goal of this approach is to generate a global view of the participating data sources that is transparent, at least from the users' point of view. Transparency in this context refers to both logical and physical transparency. The user should see the global view as a common and reliable interface that is local to a particular user. The integration algorithm uses context schemas and metadata repository as input. Context schemas have been extensively discussed in Chapter Four. The remaining part of this section discusses the metadata repository and the integration algorithm itself.

6.2.1 Metadata Repository

Metadata is information about data used to enhance its documentation and exchange. The metadata repository contains information about the constraints on objects from the local data sources. The source metadata contains information about data sources and the relationships and dependencies that exists between schemas. Such constraints have been broadly classified as integration constraints (as discussed in Chapter Five). The integration mechanism leverages information about schema transformations and the semantic understanding of the data sources from the *CTV* to dynamically identify assertions between objects from different data sources.

6.2.2 The Global Integrated Schema Algorithm (*GIS*-Algorithm)

To simplify our exposition and clarify the basic concepts used in this algorithm, we define the following functions, namely *Locate*, *Merge*, *and Compare*. These definitions are used to enhance readability and understandability of the algorithm.

The function *Locate* locates an element L_i in a context schema CS_i using a breadth-first search strategy [CLRS01]. If the search is successful the function returns "True". It returns "False" otherwise. The definition of *Locate* follows.

$$\begin{aligned} &\textit{Locate}: \text{LABEL} \times \text{SCHEMA} \\ &\forall L_1: \text{LABEL} \bullet (\exists\, CS_1, CS_2: \text{SCHEMA} \mid L_1 \in CS_1) \bullet \\ &\quad \textit{Locate}(L_1, CS_2) = \text{True} \Leftrightarrow \\ &\quad\quad \exists L_2{}^q{}_j: \text{LABEL} \mid L_2{}^q{}_j \in CS_2 \bullet L_1 = L_2{}^q{}_j) \end{aligned} \tag{6.1a}$$

The function *Merge* merges two elements $L_1 \in CS_1$ and $L_2 \in CS_2$ into $L \in GIS$. Two elements from the local schemas are merged into the *GIS* if and only if the two elements are equal. Therefore, if L_1 and L_2 are merged into L, it implies that $L = L_1 = L_2$, and whenever L is accessed in the *GIS*, there is a mapping to L_1 and L_2 in CS_1 and CS_2 respectively.

$$\begin{aligned} &\textit{Merge}: \text{LABEL} \times \text{LABEL} \rightarrow \text{LABEL} \\ &\forall\, L, L_1, L_2: \text{LABEL} \bullet \\ &\quad (\exists\, CS_1, CS_2: \text{SCHEMA} \mid L_1 \in CS_1 \wedge L_2 \in CS_2 \bullet \\ &\quad\quad \textit{Merge}(L_1, L_2) = L \Leftrightarrow (\exists GIS: \text{SCHEMA} \mid L \in GIS \bullet \\ &\quad\quad\quad L = L_1 \wedge L = L_2) \end{aligned} \tag{6.1b}$$

The function C*ompare* is a boolean function that compares child elements in *Edge-Child*(L_1) with child elements in *Edge-Child*(L_2) to determine the appropriate assertion. *Enqueue* and *Dequeue* are standard queue functions used, respectively, to add elements into a queue and remove elements from a queue.

Dequeue(*IQ*) A function that removes the pair of elements $(L_1{}^p{}_i, L_2{}^q{}_j)$ from *IQ*.

Enqueue(*IQ*, $L_1{}^p{}_i$, $L_2{}^q{}_j$) A function that adds the pair of elements $(L_1{}^p{}_i, L_2{}^q{}_j)$ to *IQ*.

Inheritance($L_1{}^p{}_i$) A function that maps $L_1{}^p{}_i \in CS_1$ into $L^r{}_k \in GIS$ such that $L^r{}_k = L_1{}^p{}_i \wedge \textit{Descendant}(L_1{}^p{}_i) = \textit{Descendant}(L^r{}_k)$.

Every other concept used in this algorithm is explained in context.

GIS-Algorithm
Inputs: CS_1, CS_2 // (λ_1, λ_2: entry point labels)
Output: *GIS* // (Global Integrated Schema)
Variables:: *IQ*: *Queue*; L, L_1, L_2: *Label* | L ∈ *GIS*; $L_1 \in CS_1$; $L_2 \in CS_2$

```
1.   begin
2.   λ = (λ1, λ2)  // entry point labels for CS1 and CS2 respectively
3.   Enqueue(IQ, λ1, λ2) // enqueue the entry point labels λ1 and λ2 into IQ
4.   while IQ not empty do
5.     (L1, L2) = Dequeue(IQ)
6.     if (L1 = L2) then
7a.      for all (Edge-Child (L1) and Edge-Child(L2)) do
7b.         Enqueue(IQ, Edge-Child(L1), Edge-Child(L2))
7c.         Compare(Edge-Child(L1), Edge-Child(L2))
8.     switch(L1 Ψ L2) // Ψ::= Identity | Equivalence | Intersection | Compatible
10.      case L1 Identity L2
11.         Merge(L1, L2, L) // merge L1 ∈ CS1 and L2 ∈ CS2 into L ∈ GIS
12.         dequeue any pair containing L1 or L2 from IQ
13.         for all (Edge-Child(L1) and Edge-Child(L2)) do
14.            Merge(L1^p_i, L2^q_j, L^r_k) // merge L1^p_i ∈ CS1 and L2^q_j ∈ CS2 into L^r_k ∈ GIS
15.            dequeue any pair containing L1^p_i or L2^q_j from IQ
16.            Enqueue(IQ, Edge-Child(L1^p_i), Edge-Child(L2^q_j))
           break;
17.      case L1 Equivalence L2
18.         Merge(L1, L2, L)
19.         dequeue any pair containing L1 or L2 from IQ
20.         for each pair (Edge-Child(L1) ≠ Edge-Child(L2)) do
21.            Inheritance(L1^p_i, L^r_k) // inherit L1^p_i and its descendants into L^r_k ∈ GIS
22.         for each pair (Edge-Child(L2) ≠ Edge-Child(L1)) do
23.            Inheritance(L2^q_j, L^r_k) // inherit L2^q_j and its descendants into L^r_k ∈ GIS
24.         for all (Edge-Child(L1) = Edge-Child(L2)) do
25.            Merge(L1^p_i, L2^q_j, L^r_k)
26.            dequeue any pair containing L1^p_i or L2^q_j from IQ
27.            Enqueue(IQ, Edge-Child(L1^p_i), Edge-Child(L2^q_j))
           break;
28.      case L1 Compatible L2
           break;
29.      case L1 Intersection L2
           break;
       else
30.      if (Edge-Parent(L1) ≠ ∅ and Edge-Parent(L2) ≠ ∅) then
31.         Inheritance(L1^p_i, L^r_k)
32.         Inheritance(L2^q_j, L^r_k)
         else
33.         Locate(L1, CS2); Locate(L2, CS1); // locate L2 in CS1 and locate L1 in CS2
34.    switch(L1 Ψ L2) // Inclusion | Mutual-Inclusion | Exclusion
35.      case L1 Mutual-Inclusion L2
```

36. (L_1, $L_2{}^q{}_j$) into L ∈ *GIS* (where $L_2{}^q{}_j = L_1$)
37. dequeue any pair containing L_1 or $L_2{}^q{}_j$ from *IQ*
38. *Enqueue*(*IQ*, *Edge-Child*(L_1), *Edge-Child*($L_2{}^q{}_j$)) // except the pair containing // $L_1{}^p{}_i$ ($L_1{}^p{}_i = L_2$)
39. *Merge*($L_1{}^p{}_i$, L_2, $L^r{}_k$) // (where $L_2{}^q{}_j = L_1$)
40. dequeue any pair containing $L_1{}^p{}_i$ or L_2 from *IQ*
41. *Enqueue*(*IQ*, *Edge-Child*($L_1{}^p{}_i$), *Edge-Child*(L_2)) //except the pair containing // $L_2{}^q{}_j$ ($L_2{}^q{}_j = L_1$)

break;

42. case L_1 *Inclusion* L_2
44. Insert elements of L_1 into *GIS* until $L_1{}^p{}_i$ is encountered
45. *Merge*($L_1{}^p{}_i$, L_2, $L^r{}_j$) // (where $L_1{}^p{}_i = L_2$)
46. dequeue any pair containing L_1 or $L_2{}^q{}_j$ from *IQ*
47. *Enqueue*(*IQ*, *Edge-Child*($L_1{}^p{}_i$), *Edge-Child*(L_2))
48. for all elements (except $L_1{}^p{}_i = L_2$) at level *i* do
49. *Inheritance*($L_1{}^p{}_i$, $L^r{}_k$)

break;

50. case L_2 *Inclusion* L_1 // same as L_1 *Inclusion* L_2, change the L_1 and L_2

break;

51. case L_1 *Exclusion* L_2 //Apply correspondence rules for exclusion assertion

break;

52. end // while
53. end // begin

Figure 6.4: Schema Integration Algorithm

The main operation of the *GIS*-Algorithm is to compare and merge elements using a top-down approach, starting with the entry-point labels. Let *d* be the degree of each element, and *p* and *q* represent the parents of a particular element in CS_1 and CS_2 respectively. $L_1{}^p{}_i$ and $L_2{}^q{}_j$ represent elements of CS_1 and CS_2 respectively (where $0 \leq i, j \leq d$, and p, q: LABEL). For example, $L_1{}^p{}_i$ represents the *i*th element whose parent is *p* in CS_1. For the entry-point labels (λ_1, λ_2: LABEL $|\lambda_1, \lambda_2 \in L_{EP}$, where L_{EP} is the set of entry-point labels), $L_1{}^p{}_i = L_1$ and $L_2{}^q{}_j = L_2$ because $p = q = \varnothing$ and $i = j = 0$. In *Del-G*, the element represented by the entry-point label has no parent. The algorithm uses an integration queue (*IQ*) to manage a breadth-first search of the two input schemas, CS_1 and CS_2, as in [Che00]. However, unlike [Che00, SK92], the *GIS*-Algorithm does not involve the DBA or user in the identification of assertions and determination of correspondence rules. The algorithm leverages the inheritance property to isolate inherited elements from the search space (lines 21, 23, 31, 32). The elements in *IQ* are pairs of labels from CS_1 and CS_2, where the initial head node in *IQ* is a pair consisting of the entry-point labels

(λ_1, λ_2) from CS_1 and CS_2, respectively (line 3). The head of *IQ* is dequeued into L_1 and L_2 (line 5), and checked for possible assertions (line 6). If L_1 and L_2 are equal, then the algorithm looks ahead to the children of L_1 and L_2 ($L_1{}^p{}_i$, $L_2{}^q{}_j$) (where $p = q$) to identify the particular assertion (lines 7a, 7b, and 7c). If L_1 and L_2 are equal, then the probable assertions to be identified regardless of the relationship that exists between the child elements are *Identity*, *Equivalence*, *Intersection*, or *Compatibility* assertions (line 8); otherwise, the probable assertions are *Inclusion*, *Mutual-Inclusion*, or *Exclusion* assertions (line 34). Notice that there are two separate *switch* statements in the *GIS*-Algorithm of Figure 6.4. The algorithm is partitioned to permit the first switch statement to control the first set of assertions and the second switch statement to control the second set of assertions.

Once a particular assertion is established, merging occurs according to the corresponding integration rules (lines 11, 18, 25, 36, 39, and 45). A call to the function *Merge* produces the correspondence rules and merges the elements of the context schemas into the *GIS*. Every other pair in *IQ* containing any of the merged elements is also dequeued (lines 12, 19 26, 37, 40, and 46). The child elements of each merged pair ($L_1{}^p{}_i$, $L_2{}^q{}_j$) are then enqueued into *IQ* (lines 13, 20, 27, 38, 41, and 47) and the process is repeated. Otherwise, there is a call to the function *Inheritance* to inherit any child element whose parent is in *GIS*, but has no assertion with other elements ($c_i \in$ *Edge-Child*($L_1{}^p{}_i$) • $c_i \notin$ *Edge-Child*($L_2{}^q{}_j$)) $\vee$ ($c_j \in$ *Edge-Child*($L_2{}^q{}_j$) • $c_j \notin$ *Edge-Child* ($L_1{}^p{}_i$)).

An element with the label $L^r{}_k \in$ *GIS* inherits its descendants from a corresponding local schema CS_1, where there is no assertion between a child element $L_1{}^p{}_i \in CS_1$ and other elements in CS_2 that share the same parent with $L_1{}^p{}_i$ in *GIS*.

Theorem 6.1 The inheritance property removes inherited elements from the search space.

Proof. Let $L^r{}_k \in$ *GIS* inherit *Descendants*($L_1{}^p{}_i \in CS_1$) $\Rightarrow L^r{}_k = L_1{}^p{}_i \vDash r = p$. Suppose there exists an element $q \in CS_2$, such that $q = p$, then no child element of q, *Edge-Child*($L_2{}^q{}_j$) $\in CS_2$ ($1 \leq j \leq d$) is equal to $L_1{}^p{}_i$. By the *GIS*-Algorithm, all pairs of elements of the form ($L_1{}^p{}_i$, $L_2{}^q{}_1$), ($L_1{}^p{}_i$, $L_2{}^q{}_2$), …, ($L_1{}^p{}_i$, $L_2{}^q{}_d$) are dequeued from *IQ*. In addition, no *Edge-Child*($L_1{}^p{}_i$) is enqueued into *IQ*, since pairs of child elements

whose parents are not equal are not enqueued into *IQ* (every pair that contains $L_1{}^p{}_i$ is disjoint). Using a similar argument, *Edge-Child*($L_1{}^p{}_i$) $\notin$ *IQ* $\models$ *Descendants*($L_1{}^p{}_i$) $\notin$ *IQ*. Therefore, *Descendants*($L_1{}^p{}_i$) $\notin$ *IQ*, and are not compared since elements can be compared by the algorithm if and only if they are enqueued in *IQ*. ■

The second switch is executed if L_1 and L_2 are entry-point labels, and they are not equal. A call to function *Locate* searches through CS_2 to locate L_1 and CS_1 to locate L_2. If *Locate* finds L_1 in CS_2 it returns "true" and the location where it was found, otherwise it returns "false". Similarly, if *Locate* finds L_2 in CS_1 it returns "true", otherwise it returns "false". If both *Locate* L_1 in CS_2 and L_2 in CS_1 return "true", then there exists a mutual inclusion assertion; if both return "false", then there exist an exclusion assertion; otherwise, it is an inclusion assertion. The above algorithm establishes the seven correspondence assertions, and for each of the assertions the algorithm generates a corresponding set of integration rules. This algorithm handles the identification and integration of inclusion assertion and its variants in a flexible manner. It leverages the inheritance property of elements to ensure that the *GIS* maintain existing local relationships among elements.

6.2.3 Analysis of the *GIS*-Algorithm

Asymptotic Analysis of the Algorithm We now highlight some features of *Del-G* that make it amenable to the *GIS*-Algorithm, and easier to explain its analysis. Every element in a given schema has a unique name which is defined at most once. Notice that elements in *Del-G* use a *CTV*, therefore, every element name in CS_1 has at most one element name in CS_2 that corresponds to it. *Del-G* exploits the parent-child relationship, and every child has at most one parent. For the most part, the algorithm compares and merges elements using a top-down approach, starting with the entry point labels (L_1, L_2). A call to *Locate* only occurs if the starting pair (L_1, L_2) is not equal, and it executes once whenever the integration algorithm is executed. To properly analyse the behaviour of the algorithm, let us examine three distinct scenarios that subsume the other behaviours of the algorithm.

Case 1 For *Identical Assertions*, for every child element of $L_1 \in CS_1$ there is a corresponding element in CS_2 such that starting with the pair of entry point labels (L_1, L_2), $L_1 \in CS_1$ and $L_2 \in CS_2$, $L_1 = L_2$. Initially, the pair (L_1, L_2) is enqueued and compared for assertion. Subsequently, all child elements of L_1 and L_2 (i.e., ($L_1{}^{L1}{}_1$, $L_2{}^{L2}{}_1$), …, ($L_1{}^{L1}{}_1$, $L_2{}^{L2}{}_d$), …, ($L_1{}^{L1}{}_d$, $L_2{}^{L2}{}_1$), …, ($L_1{}^{L1}{}_d$, $L_2{}^{L2}{}_d$)) are enqueued and compared, since their parents are equal, the child elements are compared. Suppose for every $L_1{}^p{}_i \in CS_1$ there is a corresponding $L_2{}^q{}_j \in CS_2$ (where $p = q$), then for each pair ($L_1{}^p{}_i$, $L_2{}^q{}_j$) their respective child elements are also enqueued and compared until $i = d$ and $j = d$. Let T_1 be the number of all edges in CS_1 and T_2, the number of all edges in CS_2. Let E_S be the number of all the pairs of edges in (CS_1, CS_2), then

$$E_S = \sum_{i=1}^{T1} \sum_{j=1}^{T2} (L_1{}^p{}_i, L_2{}^q{}_j) \tag{6.2}$$

It can be seen from the above that the sum of the pairs of elements (since every edge represents an element) in the two schemas is $\Theta(E_S)$, which is the "cost" of enqueuing and dequeuing those elements is $O(E_S)$. In a strict *Identity Assertion*, every element in CS_1 has a corresponding element in CS_2. In such a case, every edge in *Del-G* will be enqueued and compared. The cost of comparing all the pairs of elements in any given *Del-G* is $O(E_S)$. Therefore, the algorithm takes $O(E_S + E_S) = O(2E_S) = O(E_S)$, where E_S is the number of edges of (CS_1, CS_2).

In Case 1, the query processor does not have to decompose a given query into many different sub-queries since the context schemas are identical. This is a potential source of optimization.

Case 2 For *Exclusion assertion*, which occurs when the pair (L_1, L_2) is not equal, the function *Locate* is called to locate L_1 in CS_2 and L_2 in CS_1. If $L_1 \notin CS_2$ and $L_2 \notin CS_1$, then there will be no further scanning of child elements of L_1 and L_2, because no relationship exists between child elements (where their parents are unrelated). The empty string is then used as the root element which inherits *Descendant*(L_1), and *Descendant*(L_2). The inheritance operation does not involve any comparison. A call to locate only occurs if the starting pair (L_1, L_2) is not equal, and it is executed only once whenever the integration algorithm is executed. In this case, only one pair is enqueued into *IQ* and scanned. The

cost of this operation is $O(E_S)$. It also takes linear time, $O(E_S)$, to search CS_2 for L_2. The effective running time is $O(E_S + E_S) = O(E_S)$. Therefore, the algorithm runs in linear time in the size of E_S.

Theorem 6.2 Let ${L_1}^p{}_i$ and ${L_2}^q{}_j$ be elements in CS_1 and CS_2 respectively. Suppose $\alpha \in$ *Edge-Child*(${L_1}^p{}_i$) and $\beta \in$ *Edge-Child*(${L_2}^q{}_j$), then $\alpha \Psi \beta$ if and only if $p = q$.

Proof. Let Ψ::= *Identity* | *Equivalence* | *Intersection* | *Compatible* | *Inclusion* | *Mutual-Inclusion*. Notice that the exclusion assertion is not in Ψ because it can only occur with entry-point labels. Suppose ${L_1}^p{}_i \ \Psi \ {L_2}^q{}_j$, where ${L_1}^p{}_i \in CS_1$, ${L_2}^q{}_j \in CS_2$, and $p \neq q$. Let (${L_1}^p{}_i \ \Psi \ {L_2}^q{}_j$) be merged into ${L^r}_k \in GIS$ by a call to Merge(${L_1}^p{}_i$, ${L_2}^q{}_j$) in the *GIS*-Algorithm. This implies that $r \in GIS = p \in CS_1$ and $r \in GIS = q \in CS_2 \Rightarrow p = q$. This is a contradiction since each element in CS_1 and CS_2 is uniquely defined. Suppose $\alpha \in$ *Edge-Child*(${L_1}^p{}_i$) and $\beta \in$ *Edge-Child*(${L_2}^q{}_j$), then $\alpha \ \Psi \ \beta \Rightarrow \alpha = \beta$. This implies that there exist one $\gamma \in GIS$ such that ($\gamma = \alpha \wedge \gamma = \beta$). If $\gamma \in$ *Edge-Child*(${L^r}_k$), then ${L^r}_k = {L_1}^p{}_i \wedge {L^r}_k = {L_2}^q{}_j \vDash r = p$ and $r = q \Rightarrow p = q$. This is a contradiction since $p \neq q$, which means r is not uniquely defined. That r is not uniquely defined contradicts the assumption of a well-formed schema, and hence establishes the proof that no assertion exists between child elements whose parents are unrelated. ■

Case 3 Suppose the pair (L_1, L_2) is not equal, then the function *Locate* is used to locate L_2 in CS_1 and L_1 in CS_2. Suppose there is an ${L_1}^p{}_i \in CS_1$ that is equal to L_2, and there is no ${L_2}^q{}_j \in CS_2$ such that ${L_2}^q{}_j = L_1$, then $L_1 \supseteq L_2$. In this case, every element of CS_1 is merged into the *GIS* until ${L_1}^p{}_i = L_2$ is encountered. Then, the pair (${L_1}^p{}_i$, L_2) is merged into the *GIS*, and their child elements are enqueued into *IQ* and compared. All elements in the set of pairs where there is no match, the descendants of such elements are inherited into the *GIS* otherwise if there is a match, then all the pairs consisting of their child elements are also enqueued and compared. This process continues until all the elements have been considered. Notice, that no pair is enqueued more than once in the *IQ*. The cost of searching CS_1 and CS_2 is $O(E_S)$. The cost of enqueuing and comparing the pairs of edges of CS_1 and CS_2 is also $O(E_S)$. The effective running time, therefore, is $O(E_S)$.

Overhead of the *GIS* Algorithm The *GIS*-Algorithm is dominated by certain operations that we explain below. Queuing and dequeuing are dominant operations in the algorithm. The operations for both queuing and dequeuing take $O(1)$ time, so the total time devoted to queue operations is $O(E_S)$ since every pair can be queued at most once. Searching is another dominant operation in the algorithm. The algorithm uses a breadth-first search (BFS) strategy to search the pairs of elements in the integration queue. The inheritance property uses a depth-first search strategy to identify and isolate inherited elements from the search space. BFS takes $O(E_S + E_S)$ to execute, while DFS takes $O(E_S)$ time to execute. Another dominant operation of the algorithm is merging. Merging mainly involves writing out the *GIS* based on a given assertion. This operation takes $O(1)$ time, so the total time for the merge operation is $O(E_S)$. Comparison is also a dominant operation that takes $O(1)$ time for its execution. Thus, the *GIS*-Algorithm runs in time linear to E_S.

Correctness of the Algorithm The correctness of the algorithm relies on the fact that every element in both CS_1 and CS_2 is considered; either compared directly or inherited. Inherited elements of CS_1 need not be compared in CS_2 because they are guaranteed not to occur by the inheritance property. Effectively, every element is compared, and the corresponding assertion is invoked. The algorithm incrementally matches terms on *Del-G* based on aligning and inheriting graph labels. In other words, the algorithm shows semantic similarity of notions by finding a match between elements in the *Del-G* trees.

Theorem 6.3 Let CS_1 and CS_2 be involved in an integration process. Suppose there is an *Inheritance*($L_1{}^p{}_i$) (where $L_1{}^p{}_i \in CS_1$) by the *GIS*, then, $\neg$ ($\exists L_2{}^q{}_j \in CS_2 \bullet L_2{}^q{}_j = L_1{}^p{}_i$).

Proof. Recall that the inheritance property states that an element $L_1{}^p{}_i \in CS_1$ can be inherited by $L^r{}_k \in GIS$ if and only if there exists $L_2{}^q{}_j \in CS_2$ such that $r = p = q$ and $L_1{}^p{}_i \neq L_2{}^q{}_j$. Suppose there exists one $L_2{}^t{}_j \in CS_2$ such that $L_1{}^p{}_i = L_2{}^t{}_j$. This means that $p = t$, and $p = q$, which implies that $t = q$. This is a contradiction, since $t \in CS_2$ cannot be equal to $q \in CS_2$ because every element in a given schema has a unique name and is defined at most once. This contradiction establishes the fact that $L_1{}^p{}_i \notin CS_2$. Suppose there exists an *Edge-Child*($L_1{}^p{}_i$) $\in CS_2$, whose parent is p_1. This

implies that $p_1 = L_1{}^P{}_i$. This is a contradiction, since it has been established that $L_1{}^P{}_i \notin CS_2$. Therefore, $L_1{}^P{}_i \notin CS_2 \vDash$ *Edge-Child*$(L_1{}^P{}_i) \notin CS_2$ and *Edge-Child*$(L_1{}^P{}_i) \notin CS_2 \vDash$ *Descendant*$(L_1{}^P{}_i) \notin CS_2$. Inherited elements of a given context schema, CS_1 are guaranteed not to occur in another context schema, CS_2. ■

6.3 The Output Process

The goal of the output process is to have a global integrated schema that is a correct, complete, minimal, and understandable representation of all the schemas in the data sources. These properties show the ability of the integration algorithm to resolve heterogeneities. An integration system that resolves heterogeneities takes away the burden of identifying assertions from the users. An effective integration methodology must guarantee that an integration system allows a non-redundant, unified representation of all data managed by the participating data sources. A formal definition of a *GIS* follows.

Definition 6.1 A *Global Integrated Schema* (*GIS*) over a set of context schemas CS_{SET} | $CS_{SET} = \bigcup_{i=1}^{n} CS_i$ is a platform independent *Del-G* schema where $\forall L_1{}^P{}_i$: LABEL • ($\exists CS_{SET}$: $\mathbb{P}$SCHEMA • $L_1{}^P{}_i \in CS_{SET} \wedge$ *Edge-Parent*$(L_1{}^P{}_i) = p) \Rightarrow \forall L^r{}_k$: LABEL • ($\exists GIS$: SCHEMA • $L^r{}_k \in GIS) \wedge L_1{}^P{}_i = L^r{}_k$. ■

An effective integration methodology must guarantee that the *GIS* is a unified representation of all the data sources (without redundancies). The subsections that follow show that the *GIS* is a complete, correct, minimal and understandable representation of all the context schemas.

Theorem 6.4 The *GIS* is a well-formed schema.

Proof. If the *GIS* is a well-formed schema then; 1) no two siblings in the *GIS* are the same; and 2) every node (excluding the root node) has at most one parent. Let $CS_i \in CS_{SET}$, where CS_i is a context schema in the set of context schemas CS_{SET}.
Suppose *GIS* is not a well-formed schema.

1. This means there exist elements $x, y \in GIS$ such that $x = y \wedge Edge\text{-}Parent(x) = Edge\text{-}Parent(y)$. Now $x, y \in GIS \Rightarrow \exists_1 CS_i \in CS_{SET} \bullet x, y \in CS_i \vee x, y \in CS_{SET}$ (x, y is in at least two different schemas in CS_{SET}). If $x, y \in CS_i \wedge Edge\text{-}Parent(x) = Edge\text{-}Parent(y)$, then, CS_i is not a context schema. But, the filter mechanism accepts only context schemas. This implies that no such $x, y \in CS_i$ where, $Edge\text{-}Parent(x) = Edge\text{-}Parent(y)$. $x, y \in CS_{SET} \Rightarrow \exists\ CS_i, CS_j \in CS_{SET} \bullet (x \in CS_i \wedge y \in CS_j) \vee (y \in CS_i \wedge x \in CS_j) \wedge Edge\text{-}Parent(x) = Edge\text{-}Parent(y)$. But the integration algorithm compares every element to identify assertions. So, if there exist $x, y \in CS_{SET} \wedge Edge\text{-}Parent(x) = Edge\text{-}Parent(y)$, the integration algorithm would have identified the assertion and merged them into the *GIS* as a single element. Therefore, $x, y \notin CS_{SET} \bullet Edge\text{-}Parent(x) = Edge\text{-}Parent(y)$. A contradiction! Hence, *GIS* is a well-formed schema.
2. Suppose there exists an $x \in GIS \bullet x \in Edge\text{-}Child(p_1) \wedge x \in Edge\text{-}Child(p_2) \wedge p_1 \neq p_2$. This implies that there exists a $CS_i \in CS_{SET} \bullet \exists x, p_1, p_2 \in CS_i \mid x \in Edge\text{-}Child(p_1) \wedge x \in Edge\text{-}Child(p_2) \wedge p_1 \neq p_2$. This means that x has more than one parent. But CS_i is a context schema, and every element in CS_i has at most one parent. Therefore, $\neg(\exists x \in CS_i \bullet x \in Edge\text{-}Child(p_1) \wedge x \in Edge\text{-}Child(p_2) \wedge p_1 \neq p_2)$. Suppose $\exists\ CS_i, CS_j \in CS_{SET} \bullet \exists x, p_1 \in CS_i \mid x \in Edge\text{-}Child(p_1) \wedge \exists x, p_2 \in CS_j \mid x \in Edge\text{-}Child(p_2) \wedge p_1 \neq p_2$. The integration algorithm cannot integrate x into the *GIS* as child of p_1 and p_2 unless $p_1 = p_2$. If $p_1 \neq p_2$, then, x can only be inherited by its parent into the *GIS*. Therefore, $\neg(\exists x \in GIS \bullet x \in Edge\text{-}Child(p_1) \wedge x \in Edge\text{-}Child(p_2) \wedge p_1 \neq p_2)$. Hence, *GIS* is a well-formed schema. ■

Theorem 6.4 and the definitions of *data-path* and *path-label* are now used to establish the completeness of the *GIS*.

6.3.1 Completeness

Completeness stipulates that the *GIS* completely represents the local schemas such that all the elements in the local schemas are also represented in the global schema. That

means every data path dp_i in the power set of the local schemas CS_{SET} is also **in**[43] the sequence of a data path dp_j in the *GIS*. In addition, relationships that exist between objects in the context schemas also hold in the *GIS*. It suffices to show that every *Data-Path* (*dp*) in the local schemas (CS_i) is also in the *GIS*. Formally,

$$\begin{aligned}
&\forall dp_i\text{: DATA-PATH} \bullet \\
&\quad \exists\, CS_{SET}\text{: } \mathbb{P}_1\text{SCHEMA} \bullet dp_i \in CS_{SET} \Rightarrow \\
&\qquad (\forall dp_j\text{: DATA-PATH} \bullet \exists GIS\text{: SCHEMA} \bullet dp_j \in GIS \wedge \\
&\qquad\quad dp_i \textbf{ in } dp_j \vDash dp_i \in GIS) \wedge \\
&\qquad\quad \forall u, v\text{: NODE} \mid u, v \textbf{ in } dp_i \bullet \mathit{Edge\text{-}Parent}(v) = u) \Rightarrow \\
&\qquad\qquad (\exists u_1\text{: NODE} \mid u_1 \textbf{ in } dp_j \bullet \mathit{Edge\text{-}Parent}(v) = u_1 \Leftrightarrow u = u_1 \wedge v \textbf{ in } dp_j)
\end{aligned} \tag{6.3}$$

Miller, *et al.* [MIR93] compare two schemas based on information capacity. The information capacity of a schema *S* determines its contents. If f: $CS_1 \rightarrow CS_2$ is a mapping from CS_1 to CS_2 without loss of information, where f is both injective and total[44], then f is an information capacity preserving mapping, and CS_2 has more information capacity than CS_1 (i.e., CS_2 dominates CS_1 through f, written, $CS_1 \preccurlyeq CS_2$).

Corollary 6.1 Let Ω be a function (correspondence rule) that maps elements from the power set of context schemas, CS_{SET}, to the *GIS*. Ω: $CS_{SET} \rightarrow GIS$ is an information capacity preserving mapping.

Proof. Recall, from (6.3) every $dp_i \in CS_{SET} \Rightarrow dp_i \in GIS$. By the definition of a *data-path*, data paths in *Del-G* represent elements of *Del-G*. Therefore, Ω maps every element in CS_{SET} to *GIS*, which implies that the function Ω is *total*. By Theorem 6.4, every element in CS_{SET} is defined no more than once in the *GIS* since *GIS* is a well-formed schema. Therefore, the function Ω is *injective*. Ω: $CS_{SET} \rightarrow GIS$, therefore, is an information capacity preserving mapping since it is both injective and total. By implication, *GIS* dominates CS_{SET} through Ω, denoted as $CS_{SET} \preccurlyeq GIS$. ■

---

[43] *A* **in** *B* means that *A* is a contiguous sequence of *B* (where *A* and *B* are sequences, and **in** is the standard sequence inclusion operator).

[44] If *A* and *B* are sets, a mapping F: $A \rightarrow B$ is *injective* if for any $a \in A$, $\exists_1 b \in B \bullet F(a) = b \wedge F^{-}(a) = a$. F: $A \rightarrow B$ is *total* if for every $a \in A$, $\exists_1 b \in B \bullet F(a) = b$.

6.3.2 Correctness

The *GIS*, apart from being a complete representation of the local schemas, should also correctly represent the local schemas. Having all elements in CS_{SET} in the *GIS* is necessary, but insufficient to satisfy the correctness criteria. Context schema elements must be correctly represented in the *GIS* with all their semantic relationships maintained. It is necessary to show that the results obtained by querying the different data sources directly and querying them through the *GIS* are the same using an equivalent set of queries.

Suppose $\mathbb{Q}$: $S \rightarrow \mathcal{R}$ where S and $\mathcal{R}$ are schema and result types, respectively. Let $\mathbb{Q}$ be a query over the *GIS* that generates the result, $\mathcal{R}_{GIS}$. If the same query $\mathbb{Q}$ is posed over the union of context schemas $CS_{\text{SET}} = \bigcup_{i=1}^{n}(CS_i)$, then the query processor decomposes $\mathbb{Q}$ into a unique set of queries $\{q_1, q_2, \ldots, q_n\}$ and maps them to the set of context schemas $\{CS_1, CS_2, \ldots, CS_n\}$ such that $\sum q_i = \mathbb{Q}$, $(i = 1 \ldots n)$ and $CS_{\text{SET}} \preccurlyeq GIS$. Suppose $\{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_n\}$ are results corresponding to $\{q_1, q_2, \ldots, q_n\}$ such that $\mathcal{R}_{\text{LS}} = \bigcup_{i=1}^{n}(\mathcal{R}_i)$, then the *GIS* is a correct representation of CS_{SET} if and only if $\mathcal{R}_{GIS} = \mathcal{R}_{\text{LS}}$.

Let us declare the following basic types to represent queries and result types, respectively, to facilitate a formal definition.

[QUERY, RESULT]

Formally,

$$
\begin{aligned}
&\forall \mathbb{Q}: \mathbb{P}_1\text{QUERY} \bullet \\
&\quad (\exists GIS: \text{SCHEMA} \wedge \exists \mathcal{R}_{GIS}: \text{RESULT} \bullet \mathbb{Q}(GIS) \rightarrow \mathcal{R}_{GIS} \Leftrightarrow \\
&\qquad (\exists CS_{\text{SET}} : \mathbb{P}_1\text{SCHEMA} \bullet CS_{\text{SET}} \preccurlyeq GIS) \wedge \\
&\qquad\quad (\forall q_i: \text{QUERY} \bullet \exists\, CS_i: \text{SCHEMA} \bullet ((\bigcup_{i=1}^{n} q_i = \mathbb{Q} \wedge \bigcup_{i=1}^{n} CS_i = CS_{\text{SET}}) \wedge \\
&\qquad\qquad (\exists \mathcal{R}_i: \text{RESULT} \bullet q_i(CS_i) \rightarrow \mathcal{R}_i) \wedge \\
&\qquad\qquad\quad (\exists \mathcal{R}_{\text{LS}}: \mathbb{P}_1\text{RESULT} \mid \mathcal{R}_{\text{LS}} = \bigcup_{i=1}^{n} \mathcal{R}_i \bullet
\end{aligned}
\tag{6.4}
$$

$$(\bigcup_{i=1}^{n} q_i(CS_i) \rightarrow \mathcal{R}_{LS} \Leftrightarrow \mathcal{R}_{LS} = \mathcal{R}_{GIS})))))$$

Theorem 6.5 The global integrated schema *GIS* is a correct representation of the set of context schemas, CS_{SET}.

Proof Vacuously true from (6.4). ∎

6.3.3 Minimality

The concept of minimality stipulates that the global integrated schema is a minimal representation of the set of context schemas as each unique element in the context schema is defined not more than once in the global integrated schema. In other words, if the same concept appears in any two context schemas, that concept must be represented only once in the *GIS*, which eliminates redundancy. In general, every path label, $pl_i \in CS_{SET}$ occurs no more than once in the *GIS*. Formally,

$$\begin{aligned}
&\forall pl_i\text{: PATH-LABEL} \bullet \\
&\quad \exists\, CS_{SET} : \mathbb{P}_1 \text{SCHEMA} \mid pl_i \in CS_{SET} \Rightarrow \\
&\qquad (\forall pl_j\text{: PATH-LABEL} \bullet \exists GIS\text{: SCHEMA} \mid pl_j \in GIS \wedge \\
&\qquad\quad pl_i \textbf{ in } pl_j \vdash pl_i \in GIS) \wedge \\
&\qquad\quad (\forall CS_i, CS_j \in CS_{SET} \bullet \exists pl_i, pl_j\text{: PATH-LABEL} \mid pl_i \in CS_i \wedge \\
&\qquad\qquad pl_j \in CS_j \bullet pl_i = pl_j) \Rightarrow \\
&\qquad\qquad (\exists_1 pl_k\text{: PATH-LABEL} \mid pl_k \in GIS \bullet pl_k = pl_i \vdash pl_k = pl_j) \wedge \\
&\qquad\qquad\quad (\forall i, j\text{: } \mathbb{N} \mid i \neq j \bullet pl_i, pl_j \in GIS \Rightarrow pl_i \neq pl_j))
\end{aligned} \tag{6.5}$$

Theorem 6.6 The *GIS* minimally represents the set of local schemas, CS_{SET}.

Proof. Recall from Theorem 6.4 that the *GIS* is a well-formed schema, and from (6.5) that every path label $pl_k \in GIS$ is unique. If there exist any two path labels pl_i and $pl_j \in CS_{SET}$ such that $pl_i = pl_j$, there is only one path label $pl_k \in GIS$ such that $pl_k = pl_i = pl_j$. Therefore, the *GIS* minimally represents the elements in the set of context schemas. ∎

6.3.4 Understandability

The understandability requirement stipulates that the *GIS* should be modelable in a simple and readable format to ensure that designers and end users can understand it. *Del-*

G is simple and readable, and does not involve nested definitions. *Del-G* can therefore be easily represented using XML, which is simple to understand and easy to read. Figure 6.5 is an XML version of Figure 4.2.

```
⟨?xml version="1.0" ?⟩
⟨Schema name = "Book"
        Xmlns= "urn:schemas-microsoft-com:xml-data"
        Xmlns:dt="urn"schemas-microsft-com:datatypes"⟩
⟨ElementType name = "[Book]" sys_type = "Object" ⟩
        ⟨element type = "[Book] Title" sys_type = "Attribute" /⟩
        ⟨element type = "[Book] ISBN" sys_type = "Attribute" /⟩
        ⟨element type = "[Book;Author] Name" sys_type = "Attribute"/⟩
        ⟨element type = "[Book;Author] Id" sys_type = "Attribute"/⟩
⟨/element type⟩
⟨/Schema⟩
```

Figure 6.5: XML Representation of a *Del-G* Schema

The flexible nature of *Del-G* makes it a good framework to represent data of different formats. *Del-G*, like XML, is self-describing, and it provides a platform independent bridge between the Web and legacy data.

6.4 Query Processing

An effective integration system provides a transparent integrated schema that can be queried by different users across different platforms. The integration system must also be able to transform data from different sources to the *GIS* standard when queries are posed against the schema. The user poses a query to the existing *GIS* from the user interface using the client's Web browser. The query processor receives the queries posed by the user and initiates a plan for the decomposition of the queries to the appropriate data sources. The query processor decomposes queries into sub-queries and maps such queries to the context schemas. The query processor also computes answers to the queries and gathers the partial results to form the effective result of the query issued.

Queries against both standard databases (e.g. relational databases) and semistructured data are conceptually the same, except that in standard databases, query languages exploit the relational nature of the schema (tables) while in semistructured data

query languages exploit path expressions. Therefore, the standard SQL query format applicable to the standard database could be extended to leverage the path knowledge in the semistructured data model. Path knowledge in the semistructured data model is necessary because it improves the quality of queries passable. Semistructured query languages have the potential to reach arbitrary depths in the labelled graphs because of their ability to exploit the notion of path expressions [ABS00].

Let us define the following restriction operators to facilitate the illustration of the correctness of the integration system. The operator ($\lhd$) restricts the domain, and the operator ($\rhd$) restricts the range of relations. These operators ($\lhd$, $\rhd$) enable us to construct sets of objects that satisfy the function Υ_{GIS} and are defined as functions.

$$
\begin{aligned}
&-\lhd-:\mathbb{P}\ CS_{\text{SET}} \times (CS_{\text{SET}} \leftrightarrow GIS) \rightarrow (CS_{\text{SET}} \leftrightarrow GIS) \\
&\forall CS_i : \mathbb{P}\ CS_{\text{SET}}, \Upsilon_{GIS} : (CS_{\text{SET}} \leftrightarrow GIS) \bullet \\
&\quad CS_i \lhd \Upsilon_{GIS} = \{(L_i^p{}_j, L_k^r) \mid L_i^p{}_j \in CS_i \wedge (L_i^p{}_j, L_k^r) \in \text{Gr}(\Upsilon_{GIS})\}^{45}
\end{aligned}
\tag{6.6}
$$

The restriction imposed by the operator ($\lhd$) implies that every element in CS_{SET} is mapped to an element in *GIS* such that the domain of Υ_{GIS} is the whole of CS_{SET}. $S \lhd R$ filters $(L_i^p{}_j, L_k^r) \in \text{Gr}(\Upsilon_{GIS})$ for which $L_i^p{}_j \in CS_i$ and produces the result relation.

$$
\begin{aligned}
&-\rhd-:\mathbb{P}GIS \times (CS_{\text{SET}} \leftrightarrow GIS) \rightarrow (CS_{\text{SET}} \leftrightarrow GIS) \\
&\forall G: \mathbb{P}GIS, \Upsilon_{GIS}: (CS_{\text{SET}} \leftrightarrow GIS) \bullet \\
&\quad \Upsilon_{GIS} \rhd G = \{(L_i^p{}_j, L_k^r) \mid L_k^r \in G \wedge (L_i^p{}_j, L_k^r) \in \text{Gr}(\Upsilon_{GIS})\}
\end{aligned}
\tag{6.7}
$$

The restriction imposed by the operator ($\rhd$) implies that every element in *GIS* is mapped to an element in CS_{SET}, where the *GIS* is the range of the function, Υ_{GIS}. $\Upsilon_{GIS} \rhd G$ filters $(L_i^p{}_j, L_k^r) \in \text{Gr}(\Upsilon_{GIS})$ for which $L_k^r \in G$ and produces the result relation.

Definition 6.2 A *query* Q on a schema S is a regular expression over Σ, such that $Q(S) = (u, l_i, v_1)$. The triple (u, l_i, v_1) is a path from u to v_1 labelled with l_i. The element (u, l_i, v_1)

---

[45] $\text{Gr}(R) \cong \{\langle x, y\rangle \mid x\ R\ y\}$

is a solution (satisfies) to $Q(S)$ if and only if there is a *Path-Label* $pl \in S$ such that pl **in** l_i and l_i **in** pl (where "**in**" is sequence inclusion). ■

Recall that a query Q over the *GIS* is decomposed into sub-queries $(q_i \mid i: \mathbb{N}_1)$ over $\bigcup_{i=1}^{n} CS_i$. Suppose, there is an *exact* decomposition of Q into q_i such that every query in q_i is contained in $Q(GIS)$ and $Q(GIS)$ is also contained in $\bigcup_{i,j=1}^{n} q_i(CS_j)$, then,

$$\bigcup_{i,j=1}^{n} q_i(CS_j) \sqsubseteq Q(GIS) \Rightarrow \tag{6.8}$$

$$Q(GIS) \sqsubseteq \bigcup_{i,j=1}^{n} q_i(CS_j) \bullet Q(GIS) \equiv \bigcup_{i,j=1}^{n} q_i(CS_j)$$

where the symbol "$q_i \sqsubseteq Q$" means that q_i is contained in Q.

Corollary 6.2 Suppose $Q(GIS) \equiv \bigcup_{i,j=1}^{n} q_i(CS_j)$, where $Q(GIS) = (u, l_i, v_1)$ and $\bigcup_{i,j=1}^{n} q_i(CS_j) = (u_1, l_k, w)$, then $(u, l_i, v_1) \subseteq (u_1, l_k, w) \wedge (u, l_i, v_1) \supseteq (u_1, l_k, w)$.

Proof. $Q(GIS) \equiv \bigcup_{i,j=1}^{n} q_i(S_j)$ means that (u, l_i, v_1) satisfies $Q(GIS)$, which implies that there is a *Path-Label* pl_i in *GIS* such that pl_i **in** l_i and l_i **in** pl_i. Notice that $pl_i \in GIS$ implies that there exists some $pl_k \in CS_{\text{SET}}$ such that pl_i **in** $pl_k \wedge pl_k$ **in** $pl_i \Rightarrow pl_i \in CS_{\text{SET}}$ since every element in CS_{SET} is represented at most once in the *GIS* (from (6.6)). Also, $pl_i \in CS_{\text{SET}}$ implies that $\bigcup_{i,j=1}^{n} q_i(CS_j)$ satisfies (u, l_i, v_1) such that $\bigcup_{i,j=1}^{n} q_i(CS_j) = (u, l_i, v_1)$ (by the *closed domain assumption*). Similarly, $\bigcup_{i,j=1}^{n} q_i(CS_j) = (u_1, l_k, w)$ means that (u_1, l_k, w) satisfies $\bigcup_{i,j=1}^{n} q_i(CS_j)$, which implies that there is a *Path-Label* pl_k in CS_{SET} such that pl_k **in** l_k and l_k **in** pl_k. Observe that $pl_k \in CS_{\text{SET}}$ implies that there exists some $pl_i \in GIS$ such that pl_i **in** $pl_k \wedge pl_k$ **in** $pl_i \Rightarrow pl_k \in GIS$ (since *GIS* is the range of Υ_{GIS} from (6.7)). Every *Path-Label* $pl_i \in GIS$ is unique

(by Definition 4.8). So, pl_i **in** $pl_k \wedge pl_k$ **in** pl_i and $pl_k \in GIS$ contradicts the uniqueness of pl_i. Therefore, $i = k$ such that $pl_i = pl_k$.

An interesting implication of Corollary 6.2 is that the *GIS* correctly represents all the elements in the set of context schemas $\bigcup_{i=1}^{n} CS_i$. ∎

6.5 Integration Examples

In this section, we consider some simple integration examples to illustrate how the integration algorithm works. We also discuss example queries to show how the query processor decomposes users' queries to the appropriate data sources, and combines the partial results to form the overall query result.

Example 6.1 Consider the requisition forms used by two online bookstores, Student-Books Inc. and Books-Warehouse Ltd. The requisition form for Student-Books Inc. is represented with the following schema S_1, while the requisition form for Books-Warehouse Ltd is represented with schema S_2.

S_1 = SB-Order [Number, Amount, Quantity, Customer, Contact [Phone-Number, Address]]

S_2 = Order-BW [No, Amount, Qty, Customer [No, Name], Contact [Name, Address], Delivery]

Table 6.1: Substituting Context Labels for Source Labels in S_1

Source Schema	**Source Labels (*l*)**	**Context Schema**	**Context Labels (*cl*)**
S_1	SB-Order	CS_1	Order
S_1	SB-Order [Number]	CS_1	Order [ID]
S_1	SB-Order [Amount]	CS_1	Order [Amount]
S_1	SB-Order [Quantity]	CS_1	Order [Quantity]
S_1	SB-Order [Customer]	CS_1	Order [Customer]
S_1	SB-Order [Contact]	CS_1	Order [Contact]
S_1	SB-Order [Contact [Phone-Number]]	CS_1	Order [Contact [Phone]]
S_1	SB-Order [Contact [Address]]	CS_1	Order [Contact [Address]]

CS_1 is the corresponding context schema, for the source schema, S_1.

CS_1 = *Order* [*ID*, *Amount*, *Quantity*, *Customer*, *Contact* [*Phone*, *Address*]]

Table 6.2: Substituting Context Labels for Source Labels in S_2

Source Schema	Source Labels (*l*)	Context Schema	Context Labels (*cl*)
S_2	Order-BW	CS_2	Order
S_2	Order-BW [No]	CS_2	Order [ID]
S_2	Order-BW [Amount]	CS_2	Order [Amount]
S_2	Order-BW [Qty]	CS_2	Order [Quantity]
S_2	Order-BW [Customer]	CS_2	Order [Customer]
S_2	Order-BW [Customer [No]]	CS_2	Order [Customer [ID]]
S_2	Order-BW [Customer [Name]]	CS_2	Order [Customer [Name]]
S_2	Order-BW [Contact]	CS_2	Order [Contact]
S_2	Order-BW [Contact [Name]]	CS_2	Order [Contact [Name]]
S_2	Order-BW [Contact [Address]]	CS_2	Order [Contact [Address]]
S_2	Order-BW [Delivery]	CS_2	Order [Delivery]

CS_2 is the corresponding context schema, for the source schema, S_2.

CS_2 = *Order* [*ID*, *Amount*, *Quantity*, *Customer* [*ID*, *Name*], *Contact* [*Name*, *Address*], *Delivery*]]

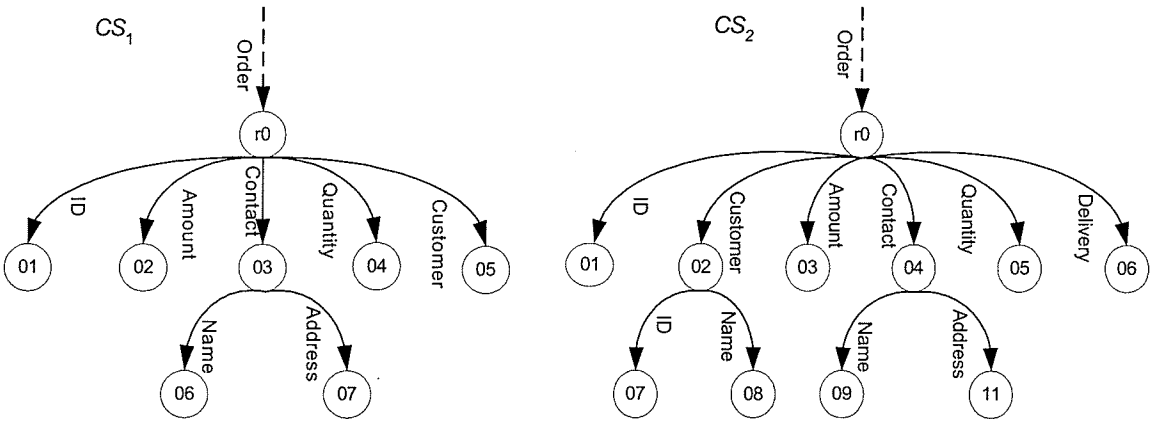

Figure 6.6: Representation of Context Schemas (CS_1, CS_2) in *Del-G*

These two schemas clearly represent the same universe of discourse, even though they are structurally different. As in Example 4.4, phase one of the integration process involves the lookup of terms from the local schemas (S_1 and S_2) in the *CTV* (shown in Table 6.1 and Table 6.2). This is a one-time process that is carried out offline. Thereafter, the software assistant, *Del-G* Template, assists the DBA to generate context schemas CS_1

and CS_2 corresponding to the local schemas S_1 and S_2 respectively. Figure 6.6 shows the context schemas represented using *Del-G*.

In the second phase, the *GIS*-Algorithm uses the two context schemas as input to generate a global integrated schema *GIS* as output. We illustrate how the algorithm integrates CS_1 and CS_2 with the following steps.

Step 1: Initialization and comparison of entry-point labels

The pair of entry-point labels of CS_1 and CS_2 (L_1, L_2) (where $L_1 = CS_1.Order$ and $L_2 = CS_2.Order$) is dequeued from *IQ* and compared. This comparison shows that $L_1 = L_2$, which takes us to Step 2.

Step 2: *Enqueue*, *Dequeue* and *Compare* child elements of L_1 and L_2

Child elements of L_1 and L_2 are also enqueued and compared. The following pairs are compared;

(*ID*, *ID*), (*ID*, *Amount*), (*ID*, *Quantity*), (*ID*, *Customer*), (*ID*, *Contact*), (*ID*, *Delivery*);

(*Amount*, *ID*), (*Amount*, *ID*), (*Amount*, *Quantity*), (*Amount*, *Customer*), (*Amount*, *Contact*), (*Amount*, *Delivery*);

(*Quantity*, *ID*), (*Quantity*, *Amount*), (*Quantity*, *Quantity*), (*Quantity*, *Customer*), (*Quantity*, *Contact*), (*Quantity*, *Delivery*);

(*Customer*, *ID*), (*Customer*, *Amount*), (*Customer*, *Quantity*), (*Customer*, *Customer*), (*Customer*, *Contact*), (*Customer*, *Delivery*);

(*Contact*, *ID*), (*Contact*, *Amount*), (*Contact*, *Quantity*), (*Contact*, *Customer*), (*Contact*, *Customer*), (*Contact*, *Delivery*);

The comparison shows that there exists equivalence assertion since

$(L_1 = L_2 \wedge (Edge\text{-}Child(L_1) \subseteq Edge\text{-}Child(L_2)))$.

Step 3: Merge Elements

3.1 (CS_1.Order, CS_2.Order) is merged into the *GIS* as *Order*

dequeue any pair containing *Order* from *IQ* (none remaining)

3.2 (*Order.ID*, *Order.ID*) is merged into the *GIS* and every other pair that has *Order.ID* is dequeued. Child elements of the pair (*Order.ID*, *Order.ID*) are then enqueued into *IQ*, but *Order.ID* is a terminal element in both CS_1 and CS_2.

3.3 (*Order.Amount*, *Order.Amount*) is merged into the *GIS*, and every other pair that has *Order.Amount* is dequeued. However, *Order.Amount* is a terminal element, and so involves no further processing.

3.4 (*Order.Quantity*, *Order.Quantity*) is merged into the *GIS*, and every other pair that has *Order.Quantity* is dequeued. *Order.Quantity* is also a terminal element, and so involves no further processing.

3.5 (*Order.Customer*, *Order.Customer*) *Order.Customer* in CS_1 is a terminal element, while *Order.Customer* in CS_2 is a non-terminal element. By the elements constraints assumption, we integrate the two elements as different elements in the *GIS*. We merge *Order.Customer* and its descendants in CS_2 into the *GIS*. We append the source of *Order.Customer* in CS_1 (i.e. CS_1.*Order.Customer*) and merge into the *GIS*. Dequeue any other pair containing *Order.Customer* from *IQ*.

3.6 (*Order.Contact*, *Order.Contact*) is merged into the *GIS*, and every other pair that has *Contact* is dequeued. Both *Order.Contact* $\in$ CS_1 and *Order.Customer* $\in$ CS_2 are non-terminal elements, so *Order.Customer* is merged into the *GIS* as a non-terminal element (by the element constraints assumption). Child elements of the pair (*Order.Contact*, *Order.Contact*) are then enqueued in *IQ* and the process is repeated.

3.7 There is no pair that has the two elements as *Order.Delivery* because *Order.Delivery* only occurs in CS_2. It follows that *Order.Delivery* $\notin$ CS_1. Therefore, *Descendants*(*Order.Delivery*) are inherited by *Order* $\in$ *GIS* (by the inheritance property).

Step 4: Check current state of *IQ*

The present state of *IQ* is empty since there is no pair of elements in the queue. Therefore, the *GIS*-Algorithm terminates.

Step 5: Output

GIS = *Order* [*ID*, *Amount*, *Quantity*, *Customer* [*ID*, *Name*], *Contact* [*Name*, *Phone*, *Address* [*Street*, *City*, *Postal-Code*]], *Delivery*]

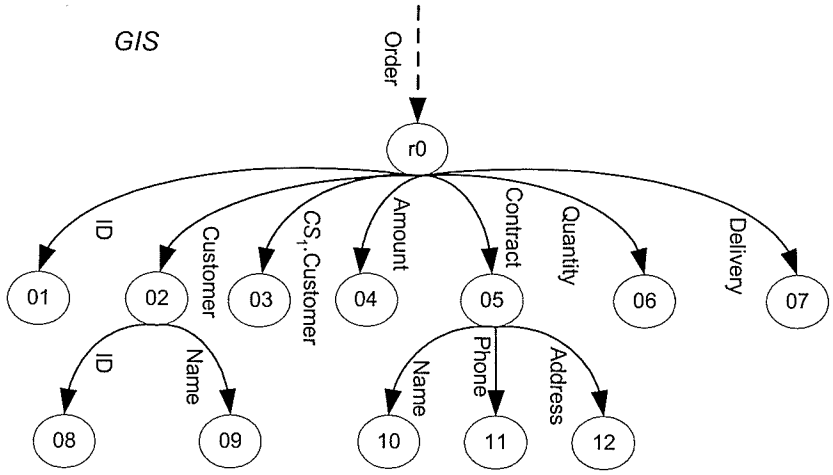

Figure 6.7: Representation of *GIS* in *Del-G*

Figure 6.7 shows the *GIS* represented in *Del-G* format, while Figure 6.8 shows an equivalent XML representation of the output (*GIS*).

```
⟨?xml version="1.0" ?⟩
⟨Schema name = "Order"
        Xmlns= "urn:schemas-microsoft-com:xml-data"
        Xmlns:dt="urn"schemas-microsft-com:datatypes"⟩
⟨ElementType name = "[Order]" sys_type = "Object" ⟩
        ⟨element type = "[Order] ID" sys_type = "Attribute" /⟩
        ⟨element type = "[Order] Amount" sys_type = "Attribute" /⟩
        ⟨element type = "[Order] Quantity" sys_type = "Attribute" /⟩
        ⟨element type = "[Order;Customer]" sys_type = "Object"/⟩
        ⟨element type = "[Order;Customer] ID" sys_type = "Attribute"/⟩
        ⟨element type = "[Order;Customer] Name" sys_type = "Attribute"/⟩
        ⟨element type = "[Order;Contact]" sys_type = "Object"/⟩
        ⟨element type = "[Order;Contact] Name" sys_type = "Attribute"/⟩
        ⟨element type = "[Order;Contact] Phone" sys_type = "Attribute"/⟩
        ⟨element type = "[Order;Contact] Address" sys_type = "Attribute"/⟩
        ⟨element type = "[Order] Delivery" sys_type = "Attribute" /⟩
⟨/element type⟩
⟨/Schema⟩
```

Figure 6.8: XML Representation of a *Del-G* Schema

For a user to access a semistructured data model, such as *Del-G*, the user has to query the system to request the information needed. Queries are posed based on the underlying syntax of a given language. The ability to query semistructured data is supported by

languages that exploit arc variables bound to labels on the arcs rather than nodes in the graph [BDHS96]. For the purpose of this research, it is assumed that all queries are written in Lorel [AQM+97]. The Lorel query language takes advantage of successive path knowledge to reach arbitrary depths in the edge-labelled graph. Let us consider an example.

Example 6.2 Consider a simple query on the *GIS*. Suppose a user "Dr. John Johnson" wants to know the quantity of books he has ordered from both Student-Books Inc., and Books-Warehouse Ltd. to enable him to manage his research grant. Dr. Johnson poses a query over the *GIS* of Figure 6.7 (Q(*GIS*)), which is decomposed into sub-queries q_1 and q_2 over the data sources ($\bigcup_{i,j=1}^{n} q_i(CS_j)$). The query processor processes the query ($Q(GIS)$) as follows:

- Send a source query q_1 to Student-Books Inc. to retrieve the quantity of all the books ordered by "Dr. John Johnson". The query is written as;

  SELECT *Quantity* X
  FROM CS_1.*Order.Quantity* X
  WHERE *Customer* = "Dr. John Johnson"

- Send a source query q_2 to Books-Warehouse Ltd. to retrieve the quantity of all the books ordered by "Dr. John Johnson" written as;

  SELECT *Quantity* X
  FROM CS_2.*Order.Quantity* X
  WHERE *Customer.Name* = "Dr. John Johnson"

- The FROM clause binds the variable X to each node specified in the path [*Order.Quantity*]. Notice the differences in the constraints from the WHERE clauses of q_1 and q_2 because of the differences in the structures of CS_1 and CS_2. The query results in the element (u, l_i, v), which is a *Data-Path*, dp_i such that $u \in$ *Root* and $v \in$ *Terminal-Node*. The computed union of the outputs of q_1 and q_2 provides the answer to the user query.

Example 6.3 Airline-A and Airline-B are two firms in the airline industry that have a service collaboration agreement with respect to baggage handling devices in their areas of operations. Airline-A has an XML database containing information about baggage handling devices. Airline-B has a relational database also containing information about baggage handling devices. With this service agreement, Airline-A and Airline-B will need to share information. However, the conflicts that exits between the underlying data sources impede the exchange of information.

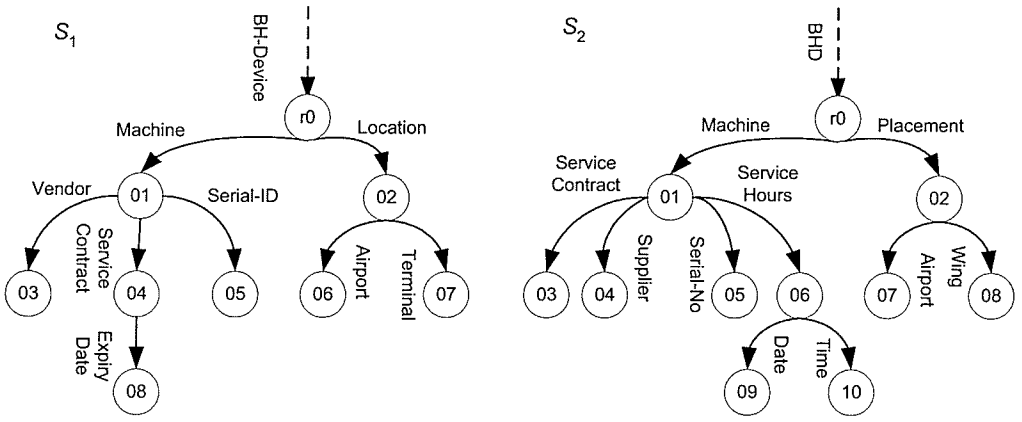

Figure 6.9: Source Schemas for the Airline Industries

In Figure 6.9, S_1 represents the XML database of Airline-A, while S_2 represents the relational database of Airline-B. Observe the naming conflicts that exist between elements in S_1 and S_2. For example BH-Device and BHD, Location and Placement, Vendor and Supplier, Serial-ID and Serial-No, and Terminal and Wing. Also notice that the structures of S_1 and S_2 are different. These conflicts impede the exchange of data.

To resolve the naming conflicts, the DBA of each data source who wants to provide more access to the information in his/her data source uses the *CTV* to annotate terms. The annotation tool here includes a software editor called *Del-G Template* that provides a unified structure to the data sources so that context labels corresponding to terms in each data source are represented as context schemas. Figure 6.10 shows the context schemas CS_1 and CS_2 corresponding to the local schemas S_1 and S_2 respectively. Notice that the naming conflicts that existed in Figure 6.9 no longer exist. However, the structural conflicts still exist.

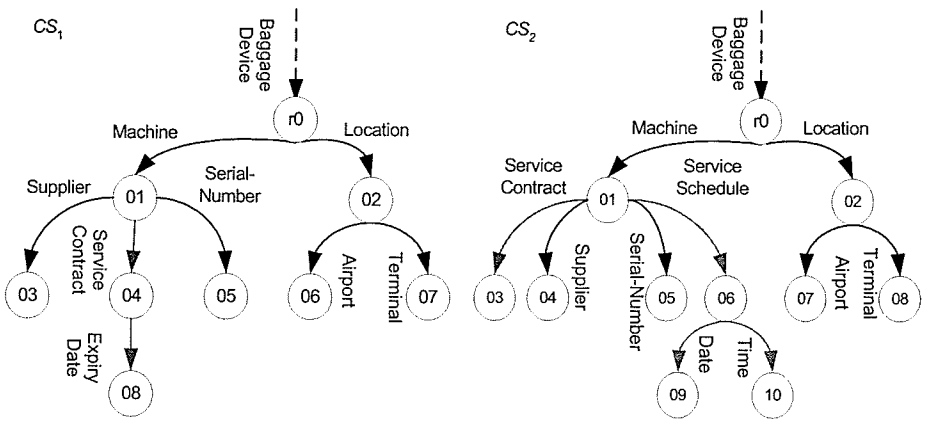

Figure 6.10: Corresponding Context Schemas for the Airline Industries

In Phase 2, the *GIS*-Algorithm uses the context schemas as input to generate the *GIS*. Notice that Service-Contract in CS_1 is a non-terminal element because it has a child (Expiry-Date), while Service-Contract in CS_2 is a terminal element because it has no child. By the elements constraints assumption, we integrate Service-Contract in the two context schemas as different elements. We append the source name (CS_2) to the terminal element in the *GIS* (i.e. CS_2.Service-Contract). Another interesting point to note is the inheritance property. Observe that Service-Schedule in CS_2 does not exist in CS_1. So, Service-Schedule and its descendants are inherited into the *GIS*. The algorithm does not need to check if the descendants of Service-Contract exist in CS_1 because they are guaranteed not to occur by the inheritance property (Theorem 6.1). Service-Contract and its descendants are isolated from the search space and merged (inherited) into the *GIS*. We now trace how the *GIS*-Algorithm integrates the two context schemas in the following steps.

CS_1 = Baggage-Device [Machine [Supplier, Service-Contract [Expiry-Date], Serial-Number], Location [Airport, Terminal]]

CS_2 = Baggage-Device [Machine [Supplier, Service-Contract, Serial-Number, Service-Schedule [Date, Time]], Location [Airport, Terminal]]

Step 1: Initialization and comparison of entry-point labels

The pair of entry-point labels of CS_1 and CS_2 (L_1, L_2) (where L_1 = CS_1.*Baggage-Device* and L_2 = CS_2.*Baggage-Device*) is dequeued from *IQ* and compared. This comparison shows that $L_1 = L_2$, which takes us to Step 2.

Step 2: *Enqueue*, *Dequeue* and *Compare* child elements of L_1 and L_2

Child elements of L_1 and L_2 are also enqueued and compared. The following pairs are compared;

(*Machine*, *Machine*), (*Machine*, *Location*);

(*Location*, *Machine*), (*Location*, *Location*);

The comparison shows that ($L_1 = L_2 \wedge$ (*Edge-Child*(L_1) $\subseteq$ *Edge-Child*(L_2)) $\wedge$ (*Edge-Child*(L_1) $\supseteq$ *Edge-Child*(L_2))).

Step 3: Merge Elements

3.1 (*CS_1.Baggage-Device*, *CS_2.Baggage-Device*) is merged into the *GIS* as *Baggage-Device* (dequeue any pair containing *Baggage-Device* from *IQ* (none remaining))

3.2 (*Machine*, *Machine*) is merged into the *GIS* and every other pair that has *Machine* is dequeued. Child elements of the pair (*Machine*, *Machine*) are then enqueued into *IQ*, and compared.

Step 3.2.0: Compare Child elements of *Machine*

(*Supplier*, *Supplier*); (*Supplier*, *Service-Contract*); (*Supplier*, *Serial-Number*); (*Supplier*, *Service-Schedule*);

(*Service-Contract*, *Supplier*); (*Service-Contract*, *Service-Contract*); (*Service-Contract*, *Serial-Number*); (*Service-Contract*, *Service-Schedule*);

(*Serial-Number*, *Supplier*); (*Serial-Number*, *Service-Contract*); (*Serial-Number*, *Serial-Number*); (*Serial-Number*, *Service-Schedule*);

3.2.1 (*Supplier*, *Supplier*) is merged in the *GIS* as *Supplier*

Dequeue any other pair containing *Supplier* from *IQ*.

3.2.2 (*Service-Contract*, *Service-Contract*) *Service-Contract* in CS_1 is a non-terminal element, while *Service-Contract* in CS_2 is a terminal element. By the elements constraints assumption, we integrate the two elements as different elements in the *GIS*. We merge *Service-Contract* and its descendants in CS_1 into the *GIS*. We append the source of *Service-Contract* in CS_2 (i.e. *CS_2.Service-Contract*) and merge into the *GIS*. Dequeue any other pair containing *Service-Contract* from *IQ*.

3.2.3 (*Serial-Number*, *Serial-Number*) is merged in the *GIS* as *Serial-Number*.

Dequeue any other pair containing *Serial-Number* from *IQ*.

3.2.4 *Service-Schedule* in CS_2 does not exist in CS_1, so *Service-Schedule* and its descendants are inherited (by the inheritance property) into the *GIS*.

3.3 (*Location*, *Location*) is merged into the *GIS* and every other pair that has *Location* is dequeued. Child elements of the pair (*Location*, *Location*) are then enqueued into *IQ*, and compared.

Step 3.3.0: Compare Child elements of *Location*

(*Airport*, *Airport*); (*Airport*, *Terminal*);

(*Terminal*, *Airport*); (*Terminal*, *Terminal*);

3.3.1 (*Airport*, *Airport*) is merged in the *GIS* as *Airport*

Dequeue any other pair containing *Airport* from *IQ*.

3.3.2 (*Terminal*, *Terminal*) is merged in the *GIS* as *Terminal*

Dequeue any other pair containing *Terminal* from *IQ*.

Step 4: Check current state of *IQ*

The present state of *IQ* is empty since there is no pair of elements in the queue. Therefore, the *GIS*-Algorithm terminates.

Step 5: Output

GIS = *Baggage-Device* [*Machine* [*Supplier*, CS_2.*Service-Contract*, *Service-Contract* [*Expiry-Date*], *Serial-Number*, *Service-Schedule* [*Date*, *Time*]], *Location* [*Airport*, *Terminal*]]

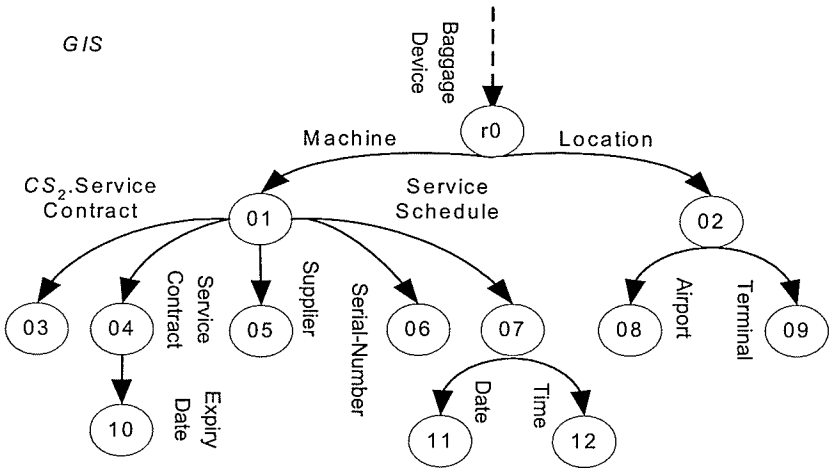

Figure 6.11: *GIS* for the Airline Industries

Figure 6.11 shows the *GIS* represented in *Del-G* format and Figure 6.12 shows an equivalent XML representation of the output (*GIS*).

```
⟨?xml version="1.0" ?⟩
⟨Schema name = "Baggage-Device"
    Xmlns= "urn:schemas-microsoft-com:xml-data"
    Xmlns:dt="urn"schemas-microsft-com:datatypes"⟩
⟨ElementType name = "[Baggage-Device]" sys_type = "Object" ⟩
    ⟨element type = "[Baggage-Device;Machine]" sys_type = "Object" /⟩
    ⟨element type = "[Baggage-Device; Location]" sys_type = "Object" /⟩
    ⟨element type = "[Baggage-Device;Machine] Supplier" sys_type = "Attribute" /⟩
    ⟨element type = "[Baggage-Device;Machine] CS2.Service-Contract" sys_type = "Attribute"/⟩
    ⟨element type = "[Baggage-Device;Machine;Service-Contract]" sys_type = "Object"/⟩
    ⟨element type = "[Baggage-Device;Machine;Service-Contract] Expiry-Date"= "Attribute"/⟩
    ⟨element type = "[Baggage-Device;Machine] Serial-Number" sys_type = "Attribute"/⟩
    ⟨element type = "[Baggage-Device;Machine;Service-Schedule]" sys_type = "Object"/⟩
    ⟨element type = "[Baggage-Device;Machine;Service-Schedule] Date" sys_type = "Attribute"/⟩
    ⟨element type = "[Baggage-Device;Machine;Service-Schedule] Time" sys_type = "Attribute"/⟩
    ⟨element type = "[Baggage-Device; Location] Airport" sys_type = "Attribute" /⟩
    ⟨element type = "[Baggage-Device; Location] Terminal" sys_type = "Attribute" /⟩
⟨/element type⟩
⟨/Schema⟩
```

Figure 6.12: A Corresponding XML Representation of the *GIS* of Figure 6.11

Example 6.4 Consider a simple query (Q(*GIS*)) on the *GIS* of Figure 6.9. Which of the Airports have the machine with serial number "SID-10-11-2004" supplied by Solid-Motors?

The query processor decomposes $Q(GIS)$ into sub-queries q_1 and q_2 over the data sources $\bigcup_{i,j=1}^{n} q_i(S_j)$ where $n = 2$.

- Send a source query q_1 to Airline A written as;

  SELECT *Airport* X
  FROM S_1.*BH-Devices.Location.Airport* X
  WHERE *Machine.Serial-ID* = "SID-10-11-2004" &&
  Machine.Vendor = "Solid-Motors"

- Send a source query q_2 to Airline B written as;

  SELECT *Airport* X
  FROM S_2.*BHD.Placement.Airport* X

WHERE *Machine.Serial-No* = "SID-10-11-2004" &&
Machine.Supplier = "Solid-Motors"

- The FROM clause binds the variable X to each node specified in the path [*BH-Devices.Location.Airport*] $\in CS_1$ and [*BHD.Placement.Airport*] $\in CS_2$. Notice the differences in the path labels and the constraints from the WHERE clauses of q_1 and q_2 because of naming conflicts that existed in the sources schemas. The query results in the element (u, l_i, v), which is a *Data-Path*, dp_i such that $u \in$ *Root* and $v \in$ *Terminal-Node*. The computed union of the outputs of q_1 and q_2 provides the answer to the user query.

The integration mechanism produces a set of pre-computed views, *GIS*, that represents the local schemas, and can be queried by different users across diverse platforms. The query processor relies on such views to derive correct answers. In *Del-G*, a regular expression represents a query that exploits path knowledge to retrieve all pairs of nodes connected by a path. A query Q expressed in terms of the *GIS* is decomposed into sub-queries q_i expressed in terms of the local schemas $\bigcup_{i=1}^{n} S_i$. The solution for Q is the computed union of all the partial solutions of q_i. The user does not need to know that S_1 is an XML database, neither does s(he) need to know that S_2 is a relational database. Our approach insulates the user from the intricacies of the underlying data sources. This is a major contribution of this research.

Chapter 7

Conclusions and Future Work

7.1 Summary of Research

The focus of this thesis has been on data infrastructure for e-commerce, in particular, data management needs for e-commerce applications. In e-commerce applications, it is currently the responsibility of the users to locate products or identify potential business partners. That is, users have to contend with the problem of identifying and interpreting Web data. However, identifying Web data involves determining which data are semantically equivalent and this remains a major problem. This research leverages Web data integration principles to achieve its objectives. These objectives include:

- To identify the data management needs of an e-commerce application.
- To semantically describe objects in local data sources so that multiple views of objects are eliminated.
- To dynamically identify and analyse correspondence assertions to remove the burden of interpreting data from the user.
- To generate an integration result that is mathematically sound.

To achieve the envisioned objectives, we characterize e-commerce data sets to identify the features of e-commerce data likely to influence design decisions of a suitable integration system. We also present a data-centric e-commerce application model that brings to focus the data management needs of an e-commerce application.

Two major problems with existing Web integration systems are that most current systems depend on manual efforts and *ad hoc* integration approaches to generate integration results. Identifying various types of data equivalence assertions manually is problematic and inhibits scalability of the integration system. The use of heuristics to identify assertions is not appealing as heuristics produce mapping results that are mathematically unsatisfying. These mathematically unsatisfying results may be unacceptable in the e-commerce domain where correctness of operations is crucial to attract and retain customers' trust. This thesis leverages a flexible algebraic data model and a common-term vocabulary (*CTV*) with sound integration principles and algorithms to provide a theoretical foundation that facilitates reliable and scalable Web data integration.

To address the issue of using *ad hoc* approaches to integration, we used algebraic signatures to specify components of the integration system. Formally specifying components of an integration system reduces its complexity, and clarifies the overall integration process. In addition, we formally defined the integration process and showed theoretically that the integration mechanism correctly maps elements from the local schemas into the global integrated schema (*GIS*). The use of theoretical methods and formal techniques provides sound mathematical principles to establish systems' correctness and completeness, and hence eliminates redundancies to resolve integration challenges. The integration approach adopted in this thesis precisely specifies the behaviour of the system and provides a framework to prove the correctness of the integration results.

A semantic description of every object in the local data sources is provided to eliminate multiple views of objects and naming conflicts between objects of different data sources. Semantic description of data sources requires the use of a common-term vocabulary (*CTV*) that provides a vocabulary of terms acceptable to all participating data sources in the e-commerce domain. We specify a *CTV* that makes explicit the content of data from the local data sources. The domain of e-commerce is wide, and it would be difficult for any group of domain experts to exhaustively define every term. Therefore, we specify area-centric *CTVs* that allow incremental development of an overall *CTV*. This area-centric approach facilitates the evolution and composition of *CTV*s. A software

assistant aids the capture of objects in the local data sources into a machine-readable format (input to the integration algorithm) that draws its terms from the *CTV*. The input schemas to the integration algorithm are free from naming conflicts, and are represented using the platform independent *Del-G* structure. We showed theoretically that the process of using the software assistant to create a machine-readable format for the local data sources is decidable. We designed a graph-based algorithm that leverages the graph structure of *Del-G* and the semantic description provided by the input process to dynamically identify assertions between objects of different data sources. The algorithm also analyses these assertions to generate integration rules. All the assertions identified are formalized using algebraic signatures.

This thesis represents a shift from the manual efforts and *ad hoc* approaches that characterize most existing integration systems. The combination of formalism with sound integration principles produces a flexible and correctness preserving approach to Web data integration. This approach goes beyond representing and reasoning about mapping to dynamically generating mappings of attribute semantics between objects in different data sources. Furthermore, the approach produces mathematically satisfying results, devoid of user intervention.

We, therefore, conclude that the use of formal techniques and a theoretical framework in an integration system guarantees integration results that are mathematically sound. We also conclude that dynamic identification of correspondence assertions is possible if multiple views of objects are eliminated and components of the integration system are formalized.

7.2 Thesis Contributions

A unique feature of this thesis is that it provides an alternative perspective on data management for Web-centric applications by identifying the data management needs of such applications and combining formalism with sound integration principles and algorithms to achieve integration. We designed a four-phase data-centric e-commerce application model [AE04e] to identify the data management needs of e-commerce applications. This model facilitates a customer-driven business environment. Making the customer a focal point of Internet-based interaction is key to customer relationship

management. An effective online customer relationship contributes to the lifetime value of customers by reducing acquisition costs, increasing sales, extending the customer life cycle, and strengthening customer retention. The model also leverages the *CTV* to provide semantic search based on context names.

This thesis also provides a characterization of e-commerce data sets [AE03a]. Characterizing e-commerce data set facilitates the choice of a representation technique and a suitable integration mechanism. The study of business data presented by Quix *et al.* [QSJ02] lacks specification and fails to mention the specific characteristics of such data. In this thesis, we characterized e-commerce data sets and also provided a formal framework for such data sets as described in [AE03c]. The insight acquired from characterizing and specifying e-commerce data sets motivated the choice of our integration approach.

Another fundamental contribution is providing a proven semantic description of objects from data sources [EA04]. We presented an algebraic data model that reduces the topological structure of Web data sources to regular expressions. The algebraic data model provides extensible data structures (that are not rigid), where algebraic operators and functions manipulate objects. Although, this algebraic data model is simple and flexible to accommodate e-commerce data, it is unable to capture the real world state of objects completely. Therefore, we specified a *CTV* to provide a vocabulary of terms acceptable to all participating data sources. A key characteristic of the *CTV* is the fact that it is area-centric, and can handle evolution and composition of *CTV*s. Central to the input process is a filter mechanism that recognizes a context schema so that all the participating schemas are guaranteed to be without naming conflicts and multiple views of objects. We showed that the process of recognizing a context schema is decidable. With this approach, the input process is isolated from the integration process to provide autonomy of data sources. There is an overhead in semantically describing objects in local data sources. However, the overhead is justified when the benefits of dynamically identifying and analyzing assertions are considered.

This thesis also identified and specified more correspondence assertions [AE04c, AE04d] than existing systems, such as [Che00, SP94]. We semantically described data sources and adopted a fine-grained comparison of objects to identify more assertions.

Until now, manual efforts and *ad hoc* approaches characterized the process of identifying and generating correspondence assertions. This research also contributes by providing dynamic identification and analysis of correspondence assertions [EA04].

An essential component of the integration process is an efficient graph-based schema integration algorithm that insulates the user from the intricacies of the underlying data sources. The algorithm optimizes the integration process by dynamically analyzing correspondence assertions to generate integration rules. Dynamically identifying and analysing assertions enhances scalability of the integration system. Thus, new data sources may be added (or existing ones removed) without undermining existing correspondence assertions. Formally specifying correspondence assertions and dynamic generation of integration rules guarantees mapping results that are mathematically sound.

We described a high-level integration architecture, WISE, that can be used to integrate e-commerce data sets [AE04a]. WISE, like Unity [Law01], isolates the input process from the integration process. However, unlike Unity, WISE adopts a theoretical approach to specify its components and can show that the integration results are complete, correct, minimal, and understandable ([AE03b]). Thus, this thesis contributes by providing an analysis, and proof of completeness and correctness of the integration process. Although, the implementation of WISE is outside the scope of this thesis, such implementation should be straightforward since components of the integration system have been formally specified. Providing a detailed description of a high level architecture for a schema integration system allows better understanding of the systems' components. In addition, the insights provided by the specification of WISE and a verbose description of its components should aid system developers to make sound implementation choices, and implement and deploy the abstract integration architecture with minimum difficulty.

Finally, a major contribution of this thesis is the use of algebraic signatures to specify components of the integration system [AE04d]. Algebraic signatures provide a formal foundation for establishing the correctness of the integration model. A formal foundation offers the framework to prove that the specification of a system is realizable and to prove properties of the system without necessarily executing such properties to determine its behaviour [Win90]. Overall, this research presents a formal framework for reasoning about the integration system's functionality, behaviour, and constraints

imposed by its underlying structure. Formally specifying components of an integration system for e-commerce transactions reduces the system's complexity (because it removes ambiguities from the informal definitions), eliminates redundancies, and elucidates the overall integration process, thereby, providing a framework to automate and verify the correctness of the integration process. Thus, this thesis provides a systematic and formal approach to Web data integration.

7.3 Directions for Future Work

The work presented in this thesis brings together three orthogonal areas of study, namely: e-commerce, Web data integration, and formal specification. No single collection of ideas can adequately address all the issues in these areas. Much work remains to be done to enhance the results presented. The following specific areas along these three orthogonal axes require further study/extension.

The four-phase data-centric e-commerce application model provided in this thesis requires extension in some respects. There is a need to extend the model to include other basic infrastructure, such as communication protocols, message passing, language requirements, payment protocols, security protocols, etc. Also, the post-sales phase of the model needs further examination. For example, most organizations use Web-based communities (WBC) to provide post-sales services to their customers. Unfortunately, most of the data generated in such WBCs are largely inaccessible because of information overload and other data management problems inherent in the community. To fully utilize the potential provided by the post-sales phase of the data-centric model requires a mechanism that eliminates or tolerates redundancies from the data generated in the WBCs. Thus, data in the WBCs needs to be presented in a manner that is easily accessible by the organizations without much pre-processing.

One problem with the autonomy of data sources that needs to be examined further is the quality of data from the data sources. The Web provides an information avalanche, but does not determine if there are errors in the content of information it provides. To achieve and sustain the overall goal of integration, it is imperative that the data be reliable, consistent and regulated. There has to be a mechanism to check if data

exchanged in the integration system is fit for use. Fitness for use implies that within any operational context, the data that is being used meets or exceeds user expectations. Essentially, the data should be free from defects, and data values should be of high quality with respect to correctness, consistency, timeliness and completeness. Guaranteeing data quality is crucial to eliciting customers trust.

While *Del-G*'s simplicity reduces the complexity of the integration problem, there is a need to relax some of its assumptions (see Chapter 4.2.2) to accommodate more complex structures. For example, objects could be allowed to have more than one parent, and *Del-G* could be allowed to be cyclic. More complex structures could be used to represent methods and activities. *Del-G*, in its present state, can only represent data. The integration of large systems requires sharing of information stored in databases beyond sharing of pure data. Such information may include business processes (methods) and business activities that are associated with data. For example, a generic search method can be defined to search catalogs of different merchants instead of each merchant having its own search method. Therefore, the main challenge will be to provide a mechanism for defining, storing, and sharing business processes and activities across different information systems. Business process sharing will be essential to the interoperability of e-commerce systems in the future as it facilitates incorporation of mutually incompatible technologies, and can help with performance by distributing workloads across multiple servers. A system that will facilitate the sharing of business processes using standard technologies to overcome compatibility issues is highly desirable.

Another possible area of extension will be to distribute the *GIS* among different nodes (servers). Under the present arrangement where the *GIS* is centralized, the failure of the *GIS* server will cause the entire system to fail. Distribution also has the potential to eliminate performance bottleneck arising from answering distributed queries with a centralized *GIS*. However, we have to contend with the problems of evolution and consistency maintenance which are common with distributed systems.

A final extension would be to refine the algebraic specifications into lower level specifications that would eventually be executable. Regardless of the fact that specifications are not normally executable, refining the specifications into executable forms improves programmers' productivity by reducing the amount of coding that they

need to do [Ehi97]. In addition, refining the specifications into executable forms facilitates the development of automated tools for verifying the correctness of the specifications. Although the specifications provided in this thesis are backed with theoretical principles that guarantee their correctness, there is still a need for a testbed environment to permit implementations of the specifications. Implementing the specifications will empirically determine their practicality, correctness and completeness.

Appendix A

List of Symbols Used in the Thesis

This appendix provides a glossary of the notations used in this thesis. This glossary is based on simple set notations and predicate logic as described in [AP98, Sch94]. The symbols used in this thesis are classified into four groups, namely numbers and strings, logic, set theory, and relations and functions.

A1. Numbers and Strings

Symbol	Name	Example	Meaning
$\mathbb{Z}$	Set of integers	{..., -1, 0, 1, 2, ...}	The set of integers (including positive, zero, and negative integers)
$\mathbb{N}$	Set of natural numbers	{0, 1, 2, ...}	The set of non-negative numbers $\{n : \mathbb{Z} \mid n \geq 0\}$
$\mathbb{N}_1$	Set of positive natural numbers	{1, 2, 3, ...}	The Set of positive natural numbers excluding zero, $\mathbb{N} \setminus \{0\}$
Del-G	Directed edge-labelled graph		The algebraic data model designed in this thesis
l	Labels		A string of characters that describes objects in *Del-G*
λ	Entry-point label		The entry-point label of a *Del-G*
ε	Epsilon		The null string that represents an unknown element in a *Del-G*

A2. Logic

The following are logic definitions. Let a, p, q, r, x, and y be expression terms; A and B be sets, P and Q be predicates, and Γ and α be well-formed formulas (wffs).

Symbol	Name	Example	Meaning
true, *false*	Boolean logical constants	Locate = *true*	Logical constants that are either true or false
$\neg$	not	$\neg P$	Negation
$\wedge$	and	$P \wedge Q$	Conjunction
$\vee$	or	$P \vee Q$	Disjunction
$\Rightarrow$	Implication	$P \Rightarrow Q$	if P then Q
$\Leftrightarrow$	Equivalence (iff)	$P \Leftrightarrow Q$	P and Q are logically equivalent
$\forall$	Universal quantifier (for all)	$\forall x : A \bullet P$	For all x of type A such that the predicate P holds (x is said to be universally quantified over a given range A).
$\exists$	Existential quantifier (there exists)	$\exists x : A \bullet P$	There exists at least one x of the type A such that P holds.
$\exists_1$	Existential unique quantifier	$\exists_1 x : A \bullet P$	There exists exactly one x of type A such that P holds.
$=$	Equality	$x = y$	x and y are equal.
$\neq$	Inequality	$x \neq y$	x and y are not equal.
$\widehat{=}$	By definition, equal to	$A \mathrel{\widehat{=}} \{p, q, r\}$	A is by definition equal to $\{p, q, r\}$.
$\models$	Tautological implication	$\Gamma \models \alpha$	Γ tautologically implies α iff every truth assignment m that satisfies Γ also satisfies α (where Γ and α are wffs).
$\vdash$	Inference	$\Gamma \vdash \alpha$	We infer α if Γ is true.

3. Set Theory

Let A and B be sets, and x a term.

Symbol	Name	Example	Meaning
$\{x, y, \ldots\}$	Set enumeration		Set construction by enumeration
$\in$	Set membership	$x \in A$	x is a member of A
$\notin$	Negation of membership	$x \notin A$	x is not a member of A or $\neg (x \in A)$
$\varnothing$	Empty set or null set	$\{\ \}$	A set without elements
#	Cardinality	$\#A$	The number of distinct members of a finite set
$\subseteq$	Set inclusion	$A \subseteq B$	A is a subset of B $(\forall x : A \bullet x \in B)$
$\subset$	Proper set inclusion	$A \subset B$	A is a proper subset of B $(\forall x : A \bullet x \in B \land A \neq B)$
$x : A$	Set declaration		By definition $x \in A$
$\cup$	Union	$A \cup B$	Set union $\{x : A \mid x \in A \lor x \in B\}$
$\cap$	Intersection	$A \cap B$	Set intersection $\{x : A \mid x \in A \land x \in B\}$
$\setminus$	Relative complement	$A \setminus B$	Set difference $\{x : A \mid x \in A \land x \notin B\}$
Σ	Sum	ΣA_i	The union of a set of disjoint sets of A_i, where $(1 \leq i \leq n)$
$\mathbb{P}$	Powerset	$\mathbb{P}A$	The set of all subsets of A
$\mathbb{P}_1$	Nonempty powerset	$\mathbb{P}_1 A$	The set of all nonempty subsets of A
$\cup$	Union of a set of sets	$\cup$ (Sets)	Union of a set of sets
$\cap$	Intersection of a set of sets	$\cap$ (Sets)	Intersection of a set of sets

A4. Relations and Functions

Let A and B be sets, R be a relation and F be a function.

Symbol	Name	Example	Meaning
dom	Domain	dom R	The domain of a relation R
ran	Range	ran R	The range of a relation R
R	Relation	$R : A \leftrightarrow B$	A relation R with domain A and range B
F	Function	$F: A \rightarrow B$	A function F from A to B
$x \rightarrow y$	Functional association	$x \rightarrow y$	Member to member association in a function
$\circ$	Composition	F1 $\circ$ F2	Functional composition
$\prec_{pre}$	Precede	R1 $\prec_{pre}$ R2	*R1* precedes *R2* (the execution of *R1* precedes *R2*)
$\lhd$	Domain restriction	A $\lhd$ F	The operator $\lhd$ restricts the domain of a function
$\rhd$	Range restriction n	F $\rhd$ B	The operator $\rhd$ restricts the range of a function

References

[Abi97] Abiteboul, S., "Querying Semistructured Data", *Proceedings of 6th International Conference on Database Theory* (ICDT '97), Delphi, Greece, January 8-10, 1997, pp. 1-18.

[ABS00] Abiteboul, S., Buneman, P., and Suciu, D., *Data on the Web: From Relations to Semistructured Data and XML*, Morgan Kaufmann Publishers, San Francisco, CA, 2000.

[ACHK93] Arens, Y., Chee, C. Y., Hsu, C., and Knoblock, C. A., "Retrieving and Integrating Data from Multiple Information Sources", *International Journal of Cooperative Information Systems*, Vol. 2, No. 2, 1993, pp. 127-158.

[ACM+99] Abiteboul, S., Cleut, S., Mignet, L., Amann, B., Milo, T., and Eyal, A., "Active Views for Electronic Commerce", *Proceedings of 25th VLDB Conference*, Edinburgh, Scotland, 1999, pp. 138-149.

[Adi05] Adiele, C., "A Characterization of Web Data Sources to Facilitate E-Commerce Transactions", *Proceedings of the IEEE International Conference on Information Technology: Coding and Computing (ITCC 2005)*, Las Vegas, NV, April 2005 (Accepted to Appear).

[AE03a] Adiele, C., and Ehikioya, S. A., "A Formal Characterization of Web Data for Electronic Commerce Data Integration", Technical Report No. TR-03-06, University of Manitoba, 2003.

[AE03b] Adiele, C., and Ehikioya, S. A., "Evolving a Wise Integration System for E-Commerce Transactions", Technical Report No. TR-03-07, University of Manitoba, 2003.

[AE03c] Adiele, C., and Ehikioya, S. A., "A Characterization of E-Commerce Data on the Web for Data Integration", Technical Report No. TR-03-10, University of Manitoba, 2003.

[AE04a] Adiele, C., and Ehikioya, S. A., "A Flexible Integration Model for Web-Based E-Commerce Transactions", *Proceedings of the 19th International Conference on Computers and Their Applications* (CATA-04), Seattle, WA, March 2004, pp. 87-91.

[AE04b] Adiele, C., and Ehikioya, S. A., "Managing Data in a B2B-Centric Web-Based Community", *Proceedings of the International Conference on Web Based Communities (WBC 2004)*, Lisbon, Portugal, March 2004, pp. 279-286.

[AE04c] Adiele, C., and Ehikioya, S. A., "Dynamic Identification of Correspondence Assertions for Electronic Commerce Data Integration", *Proceedings of the IEEE International Conference on Information Technology (ITCC 2004)*, Las Vegas, NV, April 2004, pp. 223-227.

[AE04d] Adiele, C., and Ehikioya, S. A., "A Formal Analysis of Correspondence Assertions for E-Commerce Data Integration", *Proceedings of the 2nd International Conference on Software Engineering Research, Management and Applications (SERA2004)*, Los Angeles, CA, May 2004, pp. 131-138.

[AE04e] Adiele, C., and Ehikioya, S. A., "Managing Data in a Data-Centric E-Commerce Application Model", *Proceedings of the Seventh International Conference on Electronic Commerce Research (ICECR7)*, Dallas, TX, June 2004, pp. 421-430.

[AE04f] Adiele, C., and Ehikioya, S. A., "Integrating Web-Based Medical Data Sets for Medical Research", *Proceedings of International Conference on e-Society (e-Society 2004)*, Avila, Spain, July 16-19, 2004.

[AE04g] Adiele, C., and Ehikioya, S. A., "Towards a Formal Data Management Strategy for a Web-Based Community", *International Journal of Web Based Communities*, Vol. 1, No. 3, 2004.

[AE04h] Adiele, C., and Ehikioya, S. A., "Algebraic Signatures to Semantically Describe E-Commerce Data Sets for Integration", *Proceedings of the International Conference on Knowledge Sharing and Collaborative Engineering (KSCE 2004)*, St. Thomas, Virgin Islands, USA, November 2004 (Accepted to Appear).

[AE05] Adiele, C., and Ehikioya, S. A., "Algebraic Signatures for Scalable Web Data Integration for Electronic Commerce Transactions", *Journal of Electronic Commerce Research (JECR)*, (Accepted).

[AP98] Alagar, V. S., and Periyasamy, K., *Specification of Software Systems*, Springer-Verlag Inc., New York, 1998.

[APHS02] Aberer, K., Punceva, M., Hauswirth, M., and Schmidt, R., "Improving Data Access in P2P Systems", *IEEE Internet Computing*, Vol. 6, Issue 1, 2002, pp. 58-67.

[AQM+97] Abiteboul, S., Quass, D., McHugh, J., Widom, J., Wiener, J. L., "The Lorel Query Language for Semistructured Data", *International Journal on Digital Libraries*, Vol. 1, No. 1, 1997, pp. 68-88.

[ASD+91] Ahmed, R., Smedt, P. D., Du, W., Kent, W., Ketabchi, M. A., Litwin, W. A., Raffi, A., and Shan, M., "The Pegasus Heterogeneous Multidatabase System", *IEEE Compute*, Vol. 24, No. 12, 1991, pp. 19-27.

[ASX01] Agrawal, R., Somani, A., and Xu, Y., "Storage and Querying of E-Commerce Data", *The VLDB Journal*, 2001, pp. 149-158.

[AT95] Atzeni, P., and Torlone, R., "Schema Translation between Heterogeneous Data Models in a Lattice Framework", *Proceedings of the IFIP Working Conference on Data Semantics*, Atlanta, Georgia, 1995, pp. 345-364.

[Bak98] Bakos, Y., "The Emerging Role of Electronic Marketplaces on the Internet" *Communications of the ACM*, Vol. 41, No. 8, August 1998, pp. 35-42.

[BC00] Bonifati, A., and Ceri, S., "Comparative Analysis of Five XML Query Languages", *ACM SIGMOD Record*, Vol. 29, No. 1, 2000, pp. 68-79.

[BCBV01] Bergamaschi, S., Castano, S., Beneventano, D., Vincini, M., "Semantic Integration of Heterogeneous Information Sources", *Data & Knowledge Engineering* (*Special Issue on Intelligent Information Integration*), Vol. 36, No. 1, Elsevier Science B.V. 2001, pp. 215-249.

[BCE+93] Bukhres, O., Chen, J., Elmagarmid, A., Liu, X., and Mullen, J., "InterBase: A Multidatabase Prototype System", *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Washington, D.C., 1993, pp. 534-539.

[BCV99] Bergamaschi, S., Castano, S., Vincini, M., "A Semantic Approach to Information Integration: The MOMIS Project", *SIGMOD Record,* (Special Issue on Semantic Interoperability in General Information), Vol. 28, No. 1, March 1999, pp. 54-59.

[BDFS97] Buneman, P., Davidson, S., Fernandez, M., and Suciu, D., "Adding Structure to Unstructured Data", *Proceedings of the International Conference on Database Theory*, Delphi, Greece, 1997, pp. 336-350.

[BDHS96] Buneman, P., Davidson, S., Hillebrand, G., and Suciu, D., "A Query Language and Optimization Techniques for Unstructured Data", *Proceedings of the ACM SIGMOD Conference on Management of Data*, Montreal, Canada 1996, pp. 505-517.

[BDK+02] Bhide, M., Deoasee, P., Katkar, A., Panchbudhe, A., and Ramamritham, K., "Adaptive Push-Pull: Disseminating Dynamic Web Data, *IEEE Transactions on Computers*, Vol. 51, No. 6 pp. 652–668.

[BF01] Bertino, E., and Ferrari, E., "XML and Data Integration", *IEEE Internet Computing*, Vol. 5, No. 6, 2001, pp. 75-76.

[BG77] Burstall, R., and Goguen, J., "Putting Theories to Make Specifications", *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, Cambridge, MA, 1977, pp. 1045-1058.

[BGRV99] Bright, L., Gruser, J., Raschid, L., and Vidal, M., "A Wrapper Generation Toolkit to Specify and Construct Wrappers for Web Accessible Data Sources", *International Journal of Computer Systems Science and Engineering*, Vol. 14, No. 2, 1999, pp. 83-97.

[BKLW99] Busse, S., Kutsche, R., Leser, U., and Weber, H., "Federated Information Systems: Concepts, Terminology and Architectures", *Forschungsberichte des Fachbereichs Informatik 99-9*, 1999. Available at: http://citeseer.nj.nec.com/busse99federated.html

[BL02] Barker, K., and Lawrence, R., "Flexible Semantic B2B Integration using XML Specifications", *The 6th World Multi-Conference on Systemics, Cybernetics, and, Informatics*, Orlando, Florida, July 2002.

[BLN86] Batini, C., Lenzerini, M. and Navathe, S., "A Comparative Analysis of Methodologies for Database Schema Integration", *ACM Computing Surveys*, Vol. 18, No. 4, December 1986, pp. 323-364.

[BM02] Berlin, J., and Motro, A., "Database Schema Matching using Machine Learning with Feature Selection" *Proceedings of the 14th International Conference on Advanced Information Systems Engineering (CAiSE02)*, Toronto, Canada, 2002.

[Bor99] Bornhövd, C., "Semantic Metadata for the Integration of Web-Based Data for Electronic Commerce", *Proceedings of the International Workshop on E-Commerce and Web-Based Information Systems*, Santa Clara, CA, 1999, pp. 137-145.

[Bra96] Brackett, M. H., *The Data Warehouse Challenge*: *Taming Data Chaos*, John Willy and Sons, Inc., 1996.

[BS95] Brodie, M. L., and Stonebraker, M., *Migrating Legacy Systems: Gateways, Interfaces, and the Incremental Approach*, Morgan Kaufmann, 1995.

[Bun97] Buneman, P., "Semistructured Data" (*Tutorial in*) *Proceedings of the 16th ACM Symposium on Principles of Database Systems*, Tucson, Arizona, May 1997, pp. 117-121.

[CBB+02] Cattell, R. G. G., Barry, D. K., Berler, M., Eastman, J., Jordan, D., Russell, C., Schadow, O., Stanienda, T., and Velez, F., *The Object Data Standard: ODMG 3.0*, Morgan Kaufmann Publishers, 2002.

[CCG+01] Calvanese, D., Castano, S., Guerra, F., Lembo, D., Melchiori, M., Terracina, G., Ursino, D., and Vincini, M., "Towards a Comprehensive Methodological Framework for Semantic Integration of Heterogeneous Data Sources", *Proceedings of the 8th International Workshop on Knowledge Representation meets Databases (KRDB 2001)*, Rome, Italy, September, 2001.

[CDL01a] Calvanese, D., De Giacomo, G. and Lenzerini, M., "A Framework for Ontology Integration", *Proceedings of the International Semantic Web Working Symposium (SWWS 2001)*, Stanford University, California, July 30 - August 1, 2001, pp. 303-316.

[CDL01b] Calvanese, D., De Giacomo, G. and Lenzerini, M., "Ontology Integration and Integration of Ontologies", *Proceedings of the Descriptive Logic Workshop (DL 2001)*, Stanford University, California, July 30 - August 1, 2001, pp. 10-19.

[CDL+98] Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R., "Description Logic Framework for Information Integration", *Proceedings of the 6th International Conference on the Principles of Knowledge Representation and Reasoning* (KR-98), Trento, Italy, 1998, pp. 2-13.

[CG00] Corcho, O., and Gómez-Pérez, A., "Evaluating Knowledge Representation and Reasoning Capabilities of Ontology Specification Languages", *Proceedings of the ECAI Workshop on Applications of Ontologies and Problem Solving Methods*, Berlin, Germany, 2000.

[CGH+94] Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J., and Widom, J., "The TSIMMIS Project: Integration of Heterogeneous Information Sources", *Proceedings of IPSJ Conference*, Japan, 1994, pp. 7-18.

[Cha01] Chau, P. Y. K., Inhibitors to EDI Adoption in Small Businesses: An Empirical Investigation, *Journal of Electronic Commerce Research*, Vol. 2, No. 2, 2001, pp. 78-88.

[Che00] Chen, Y., "Integrating Heterogeneous OO Schemas", *Journal of Information and Engineering*, Vol. 16, 2000, pp. 555-591.

[CHS91] Collect, C., Hunhs, M., and Shen, W., "Resource Integration Using a Large Knowledge Base in Carnot", *IEEE Computer*, Vol. 24, No. 12, 1991, pp. 55-62.

[CJO02] Cui, Z., Jones, D., and O'Brien, P., "Semantic B2B Integration: Issues in Ontology-Based Approaches", *ACM SIGMOD Record*, Vol. 31, Issue 1, March 2002, pp. 43-48.

[CKMS97] Chen, L. M., Kosky, A., Markowitz, V., and Szeto, E., "Constructing and Maintaining Scientific Database Views in the Framework of Object-Protocol Model", *Proceedings of 9th International Conference on Scientific and Statistical Database Management*, Washington DC, August 1997, pp. 237-248.

[CLRS01] Cormen, T. H., Leiserson, C. E., Rivest, R. L and Stein, C., *Introduction to Algorithms*, Second Edition, The MIT Press, 2001.

[Coh00] Cohen, W., "Data Integration using Similarity Joins and Word-Based Information Representation Language", *ACM Transactions on Information Systems,* Vol. 18, No. 3, July 2000, pp. 288-321.

[Coh99] Cohen, W., "Some Practical Observations on the Integration of Web Information", *Proceedings of the ACM International Workshop on the Web and Databases (WebDB'99),* Philadelphia, USA, June 1999, pp. 55-60.

[CS01] Carbonne, G. and Stoddard, D., *Open Source Enterprise Solutions*, John Willey and Sons, Inc. 2001.

[DCB+01] Davidson, S., Cabtree, J., Brunk, B., Schug, J., Tannen, V., Overton, C., and Stoekert, C., "K2/Kleisli and Gus: Experiments in Integrated Access to Genomic Data Sources", *IBM Systems Journal*, Vol. 40, No. 2, 2001, pp. 512-531.

[DDH00] Doan, A., Domingos, P., and Halevy, A., "Learning Sources Descriptions for Data Integration", *Proceedings of International Workshop on the Web and Databases (WebDB)*, Dallas, TX, 2000, pp. 81-92.

[DDH01] Doan, A., Domingos, P., and Halevy, A., "Reconciling Schemas of Disparate Data Sources: A Machine-Learning Approach", *Proceedings of ACM SIGMOD Conference on Management of Data*, Santa Barbara, California, 2001, pp. 509-520.

[DMDH02] Doan, A., Madhavan, J., Domingos, P., and Halevy, A., "Learning to Map between Ontologies on the Semantic Web", *Proceedings of the 11th*

International World Wide Web Conference (WWW2002), Honolulu, Hawaii, 2002.

[DP97] Domingos, P., and Pazzani, M., "On the Optimality of the Simple Bayesian Classifier Under Zero-One Loss", *Machine Learning,* Vol. 29, 1997, pp. 109-130.

[EA03] Ehikioya, S. A. and Adiele, C., "Towards an Automatic Integration of Web Data Sources for Electronic Commerce Transactions", Technical Report No. TR-03-05, University of Manitoba, 2003.

[EA04] Sylvanus A. Ehikioya and Chima Adiele, "Algebraic Signatures to Analyze Correspondence Assertions for Web Data Integration", *International Journal of Computer and Information Science,* Vol. 5, No. 2, June 2004.

[Ehi97] Ehikioya, S. A., *Specification of Transaction Systems Protocols*, Ph.D. Thesis, Department of Computer Science, The University of Manitoba, Winnipeg, Manitoba, Canada, September 1997.

[Eik99] Eikvil L., "Information Extraction from World Wide Web – A Survey", *Technical Report No.* 945, Norwegian Computing Center, July 1999.

[EM01] Eyal, A. and Milo, T., "Integrating and Customizing Heterogeneous E-Commerce Applications", *VLDB Journal*, Vol. 10, No. 1, 2001, pp. 16-38.

[Euz01] Euzenat, J., "Towards a Principled Approach to Semantic Interoperability", *Proceedings of IJCAI-01 Workshop on Ontologies and Information Sharing*, Seattle, USA, August 2001, pp. 19-25.

[Evi03] Evincible Software "E-Sign Transactions Platform: The Enterprise Architecture to Implement Electronic Transactions Requiring Electronic Signatures", *White Paper*, October 22, 2003. Available at: http://www.evincible.com/resources/eSign%20Transaction%20Platform.pdf

[FCF93] Fox, M. S., Chionglo, J. F., and Fadel, F. G., "A Common Sense Model of the Enterprise", *Proceedings of the 2nd Industrial Engineering Research Conference*, Nacross GA, USA, 1993.

[FDO+01] Fensel, D., Ding, Y., Omelayenko, B., Schulten, E., Botquin, G., Brown, M., and Flett, A., "Product Data Integration in B2B E-commerce", *IEEE Intelligent Systems (Special Issue on Intelligent E-Business)* Vol. 16, No. 4, July/August 2001, pp. 54-59.

[Fel98] Fellbaum, C., *WordNet: An Electronic Lexical Database*, MIT Press, 1998.

[Fen01] Fensel, D., *Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce*, Springer-Verlag, 2001.

[FG97] Fox, M. S., and Gruninger, M., "On Ontology and Enterprise Modeling", *Proceedings of the International Conference on Enterprise Integration Modelling Technology 97*, Springer-Verlag, 1997.

[FLM98] Florescu, D., Levy, A. and Mendelzon, A., "Database Techniques for the World Wide Web: A Survey", *ACM SIGMOD Record*, Vol. 27, No. 3, Sep. 1998, pp. 59-74.

[FWM97] Fiebig, T. Weiss, J., and Moerkotte, G., "RAW: A Relational Algebra for the Web", *ACM SIGMOD Workshop on Management of Semistructured Data*, Tuscon, Arizona, 1997.

[GBMS99] Goh, C., Bressan, S., Madnick, S., Siegel, M., "Context Interchange: New Features and Formalisms for the Intelligent Integration of Information", *ACM Transactions on Information Systems,* Vol. 17, No. 3, July 1999, pp. 270-293.

[GF95] Grüninger, M. and Fox, M., "Methodology for the Design and Evaluation of Ontologies", *IJCAI'95 Workshop on basic Ontological issues in Knowledge Sharing,* Montreal, Canada, April 1995.

[GKD97] Genesereth, M., Keller, A. and Duschka, O., "Infomaster: An Information Integration System", *ACM SIGMOD Record*, Vol. 26, No. 2, May 1997, pp. 539-542.

[GMW99] Goldman, R., McHugh, J., and Widom, J., "From Semistructured Data to XML: Migrating the Lore Data Model and Query Language", *Workshop on the Web and Databases (WebDB '99)*, Philadelphia, Pennsylvania, June 1999, pp. 25-30.

[GPQ+97] Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajararnan, A., Sagiv, Y., Ullman, J., Vassalos, V., and Widom, J., "The TSIMMIS Approach to Mediation: Data Models and Languages", *Journal of Intelligent Information Systems,* Vol. 8, No. 2, 1997, pp. 117-132.

[GR93] Gray, J., and Reuter, A., *Transaction Processing: Concepts and Techniques*, Data Management Systems, Morgan Kaufmann Publishers, Inc., San Mateo (CA), USA, 1993.

[Gru93] Gruber, T., "A Translation Approach to Portable Ontology Specifications", *Knowledge Acquisition*, Vol. 5, No. 2, 1993, pp. 199-220.

[GZC89] Güting, R. H., Zicari, R., and Choy, D. M., "An Algebra for Structured Office Documents," *ACM Transactions on Information Systems* (TOIS) Vol. 7, No. 2, 1989, pp. 123-157.

[Hal02] Halé, J., "From EDI to Web Services – The Evolution of E-Commerce", White Paper, Butler Direct Limited, June 2002.

[Har96] Harry, A., *Formal Methods Fact File: VDM and Z*, John Wiley and Sons, England, 1996.

[HF99] Harmelen, F., and Fensel, D., "Practical Knowledge Representation for the Web", *Proceedings of the IJCAI'99 Workshop on Intelligent Information Integration*, Stockholm, Sweden, 1999.

[HH00a] Heflin, J. and Hendler, J., "Semantic Interoperability on the Web", *Proceedings of Extreme Markup Languages 2000,* Graphic Communications Associations, Montreal, Canada, August 2000, pp. 111-120.

[HH00b] Heflin, J. and Hendler, J., "Dynamic Ontologies on the Web", *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000),* AAAI/MIT Press, Menlo Park, CA, 2000, pp. 443-449.

[HKR+00] Haas, L., Kodali, P., Rice, J., Schwarz, P., and Swope, W., "Integrating Life Sciences Data – With a Little Garlic", *Proceedings of the IEEE International Symposium on Bio-Informatics and Biomedical Engineering*, Washington DC, November 2000, pp. 5-12.

[HKWY97] Haas, L. M., Kossman, D., Wimmers, E. L., and Yang, J., "Optimizing Queries Across Diverse Data Sources", *23rd Conference on Very Large Database Systems*, Athen, Greece, 1997, pp. 276-285.

[HMN+99] Haas, L. M., Miller, R. J., Niswonger, B., Roth, M. T., Schwarz, P. M., Wimmers, E. L., "Transforming Heterogeneous Data with Database Middleware: Beyond Integration", *IEEE Data Engineering Bulletin*, 1991, pp. 31-36.

[Hoa77] Hoare, C. A. R., "An Axiomatic Approach to Computer Programming", *Communications of the ACM*, Vol. 12, No. 10, 1969, pp. 576-583.

[Hul97] Hull, R., "Managing Semantic Heterogeneity in Databases: A Theoretical Perspective", *ACM Symposium on Principles of Database Systems*, Tucson, Arizona, 1997, pp. 51-61.

[IHW01] Ives, Z. G., Halevy, A. Y., and Weld, D. S., "Integrating Network-Bound XML Data", *IEEE Data Engineering Bulletin*, Vol. 24, No. 2, 2001, pp. 20-26.

[JBV98] Jones, D., Bench-Capon, T., and Visser, P., "Methodologies for Ontology Development", *Proceedings of IT and Knows Conference of the 15th IFIP World Computer Congress*, Budapest, Hungary, August, 1998.

[Jhi00] Jhingran, A., "Moving up the Food Chain: Supporting E-Commerce Applications on Databases", *ACM SIGMOD Record*, Vol. 29, No. 4, December 2000, pp. 50-54.

[Kee03] Keeffe, M., "CRYPTOCARD Smart Card Supports Microsoft; CRYPTOCARD's SC-1 Smart Card Enables Microsoft Users to Digitally Prove Identity", Business Wire, May 1, 2002.

[KG03] Kuechler, W. and F. Grupe, "Digital Signatures: A Business View," *Information Systems Management*, Vol. 20, No.1, 2003, pp. 19-28.

[KL88] Kuhn, E., and Ludwig, T., "VIP-MDBMS: A Logic Multidatabase System", *Proceedings of the International Symposium on Database in Parallel and Distributed Systems*, Austin, TX, 1988, pp. 190-201.

[Kle01] Klein, M., "Combining and Relating Ontologies: An Analysis of Problems and Solutions", in Gomez-Perez, A., Gruninger, M., Stuckenschmidt, H., and Uschold, M., editors, *Workshop on Ontologies and Information Sharing,* Seattle, USA, August 2001.

[KLSS95] Kirk, T., Levy, A. Sagiv, Y., and Srivastava., D., "The Information Manifold", *Proceedings of AAAI Spring Symposium on Information Gathering in Distributed Heterogeneous Environment*, Menlo Park, USA, 1995.

[KMR04] Knublauch, H., Musen, M. A., Rector, A. L., "Editing Description Logic Ontologies with the Protégé OWL Plugin", *Proceedings of the International Workshop on Description Logics (DL2004)*, Whistler, BC, Canada, 2004.

[Kos97] Kosiur, D., *Understanding Electronic Commerce*, Microsoft Press, 1997.

[LA87] Litwin, W., and Abdellatif, A., "An Overview of the Multidatabase Manipulation Language MDSL", *Proceedings of the IEEE*, Vol. 75, No. 5., 1987, pp. 621-632.

[Lam78] Lamport, L., "Time, Clocks, and the Ordering of Events in a Distributed System", *Communications of the ACM*, Vol. 21, No. 7, 1978, pp. 558-565.

[Law01] Lawrence, R., "Automatic Conflict Resolution to Integrate Relational Schema", *PhD Thesis,* Department of Computer Science, University of Manitoba, May 2001.

[LB99] Lee, J., and Baik, D., "SemQL: A Semantic Query Language for Multidatabase Systems", *Proceedings of the 8th International Conference on Information Knowledge Management (CIKM'99)*, Kansas City, MO, November 1999, pp. 259-266.

[LBGR99] Lee, M., Bressan, S., Goh, G. H., and Ramakrishnan, R., "Integration of Information from Disparate Sources: A Short Survey", *Workshop on Logic of Programming and Distributed Knowledge Management*, UK, April 1999, pp. 155-158.

[LC00] Li, W. and Clifton, C., "SEMINT: A Tool for Identifying Attribute Correspondences in Heterogeneous Databases Using Neural Networks", *Data and Knowledge Engineering,* Vol. 33, 2000, pp. 49-84.

[LCL00] Li, W., Clifton, C. and Liu, S., "Database Integration Using Neural Networks: Implementation and Experiences", *Knowledge and Information Systems,* Vol. 2, No. 1, 2000, pp. 73-96.

[Len02] Lenzerini, M., "Data Integration: A Theoretical Perspective", *ACM PODS*, Madison, Wisconsin, June 2002, pp. 233-246.

[Lev01] Levy, A., "Logic-Based Techniques in Data Integration", *In J. Minker, Editor, Logic Based Artificial Intelligence,* Kluwer Publishers, 2001, pp. 575-595.

[Lev99] Levy, A., "More on Data Management for XML", University of Washington, May 9, 1999. Available at: http://www.cs.washington.edu/homes/alon/widom-response.html

[Lew91] Lewis, J. W., "Wrappers: Integration Utilities and Services for the DICE Architecture", *Proceedings of the Second National Symposium on Concurrent* Engineering, Concurrent Engineering Research Center, Morgantown WV, 1991, pp. 445-457.

[LG01] Lacher, M. S., and Groh, G., "Facilitating the Exchange of Explicit Knowledge through Ontology Mappings", *Proceedings of the 14th International FLAIRS conference*, Key West, Florida, 2001.

[LNE89] Larson, J. A., Navathe, S. B., and Elmasri, R., "A Theory of Attribute Databases with Application to Schema Integration", *IEEE Transaction on Software Engineering*, Vol. 15, No. 4, 1989, pp. 449-463.

[LR82] Landers, T., and Rosenberg, R., "An Overview of Multidatabase", *Proceedings of the 2nd International Symposium for Distributed Databases*, 1982, pp. 153-183.

[LRO96] Levy, A. Y., Rajaraman, A., and Ordille, J., "Querying Heterogeneous Information Sources Using Source Descriptions", *Proceedings of the 22nd International Conference on Very Large Databases*, Bombay, India, 1996, pp. 251-262.

[LSAF05] Lu, S., Sun, Y., Atay, M., and Fotouhi, F., "On the Consistency of XML DTDs", *Data and Knowledge Engineering*, Vol. 52, 2005, pp. 231-247.

[LSK95] Levy, A. Y., Srivastava, D., Kirk, T., "Data Model and Query Evaluation in Global Information System", *Journal of Intelligent Information Systems, Special Issue on Networked Information Discovery and Retrieval*, Vol. 5, No. 2, 1995, pp. 121-143.

[LT02] Laudon, K. C. and Traver, C. G., *E-Commerce: Business, Technology and Society*, Addison Wesley, 2002.

[Lus94] Lustman, F. "Specifying Transaction Based Information Systems with Regular Expressions", *IEEE Transactions on Software Engineering*, Vol. 20. No. 3, March 1994, pp. 207-217.

[LW00] Levy, A. and Weld, D., "Intelligent Internet Systems", *Artificial Intelligence Journal,* Vol. 118, No. 1-2, 2000, pp. 1-14.

[Maa03] Maamar, Z., "Commerce, E-Commerce, and M-Commerce: What Comes Next", *Communications of the ACM*, Vol. 46, No. 12, December 2003, pp. 251-257.

[Man00] Mann, C. L., "Electronic Commerce in Developing Countries: Issues for Domestic Policy and WTO Negotiations", March 2000. Available at: http://www.iie.com/publications/wp/2000/00-3.pdf

[MAG+97] McHugh, J., Abiteboul, S., Goldman, R., Quass, D., and Widom, J., "Lore: A Database Management System for Semistructured Data", *ACM SIGMOD Record*, Vol. 26, No. 3, 1997, pp. 54-66.

[MB81] Motro, A., and Buneman, P., "Constructing Superviews", *Proceedings of the ACM SIGMOD Conference*, Ann Arbor, Michigan, May 1981, pp. 54-64.

[MBDH02] Madhavan, J., Bernstein, P., Domingos, P.; and Halevy, A. Y., "Representing and Reasoning about Mappings between Domain Models", *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI)*, Edmonton, Canada, 2002.

[ME84] Mannino, M. V., and Effelsberg, W., "Matching Techniques in Global Schema Design", *Proceedings of the IEEE COMPDEC Conference*, LA, California, 1984, pp. 418-425.

[Met97] The Metadata Coalition, "Metadata Interchange Specification" *Technical Report* Version 1.1, August 1997. Available at: http://www.mdcinfo.com/MDIS/MDIS11.html

[Mic00] Microsoft Corporation, "BizTalk Framework 2.0 – Independent Document Specification", *Technical Report*, December 2000. Available at: http://www.microsoft.com/biztalk/default.htm

[Mic03] Micromedia Products (Accessed on December 30, 2003: http://www.macromedia.com/)

[Mic94] Microsoft Corporation, *Microsoft Open Database Connectivity Software Development Kit,* Version 2.0, Microsoft Press, 1994.

[MIR93] Miller, R. J., Ioannidis, Y. E., and Ramakrishnan, R., "The Use of Information Capacity in Integration and Translation", *Proceedings of the 19th International Conference on Very Large Databases*, Dublin, Ireland, August 1993, pp. 120-133.

[MKSI96] Mena, E., Kashyap, V., Sheth, A., and Illarramendi, A., "Observer: An Approach for Query Processing in Global Information Systems Based on Interoperation Across Pre-existing Ontologies", *Proceedings of the First IFCIS International Conference on Cooperative Information Systems (CoopIS'96)*, Brussels, Belgium, June 1996, pp. 14–25.

[MLW03] Marrón, P. J., Lausen, G., and Weber, M., "Catalog Integration Made Easy", *International Conference on Data Engineering (ICDE 2003)*, Bangalore, India, March 5-8, 2003, pp. 677-679.

[MMS03] Maedche, A., Motik, B., and Stojanovic, L., "Managing Multiple and Distributed Ontologies on the Semantic Web", *The VLDB Journal*, Vol. 12, Issue 4, Nov. 2003, pp. 286-302.

[MVU84] Maier, D., Vardi, M. and Ullman, J., "On the Foundations of the Universal Relationship Model", *ACM Transactions on Database Systems* Vol. 92, No. 2, June 1984, pp. 283-308.

[Myl01] Myllymaki, J., "Effective Web Data Extraction with Standard XML Technologies", *Proceedings of the 10th International World Wide Web Conference* (*WWW10*), Hong Kong, May, 2001, pp. 689-696.

[MZ98] Milo, T., and Zohar, S., "Using Schema Matching to Simplify Heterogeneous Data Translation", *Proceedings of 24th International Conference on Very Large Data Bases (VLDB)*, NY, USA, 1998.

[Nel02] Nelson, W. B., "OFAC and ACH Compliance", *Electronic Payment Journal*, Vol. 1, Issue 1, 2002.

[NLH+98] Ng, W. K., Lim, E., Huang, C. T., Bhowmick, S., and Qin, F. Q., "Web Warehousing: An Algebra for Web Information", *Proceedings of the IEEE International Conference on Advances in Digital Libraries* (ADL'98), Santa Barbara, California, April 22-24, 1998, pp. 228-237.

[NM00] Noy, N., and Musen, M., "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment", *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI)*, Austin, Texas, 2000.

[OF01] Omelayenko, B., and Fensel, D., "A Two-Layered Integration Approach for Product Information in B2B E-Commerce", *Proceddings of the 2nd International Conference on Electronic Commerce and Web Technologies (EC-WEB 2001)*, Munich, Germany, September 2001, pp. 226-239.

[OV99] Özsu M. T., and Valduriez, P., *Principles of Distributed Database Systems*, Prentice Hall Inc. (2nd Edition), Upper Saddle River, NJ, 1999.

[PDEW97] Perkowitz, M., Doorenbos, R., Etzioni, O., Weld, D., "Learning to Understand Information on the Internet: An Example Based Approach", *Journal of Intelligent Information Systems*, Vol. 8, No. 2, 1997, pp. 133-153.

[PE95] Perkowitz, M. and Etzioni, O., "Category Translation: Learning to Understand Information on the Internet", *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada, 1995, pp. 930-936.

[PGW95] Papakonstantinou, Y., Garcia-Molina, H., and Widom, J., "Object Exchange Across Heterogeneous Information Sources", *IEEE International Conference on Data Engineering*, Taipei, Taiwan, March 1995, pp. 251-260.

[PKT92] Plat, N., Katwijk, J., and Toetenel, H., "Application and Benefits of Formal Methods in Software Development", *Software Engineering Journal*, Vol. 7, No. 5, September 1992, pp. 335-346.

[PM01] Pinto, H. and Martins, J., "Ontology Integration: How to Perform the Process", *Proceedings of IJCAI-01 Workshop on Ontologies and Information Sharing*, Seattle, USA, August 2001.

[PMMB02] Passi, K., Madria, S. K., Mohania, M., and Bhowmick, S. S., "A Model for XML Schema Integration", *Proceedings of the 13th International Conference on Database and Expert Systems Applications (DEXA 2002)*, Aix-en-Provence, France, September 2002.

[PVT02] Pierre, G., van Steen, M., and Tanenbaum, A. S., "Dynamically Selecting Optimal Distribution Strategies on Web Documents", *IEEE Transactions on Computers*, Vol. 51, Issue 6, June 2002, pp. 637–651.

[QSJ02] Quix, C., Schoop, M., and Juesfeld, M., "Business Data Management for Business-to-Business Electronic Commerce", *ACM SIGMOD Record*, Vol. 31, No. 1, March 2002, pp. 49-54.

[Raj00] Rajput, W., *E-Commerce Systems Architecture and Applications*, Artech House, Norwood MA, 2000.

[RB01] Rahm, E. and Bernstein P., "On Matching Schemas Automatically", *The VLDB Journal*, Vol. 10, No. 4, December 2001, pp.334-350.

[RR99] Ram, S. and Ramesh, V., "Schema Integration: Past, Current, and Future" *In A. Emalgarmid, M. Rusinkiewiecz, and A. Sheth, Editors, Management of Heterogeneous and Autonomous Data Systems*, Morgan Kaufmann Publishers, 1999, pp. 119-155.

[RTU01] Rosaci, D., Terracina, G., and Ursino, D., "A Semi-Automatic Technique for Constructing a Global Representation from Information Sources having Different Formats and Structure", *Proceedings of International Conference on Database and Expert Systems Applications (DEXA 2001)*, Munich, Germany, 2001, pp. 734-743.

[Sch94] Scheurer, T., *Foundations of Computing*: *System Development with Set Theory and Logic*, Addison-Wesley Publishers, Reading, MA, 1994.

[SDG92] Samuelson, P. Denber, M., and Glushko, R. J., "Developments on the Intellectual Property Front", *Communications of the ACM*, Vol. 35, No. 6, June 1992, pp. 33-39.

[SG89] Sheth, A., and Gala, S., "Attribute Relationships: An Impediment in Automating Schema Integration", *Workshop on Heterogeneous Database Systems*, Chicago, December 1989.

[SH01] Stonebraker, M., and Hellerstein, J. M., "Content Integration for E-Business", *Proceedings of ACM SIGMOD International Conference on Management of Data, Santa Barbara, California*, May 2001, pp. 552-560.

[SH98] Schmid, B., and Lindeman, M., "Elements of a Reference Model for Electronic Markets", *Proceedings of the 31st Hawaiian International Conference on Systems Sciences (HICSS-31)*, Hawaii, 1998.

[Sin98] Singh, N., "Unifying Heterogeneous Information Models", *Communications of the ACM*, Vol. 41, No. 5, 1998, pp. 37-44.

[SK92] Sull, W., and Kashyap, R. L., "A Self-Organizing Knowledge Representation Scheme for Extensible Heterogeneous Information Environment", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 4, No. 2, 1992, pp. 185-191.

[SK93] Sheth, A. and Karabatis, G., "Multidatabase Interdependencies in Industry", *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Washington, D.C., 1993, pp. 483-486.

[SKLQ01] Schoop, M., Köller, J., List, T., and Quix, C., "A Three-Phase Model of Electronic Marketplaces for Software Components in Chemical Engineering", *Proceedings of the 1st IFIP Conference on E-Commerce, E-Government, E-Business (I3E)*, Zurich, Switzerland, Kluwer Academic Publishers, 2001, pp 507-522.

[Sku95] Skuce, D., "Conventions for Reaching Agreement on Shared Ontologies", *Proceedings of the 9th Knowledge Acquisition for Knowledge-Based Systems Workshop,* Banoe, Canada, Vol. 3, Feb. 1995, pp. 1 –19.

[SL90] Sheth, A. P., and Larson, J. A., "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases", *ACM Computing Survey*, Vol. 22, No. 3, September 1990, pp. 184-236.

[SLLL97] Seo, D., Lee, D., Lee, K., and Lee, J., "Discovery of Schema Information from a Forest of Selectively Labelled Ordered Trees", *Proceedings of the Workshop on Management of Semistructured Data*, Tucson, May 1997, pp. 54-59.

[SP94] Spaccapietra, S., and Parent, P., "View Integration: A Step Forward in Solving Structural Conflicts", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 6, No. 2, 1994, pp. 258-274.

[SQS+00] Shim, J. K., Qureshi, A. A., Siegel, J. G., Siegel, R, M., and Siegal, J., *The International Handbook of Electronic Commerce*, The Glenlake Publishing Company Ltd., 2000.

[SR01] Seligman, L., and Rosenthal, A., "XML's Impact on Databases and Data Sharing", *IEEE Computer*, Vol. 34, Issue 6, 2001.

[SS91] Swatman, P.M.C., and Swatman, P.A. "Electronic Data Interchange: Organisational Opportunity, Not Technical Problem", *Proceedings of the 2nd Australian Database-Information Systems Conference "DBIS'91"*, Sydney, Australia, February 1991, pp. 290-307.

[Sta01] Standefer, R., *Enterprise XML: Clearly Explained*, Morgan Kaufmann Publishers, 2001.

[Sun98] Sun Microsystems Inc., JDBC 2.0 Standard Extension API, 1998. Available at: http://java.sun.com/products/jdbc/jdbc20.stdext.pdf

[SV03] Shiralkar, P., and Vijayaraman, B. S., "Digital Signature: Application Development Trends In E-Business", *Journal of Electronic Commerce Research*, Vol. 4, No. 3, 2003, pp. 94-101.

[SZ96] Silberschaltz, A. and Zdonik, S., "Strategic Directions in Database Systems: Breaking Out of the Box", *ACM Computing Surveys*, Vol. 28, No. 4, 1996, pp. 764-778.

[TBC+87] Templeton, M., Brill, D., Chen, A., Dao, S., Lund, E., Macgregor, R., and Ward, P., "Mermaid: A Front End to Distributed Heterogeneous Databases", *Proceedings of IEEE* (Special Issue on Distributed Database Systems), May 1987, pp. 695-708.

[TD92] Thierry-Mieg, J., and Durbin, R., "Syntactic Definitions for the ACEDB Data Base Manager", *Technical Report MRC-LMB xx.92*, MRC Laboratory for Molecular Biology, Cambridge, CB2 2QH, UK, 1992.

[TIKL04] Themistocleous, M., Irani, Z., Kuljis, J., and Love, P., "Extending the Information System Lifecycle through Enterprise Application Integration: A Case Study Experience", *Proceedings of the 37th Annual Hawaii International Conference on Systems Sciences (HICSS'04)*, Big Island, Hawaii, January 2004.

[TKM96] Tejada, S., Knoblock, C. and Minton, S., "Learning Models for Multi-Source Integration", *Proceedings of AAAI96 Spring Symposium on Machine Learning and Information Access*, Portland, Oregon, August 1996.

[TRV95] Tomasic, A., Raschid, L., and Valduriez, P., "Scaling Heterogeneous Databases and the Design of DISCO", *Proceedings of the 16th International Conference on Distributed Computing Systems*, Hong Kong, 1995.

[TS99] Thiery-Mieg, J., and Stein, L. D., "AceDB: A Genome Database Management System", *Computing in Science and Engineering*, Vol. 1, No. 3, 1999, pp. 44-52.

[Tyg96] Tygar, J. D., "Atomicity in Electronic Commerce", *Proceedings of the 15th Annual ACM Symposium on Principles of Distributed Computing (PODC '96)*, New York, May, 1996, pp. 8-26.

[UCC99] Uniform Code Council Inc., "SIL – Standard Interchange Language", *Technical Report*, January 1999. Available at: http://www.uc-council.org/e-commerce/ec-sil-general-overview.html

[UKMZ98] Uschold, M., King, M., Moralee, S., and Zorgio, Y., "The Enterprise Ontology", *Knowledge Engineering Review*, Vol. 13, No. 1, 1998.

[Ull91] Ullman, J., *Principles of Database and Knowledge-Base Systems*, Vol. 2, Computer Science Press, 1991.

[Ver03] VeriSign® Fraud Protection Services, "Securing the Enterprise Payments Network", *White Paper*, (Accessed on December 22, 2003). Available at: http://www.verisign.com/resources/wp/fraud/PreventingFraud.pdf

[Via01] Vianu, V., "A Web Odyssey: From Codd to XML", *Proceedings of ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 2001, pp. 1-15.

[W3C02] W3C, "E-Business XML (ebXML): Enabling a Global Electronic Marketplace", 2002. http://www.ebxml.org

[W3C98] W3C, "Extensible Markup Language (XML) 1.0", *Technical Report*, February 1998. http://www.w3.org/XML/

[W3C99] W3C, "HyperText Markup Language (HTML) 4.0", *Technical Report*, December 1999. http://www.w3.org/TR/html401/

[W3C99] W3C, "HTTP - Hypertext Transfer Protocol (HTML) 1.1", *Technical Report*, June 1999. http://www.w3.org/Protocols/

[Wac99] Wache, H., "Towards Rule-Based Context Transformation in Mediators", *Proceedings of the International Workshop on Engineering Federated Information Systems*, S. Conrad and W. Hasselbring and G. Saake (Eds.), Kühlungsborn, May 1999, pp. 107 - 122.

[WCK91] Wallace, D. R., Kuhn, D. R., and Cherniavsky, J. C., *Report on the NIST Workshop on Standards for the Assurance of High Integrity Software: NIST Special publication 500-190*, Computer Systems Laboratory, National Institute of Standards and technology, Gaithersburg, MD, August 1991.

[Wid95] Widom, J., "Research Problems in Data Warehousing", *Proceedings of the 4th International Conference on Information and Knowledge Management (CIKM)*, Baltimore, Maryland, November 1995.

[Wid99] Widom, J., "Data Management for XML: Research Directions", *IEEE Data Engineering Bulletin*, Vol. 22, No. 3, 1999, pp. 44-52.

[Wie92] Wiederhold, G., "Mediators in the Architecture of Future Information Integration Systems", *IEEE Computer*, Vol. 25, pp. 38-49.

[Win90] Wing, J. M., "A Specifier's Introduction to Formal Methods", *IEEE Computer*, Vol. 23, No. 9, September 1990, pp. 10-24.

[Win98] Wing, J. M., "A Symbiotic Relationship Between Formal Methods and Security", *Proceedings of IEEE Computer Security, Dependability and Assurance: From Needs to Solutions*, Williamsburg, VA, July 1998, pp. 26-38.

[Wir90] Wirsing, M., "Algebraic Specification", In J. van Leeuwen (Ed.), *Handbook of Theoretical Computer Science*, North Holland, Amsterdam, 1990.

[WSSK99] Wache, H., Scholz, T., Stieghahn, H., König-Ries, B., "An Integration Method for the Specification of Rule-Oriented Mediators", *Proceedings of the International Symposium on Database Applications in Non-Traditional Environments (DANTE'99)*, Kyoto, Japan, November 1999, pp. 109-112.

[WVV+01] Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hübner, S., "Ontology-Based Integration of Information – A Survey of Existing Approaches", *Proceedings of IJCAI-01 Workshop on Ontologies and Information Sharing*, Seattle, USA, August 2001, pp. 108-117.

[WW96] Wand, Y., and Wang, R., "Anchoring Data Quality Dimensions in Ontological Foundations", *Communications of the ACM*, Volume 39, No. 11, 1996, pp. 86-95.

[YA01] Yee, A. and Apte, A., *Integrating your E-Business Enterprise*, Sams Publishing, 2001.

[YSDK91] Yu, C., Sun, W., Dao, S., and Keirsey, D., "Determining Relationships Among Attributes for Interoperability of Multi-Database Systems", In Y. Kambayashi and M. Rusinkiewicz (Eds.): *First International Workshop on Research Issues on Data Engineering: Interoperability in Multidatabase Systems*, April 7-9, Kyoto, Japan, 1991, pp. 251-257.