An Information Integration System for Structured Documents, Web, and Databases

by

Atsuyuki Morishima

A Dissertation Presented to
the Faculty of University of Tsukuba
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Engineering

July 1998

For my parents

Acknowledgments

I do not know how to express my deep gratitude to my supervisor Professor Hiroyuki Kitagawa. I could not have begun my research, unless he accepted me as one of his students when I was a stranger in this area. When I told him that I would like to carry out research into document processing and management, he suggested me that I should consider matters relevant to structured documents and databases. Without his vision and guidance, I would not have studied integration of heterogeneous information sources. All the major topics in the dissertation have been started to study and explored with his advice. He has given me constructive suggestions, precise criticism, and kind encouragement at all times. Because he really enjoys the activities of studying and education, I have been able to enjoy the research.

I am grateful to the members of "NR/SD group," Kazunori Kato, Shan Lin, Tsuyoshi Nemoto, Kumiko Kaimasu, and Hironori Mizuguchi. I have enjoyed the weekly discussion at the "NR/SD meeting" with them. The prototype system development could not progress so fast without their discussions, positive efforts, and the feeling that we share the same goals and dreams.

I have been studying at the database laboratory of the University of Tsukuba. I am grateful to Professor Nobuo Ohbo, Professor Yuzuru Fujiwara, Professor Isao Suzuki, and Assistant Professor Kiminori Utsunomiya for giving me constant encouragement. Also, I would like to thank Takayuki Suzuki, Junichi Notoya, and Norihide Shinagawa. I have asked them a lot of questions about any things through my laboratory life.

Finally, I am grateful to the members of my dissertation committee for giving comments to improve this dissertation. Professor Hiroyuki Kitagawa, Professor Yoshihiko Ebihara, Professor Kozo Itano, Professor Seiichi Nishihara, and Professor Nobuo Ohbo served on my dissertation committee.

Abstract

Rapid advance in computer network technology has changed the style of computer utilization. Distributed computing resources over world-wide computer networks are available from our local computers. They include powerful computers and a variety of information sources. This change is raising more advanced requirements. *Integration of distributed information sources* is one of such requirements. In addition to conventional databases, structured documents have been widely used, and have increasing significance in such advanced applications as digital libraries, electronic commerce, World Wide Web, and hyper-media descriptions. Thus, integration of structured documents and conventional databases is one of the most important issues today. However, structured documents are different from conventional databases in their self-descriptive nature and various structural constructors. And the difference makes the integration much difficult.

This dissertation proposes an information integration system for not only conventional databases but also information sources containing structured documents. In particular, structured document repositories, Web, and relational databases are chosen as target information sources.

Data models for integration of structured documents and relational databases are proposed. They are *hybrid* data models based on nested relational structures and an abstract data type for structured documents, and allow *symmetric* transformation between relational structures and structured documents. Utilization of such hybrid and symmetric data models is the most distinguishing point of the information integration system studied in this dissertation.

A query processing and optimization scheme for the integration system which utilizes the hybrid and symmetric data models is studied. The integration system is supposed to consist of software modules which communicate with each other and distributed information sources. The scheme plans to efficiently utilize query processing capability of the information sources, such as SQL processing capability of relational databases. Also, it realizes efficient manipulation of those structured documents whose size is potentially very large.

A visual user interface for the integration of structured document repositories, Web, and relational databases is proposed. In contrast to user interfaces of traditional databases, the user interface provides an interactive and visual operation framework for metadata and data, in order to cope with the largeness and complexity which arise in integration of heterogeneous and distributed information sources.

In addition to the above theoretical and academic issues, basic design of a prototype system is explained. The fact that the prototype system is implementable shows the practicability of this approach.

Contents

List of Figures

Chapter 1

Introduction

Rapid advance in computer network technology has changed the style of computer utilization. Distributed computing resources over world-wide computer networks are available from our local computers. They include powerful computers and a variety of information sources. This change is raising more advanced requirements. *Integration of distributed information sources* is one of such requirements. This requirement has motivated many researchers to develop *multidatabases*, which are intended to integrate distributed, already existing, and autonomous databases. In this context, the data objects stored in databases are assumed to obey explicit and rigid structures specified at the schema level. However, there exist many types of data objects other than those managed by such conventional databases. Among such data objects, *structured documents* have been widely used, and have increasing significance in such advanced applications as digital libraries, electronic commerce, World Wide Web, and hyper-media descriptions. Structured documents are different from data objects managed by conventional databases in the following points. (1) They are *self-descriptive*. Structural information of them is described in the documents themselves, while that of conventional databases is managed at the schema level. (2) They have *more structural constructors* than databases have. For example, they allow variant structures and optional structures to appear.

This dissertation describes development of an information integration system for not only conventional databases but also information sources containing structured documents. In particular, structured document repositories, Web, and relational databases are chosen as target information sources. The reasons are as follows: (1) Relational databases are representatives of conventional databases. They play the main roles in many practical information systems. (2) Structured document repositories mean information sources which manage collections of structured documents, in this context. For example, text retrieval systems and file systems are representatives of them. Such document repositories are widely used in practical document management systems. (3) Web has been giving huge impact on society and more and more people use it. There is a great demand for integration of Web and databases. From the technical point of view, the main components of Web are also structured documents. Web pages are usually written in HTML and XML.

As basic architecture, the integration system follows the *mediator-based approach*, which is one of the promising approaches for integration of information sources [PGMW95]. In general, a system following this approach first transforms information sources into a pivot data model to uniformly represent data objects, and integration process is performed in terms of the pivot data model.

This dissertation discusses the development of the integration system mainly from the following three aspects.

1. Pivot data models
2. Query processing and optimization issues
3. User interface problems

For the integration system of this dissertation, three pivot data models have been developed. Common features of them are as follows: (1) They are *hybrid data models* which introduce *an abstract data type for structured documents* (named *SD type*) into nested relational structures. (2)They achieve *symmetric integration* of structured documents and relational databases through operators named *converters*. The converters dynamically transform nested relational structures into structured documents and vice versa. Utilization of such hybrid and symmetric data models make the system unique compared with the other information integration systems. However, even though it is proved that the data models *logically* realize the integration of heterogeneous information sources, several problems remain for practical applications. This dissertation is also focused on other two important problems. The first problem is how queries based on such data models can be processed and optimized in the actual system. As for this problem, a query processing and optimization scheme for the mediator-based system architecture is discussed. The second problem is how to make such systems easy to use for end users. As for this problem, a visual user interface for integration of heterogeneous information sources is designed. In the following paragraphs, the above points are explained in more details.

Utilization of Hybrid Data Models

The integration system first transforms information sources into a pivot data model, and then manipulates them at the uniform representation level. For this purpose, three versions of the pivot data model — *NR/SD*, *NR/SD+*, and *WebNR/SD* — were developed. NR/SD and NR/SD+ were designed for integration of structured document repositories and relational databases. WebNR/SD was designed to incorporate the Web as an information source into the integration framework, in addition to the two types of information sources. Because NR/SD has all the basic features of these models, and the others can be considered as extended versions of NR/SD, all the three models are referred to as *NR/SD integration models*.

NR/SD integration models are hybrid data models which combine nested relational structures and abstract data type concept. More precisely, they introduce *an abstract data type for structured documents* (named *SD type*) into nested relational structures. In addition, they achieve symmetric integration through operators named *converters*. The converters dynamically transform nested relational structures into structured documents and vice versa. Thus, NR/SD integration models allow us to represent data objects in the form of both atomic values of SD type and nested relational structures.

The main reason of introducing such hybrid and symmetric data models is that they allow us to properly use different features of structured documents and databases. For example, nested relational structures are appropriate from the viewpoint of data restructuring based on the nested relational algebra operators, such as join and nest, while SD type is appropriate in manipulating a collection of structurally heterogeneous data objects. Consequently, in addition to data retrieval from structured document repositories, Web, and relational databases, NR/SD integration models enables *restructuring of heterogeneous data objects*. For example, various user views on top of structured documents can be developed in analogy to views in relational databases [CGT75]. Of course, query results can be in the form of relations, structured documents (including Web pages in the case of WebNR/SD), or their combinations.

Query Processing and Optimization Scheme

Query processing and optimization issues are important problems for practical applications. This dissertation pays attention to the following points.

(1) Efficient utilization of the local query processing capability of document repositories and relational databases.

Needless to say, the system should use the capability of the document repository for text manipulation and that of the relational database for relation manipulation. However, it is important to note that, in the NR/SD integration models, we can convert nested relational structures into structured documents and vice versa. Therefore, for example, data represented in nested relational structures may originally resides in the document repository. In this case, when some nested algebra operators are applied to the nested relational structures, it is desirable to make use of the local filtering capability of the document repository to support the operation. The query processing and optimization scheme presented in the dissertation incorporates *query rewriting rules* to cope with such issues in addition to conventional query optimization rules.

(2) Reduction of the working space and intermediate query processing cost in structured document manipulation.

In the integration system, data objects in information sources are transferred into the software module called *mediator*, and data manipulation involving different information sources is performed there. Since structured documents could contain a large amount of data, naive transfer of documents would require the mediator to have large work space. It would also bring about very large intermediate query processing cost and data transfer cost. The concept of *abstract SD value* is introduced in order to alleviate the problem. In the NR/SD integration models, it is often the case that operations of structured documents require only higher-level text elements and document structures, and that detailed parts are necessary only when the final query results are constructed. In such cases, utilization of abstract values allows us to defer transfer of detailed document data until the final query processing phase. Then, we can reduce the work space and the intermediate query processing cost of the mediator, and sometimes attain the reduction of the total data transfer cost.

User Interface Problems

In case of traditional database utilization, usually, we first browse metadata such as the schema information of a database, and then submit query statements. This simple procedure causes problems in the context of heterogeneous information integration, because of the following reasons: (1) It is difficult to identify target data objects from a sea of data objects in information sources. There are three reasons. First, the information sources include structured documents and Web. Their data structures are more complicated compared to those of relational databases. Second, in general, there is no way to bundle data objects of the same type like the extension in ODBs. Moreover, target data objects may be collected from different information sources. Without identifying target data objects, it is impossible to manipulate data objects. (2) Even if target objects of operation have been identified, the complexity of target data structure makes data operation more difficult for end users.

The visual user interface is designed to overcome the problems. The user interface models integration activity as combinations of *target discovery* and *data operation*, and provides them with visual and interactive supporting tools. Target discovery is defined as the process of discovering and extracting target data objects from a number of information sources. In this process, only data objects relevant to user requests are identified. Data operation is defined as the process of manipulating the identified target data objects to construct final answer for user requests. Because the two processes are essentially different in their purposes, different supporting tools should be provided for them.

Outline

The remainder of this dissertation is organized as follows. In Chapter 2, background of the research is described. Characteristics of structured documents written in XML is explained. And the problems which are caused in the integration of structured documents, Web, and databases are clarified through an example scenario. Chapter 3 surveys related works. In Chapter 4, architecture of the information integration system is described. In Chapter 5, pivot data models NR/SD, NR/SD+, and WebNR/SD are described. First, data structure and operators of NR/SD, and basic properties of the operators are explained. Then, differences of NR/SD+ and WebNR/SD from NR/SD are described. In Chapter 6, query processing and optimization issues in the information integration system are described. Although it is discussed in the context of WebNR/SD, the same discussion applies to the case of NR/SD and NR/SD+. In Chapter 7, the visual user interface of the information integration system is explained. In Chapter 8, basic design of a prototype information integration system is described. Chapter 9 concludes this dissertation.

Chapter 2

Background

2.1 Overview

This chapter first explains structured documents. Then, the problems which are caused in the integration of structured documents, Web, and relational databases are clarified through an example scenario.

2.2 Structured Documents

In general, documents such as papers, letters, and books have their internal structures. For example, a paper consists of its title, author information, a sequence of sections, and references. In order to interexchange not only text itself but also such structural information among different information systems, SGML (Standard Generalized Markup Language) was developed in 1986 [ISO86]. SGML documents have two main parts. The first part is called *DTD* (Document Type Definition) and describes the permitted logical structure of the document. The second part is tagged text part which conforms to the DTD. SGML has become widely recognized since U. S. Department of Defense adopted it as the standard for military documentation. Also, SGML was adopted as the basis of hypermedia description language Hytime [ISO92]. However, the most famous application of SGML would be HTML (Hyper Text Markup Language), a page description Language of World Wide Web [W3C97]. HTML documents are SGML documents whose inner document structures are prescribed by a predefined DTD. In 1997, XML (Extensible Markup Language) has been developed as a next generation language for Web page description [W3C98]. XML is a simplified version of SGML, and allows Web pages to have user-defined document types.

Figure 2.1 shows a sample XML document. The text surrounded by "`<!DOCTYPE[`" and "`]>`" in the document is the DTD. It is followed by the tagged text part. A tagged text is divided into *elements* surrounded by a *start tag* <g> and an *end tag* </g>, where g is a *generic identifier* representing the *element type*. Elements can be nested within other elements. The DTD prescribes how the elements can be hierarchically constructed by sub-elements.

In Figure 2.1, each line starting with "`!ELEMENT`" in the DTD is an *element type definition*. An element "memo" is a sequence of "prolog" and "body" (sequence structure). An element "list" consists of zero or more "item" elements (repetition structure). An element "to" can contains either "faxno" or "email" (or structure). An element "from" has sequence structure and can contain at most one "homepage" element (optional structure). An element "list" can contain other "list" elements under its "item" elements (recursive structure). The element types such as "name," "faxno," "email," and "para" are defined as having no internal structures. An element "homepage" is a special type element which represents a hypertext link. An attribute definition (starting with "`!ATTLIST`") in the DTD says that each "homepage" element is a hypertext link and specifies that it requires 'href' attribute as in the case of anchor elements of HTML.

Because structural irregularity is caused by "or" structures, "optional" structures," and so on, structured documents are often said to be semi-structured data [Abi97] [Bun97].

```
<?XML version='1.0' encoding='UTF-8' RMD='ALL'?>

<!DOCTYPE memo[
  <!ELEMENT  memo      (prolog, body)>
  <!ELEMENT  prolog    (from, to)>
  <!ELEMENT  from      (name, homepage?)>
  <!ELEMENT  to        (faxno|email)>
  <!ELEMENT  name      (#PCDATA)>
  <!ELEMENT  faxno     (#PCDATA)>
  <!ELEMENT  email     (#PCDATA)>
  <!ELEMENT  body      (para|list)*>
  <!ELEMENT  para      (#PCDATA)>
  <!ELEMENT  list      (item*)>
  <!ELEMENT  item      (para|list)>
  <!ELEMENT  homepage  ANY>
  <!ATTLIST  homepage
             xml-link #FIXED 'SIMPLE'
             href #REQUIRED              >
]>

<memo>
<prolog>
<from>
<name>Lee</name>
<homepage href="http://...xml">homepage</homepage>
</from>
<to><faxno>0298-12-3456</faxno></to>
<prolog>
<body>
<para> ...  </para> ...
<list><item>...</item><item><list>...</list></item></list>
</body>
</memo>
```

Figure 2.1: Sample XML document

2.3 Integration of Structured Documents, Web, and Databases

This section first shows an example scenario where structured document repositories, Web, and relational databases are integrated, and then, clarifies the problems caused in the integration. We assume that structured document repositories store SGML documents, and that Web pages are written in XML and have user-defined inner document structures. We have three information sources. First, we have a relational database which manages information on a CS (computer science) department of some university (say, T university). It contains a relation “Faculty” (Figure 2.2(a)). Second, we have a document repository, which contains a number of papers in the form of structured documents (Figure 2.2(b)). And finally, there are Web pages accessible from “CS department home page.” They include *PL pages*, which list publications written by the CS department faculties (Figure 2.2(c)). PL pages for different years have different inner page structures. For example, the PL page for 1996 categorizes publication items by projects, while that for 1998 annotates each publication item with some key words. However, every publication item in all PL pages is assumed to have the same substructure shown in Figure 2.3 (It uses tree representation instead of directly showing corresponding parts of the DTD). Figure 2.4 shows examples of publication items. Note that each publication item has a hypertext link to the homepage of the journal or the

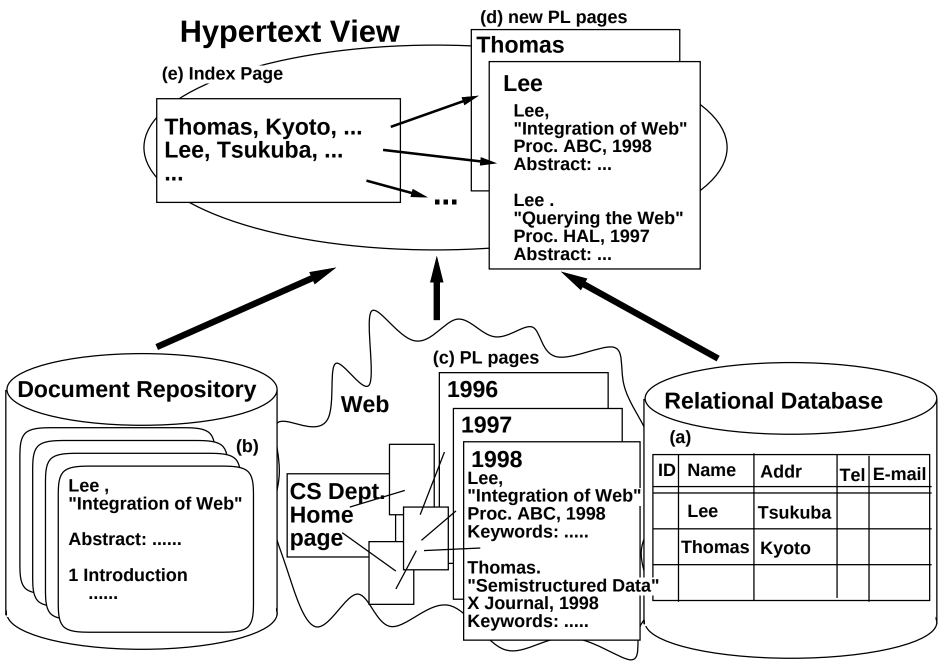


Figure 2.2: Hyper-text view over a document repository, the Web, and a relational database

conference (proceedings), where the publication were published.

The requirement here is to get a *hyper-text view* in Web over those information sources. It consists of new Web pages (new PL pages), each of which lists publications like original PL pages (Figure 2.2(d)). But they are different from original ones in the following three points. (1) Only publications with its title containing word “Web” are listed in new PL pages. (2) Each of new pages lists publications of each faculty member (not of each

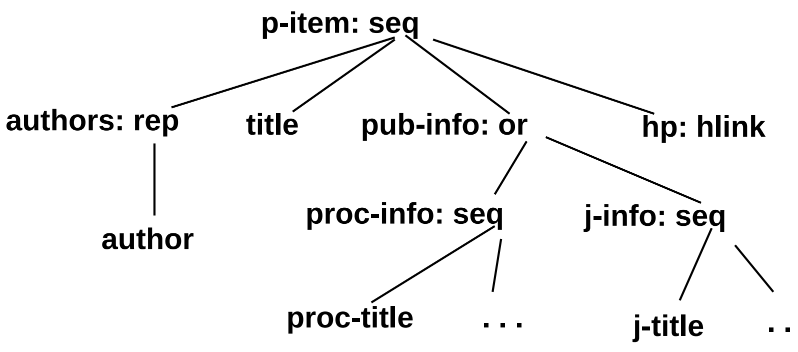


Figure 2.3: Structure of a publication item

year). (3) The new pages list not only publication information in original PL pages (including hypertext links to journals and conferences), but also publications' abstracts originally contained in papers in the document repository. In addition to the new PL pages, the view also has an index Web page. It contains hyper-text links to the new PL pages, and shows the faculty members' information originally stored in the relation "faculty" (Figure 2.2(e)).

This example scenario requires the following problems to be solved.

P1 To develop an integrated schema over the document repository, the Web, and the relational database.

P2 To amalgamate data objects originally contained in the document repository, the Web, and the relational database.

```
<p-item><authors><author>T. Johnson</author>
<author>G. Mark</author></authors><title>
On structured documents</title><pub-info><proc-
info><proc-title>CODAS</proc-title><date><month>
Nov.</month><year>1996</year></date></proc-info>
</pub-info><hp href="http://.." >CODAS'96</hp></p-item>
    ..
<p-item><authors><author>H. Smith</author>
</authors><title>Access control in a multidatabase
</title><pub-info><j-info><j-title>A-Journal
</j-title><vol>1</vol><no>7</no><date><month>June
</month><year>1994</year></date></j-info></pub-
info><hp href="http://..">A-Journal</hp></p-item>
    ..
```

Figure 2.4: Sample publication items contained in PL pages

P3 To cope with heterogeneous structures in the original PL pages.

P4 To restructure data objects and construct new Web pages and new hypertext link structures.

Chapter 3

Related Works

3.1 Overview

This chapter surveys related works. As mentioned in Section 1, the main issues of the dissertation are integration data models, the query processing and optimization scheme, and the visual user interface. First, works related to integration of heterogeneous information sources are described. Although studies on this issue can be classified according to a number of aspects, the uniqueness of the integration system is due to the data modeling framework it adopts. It is explained in details there. Second, works related to query processing and optimization schemes for integration of structured documents and databases are described. Finally, works related to visual user interfaces are described.

3.2 Integration of Heterogeneous Information Sources

The most unique point of the integration system lies in the pivot data models it uses. Thus, the related works are classified from a data modeling point of view here.

There are a number of approaches to integration of heterogeneous information resources, in particular including structured documents and databases. One approach is to extract common structures and properties of data stored in the heterogeneous information repositories and to provide a data model which can uniformly accommodate them. An advantage of this approach is that users can manipulate different kinds of data in a single uniform modeling and operational framework. Recently, CPL [BDH$^+$95], UnQL [BDHS96], and OEM [PGMW95] [Abi97] have been developed as data models which follow this approach. CPL provides a rich set of data types including lists and variants to prepare for the heterogeneity of data. (Although not for purpose of information integration, variants are also introduced into other data models [HY84] [KD95]. Data models allowing nested lists are found in [CSG94], [GZ89], and [GZC89].) In contrast of this, UnQL and

OEM use only simple nested data structures into which original data are abstracted and translated. Each of those models utilizes the comprehension [Tri91] syntax and/or SQL as the basis of their query specification scheme [AQM+97] [BDHS96] [BLS+94] [QRS+95]. Since the above models use rather abstract pivot data representations, they could be applicable to various information sources. However, the translation overhead by wrappers is not negligible. Moreover, relevant technologies of the models, such as query processing techniques, index structure, and efficient implementation scheme have not yet been matured.

Another promising approach to the integration is the *hybrid* approach. In this approach, existing data modeling frameworks fit for different information repositories are combined, or additional facilities are incorporated into a well-known existing data modeling framework. This approach intends to provide a natural combination of familiar frameworks rather than introducing a novel but unfamiliar data model. While they are not general purpose data models compared to those of the former approach, it allows us to make use of the well-known established technologies including storage management, index structure, and query processing. In addition, the translation overhead is generally smaller than the former approach. The integration through the NR/SD integration models fall in this category.

There are several studies along the line of the hybrid approach relating to structured documents and databases [ACM93] [ACM95] [BCK$^+$94] [CACS94] [CST92] [SDKR$^+$95] [VAB96] [YA94]. Abiteboul and others [ACM93] [ACM95] proposed the notion of structuring schema which specifies how data contained in text should be incorporated into databases. Atlas [SDKR$^+$95] represents document data in nested relations and provides querying facilities. Christophides and others extended the object-oriented data model of O_2 to represent SGML DTDs [CACS94]. COINS [CST92], Volz and others [VAB96], and Yan and others [YA94] all proposed schemes to jointly use the DBMS and the IR system to manage structured documents. T/RDBMS [BCK$^+$94] is similar to our approach in that it combines the relational data model and the abstract data type representing structured documents. In T/RDBMS, information inside structured documents can be viewed as a predefined collection of relations and queried in the extended SQL.

Features of the NR/SD integration models in the light of those studies are as follows.

- The NR/SD integration models provide a framework for *symmetrically* and *dynamically* amalgamating structured documents and databases. The converters transform structured documents into relational structures and vice versa. Therefore, users can restructure existing documents based on nested relational algebra and create new documents from relational structures.

- The converters enable *incremental* conversion between structured documents and nested relational structures. For example, given a collection of documents with different structures, we can extract common substructures and represent them in nested relational structures, ignoring detailed structures.

In particular, those studies provides no means of transforming relational structures (or composite objects in the case of OODBs) into structured documents. This *asymmetric* bridge between the structured document world and the relational database world prohibits us from utilizing the full power of the relational data model for manipulating structured documents. For example, we cannot apply the relational algebra to restructure structured documents. The NR/SD integration models allow us to first transform structured documents into nested relational structures with the converters, then to manipulate them with the nested relational algebra operators, and finally to transform the result data into structured documents again.

In short, NR/SD integration models are *hybrid* and *symmetric* data models. "*Hybrid*" means they combine nested relational structures and the abstract data type for structured documents, and "*Symmetric*" means they allow dynamic transformation between relational structures and structured documents. Utilization of such hybrid and symmetric data models as pivot data models is the most distinguishing point of the proposed information integration system. With the above features, the NR/SD integration models allow us to retrieve and/or restructure data stored in relational databases and structured document repositories (and Web pages in the case of WebNR/SD). Of course, the final result can be in the form of relations or structured documents (or Web pages).

WebNR/SD is the most extended version of the NR/SD integration models, and incorporates Web into the dynamic and symmetric integration world of NR/SD integration models. It can be used for integrated querying and restructuring of information residing in the Web, relational databases, and structured document repositories. Since management of the Web information and integration of heterogeneous information sources are hot topics, there are a number of related works.

Infomaster [GKD97] provides integrated access to multiple distributed heterogeneous information sources. STRUDEL [FFK$^+$97] is a Web-site management system which constructs Web pages from underlying information sources. They are similar to WebNR/SD in that they can create Web pages based on different information sources including Web pages, and that they use wrappers to translate information sources into common data representations. Infomaster uses extensions of predicates, and STRUDEL uses OEM as pivot data representations. Since they use abstract pivot data representations such as logical predicates and OEM, the same discussion as that for the first approach can be applied to them. WebNR/SD is more dedicated to integration of the Web, relational databases, and structured document repositories, and the translation overhead is far smaller.

There are a number of academic works and commercial implementations in which Web pages are used as the interface to databases [Kra97] [NS96]. There, new Web pages are constructed on the fly to show query results. Techniques such as the CGI and dedicated Web servers are also included in this category. These approaches are very weak in providing common data models for data integration. Thus, it is often the case that, except for querying relational databases and specifying HTML-based query results, users have to write application programs from scratch.

Functions of WebNR/SD include querying of the Web. Recently, a number of languages to query the Web have emerged, although they are not intended for amalgamating the Web and databases. Their queries can be based on hyper-text link structures and/or pages' contents. However, their capabilities of exploiting pages' inner structures are generally much restricted. W3QL [KS95] and WebSQL [MMM96] are SQL-like query languages, and RAW [FWM97] is an extension of the relational algebra. They are dedicated to querying and provide no way to restructure Web pages. WebLog [LSS96] is a logic-based language which enable us to query and restructure Web pages, exploiting various extensible built-in predicates. However, WebLog deals with Web pages at rather abstract level. A page in WebLog is defined as a set of *rel-infons*, each of which is a group of related information appearing in the page. Mapping between WebLog pages and actual Web pages is outside the scope of WebLog and has to be done by users. Instead, WebNR/SD directly manipulates internal page structures when querying and restructuring Web pages, and query expressions decide complete internal page structures.

Computational characteristics of Web queries are studied in [AV97] and [MM97].

The NR/SD integration models utilize as the basis the nested relational structures. Relaxing the first normal form assumption of Codd's relational data model [Cod70] was suggested by Makinouchi [Mak77] in 1977. Later, operators including NEST and UNNEST were studied by some researchers. Work by Jaeschke and Schek [JS82] is one of them. Up to now, a lot of studies on nested relational models and nested relational algebras have been done by many researchers [AB84] [AB86] [AFS89] [Col89] [Col90] [FT83] [FSTG85] [GF88] [GG88] [Guc87] [KK89]. Formalism to define the NR/SD integration models is based on the model in [TF86]. The reason the nested relational structures are utilized as the basis is that (1) they have primitive constructs to represent logical structures embedded in structured documents, and (2) they suit well to the widely-used relational database structures as natural extensions of relations. For example, Järvelin and Niemi [JN95] use nested relational structures as a basic modeling framework of structured documents, although they are not supposed to work for integration of structured documents and databases.

Integration based on object-oriented models is also another promising approach [PBE95] [RAH$^+$96] [RS97]. Although object-oriented models are powerful and practically useful frameworks, they are too rich to be used as a formal basis for discussion on the above mentioned symmetric, dynamic, and incremental data transformation. When we focus on structural aspects of object-oriented models such as complex objects, many concepts in the NR/SD integration models could be applied to object-oriented models.

Also, from broader point of view, there is an approach which realizes instance-based integration of structured documents and databases. For example, Yoshikawa and others' approach [YIU96] is to provide a general mechanism to make reference links from components of SGML documents to database objects.

3.3 Query Processing and Optimization

Although a lot of works have been done on query processing in distributed database and multidatabase environments [DKS92] [EP90] [HBP94], there is little work dedicated to query processing involving structured documents and databases. Abiteboul and others [ACM93] proposed a grammar-based approach to incorporate text data into the object-oriented database. They presented an optimization technique to push down selections and projections so that they could be executed in the text parser. Volz and others [VAB96] outlined query processing in environments where structured documents are cooperatively managed by the OODBMS and the IR system. They mainly discussed dynamic derivation of relevance values of text data which is not actually stored in the IR system. Yan and others [YA94] discussed query processing issues which occur when the external function capability of the OODBMS is used for incorporating the functionality of the IR system.

The query processing issues discussed in the previous works are different from ours in the following points. (1) The query processing issues are discussed in asymmetric systems such that a database system acts in the front end of the system. (2) They provide no means of delayed transfer of structured documents.

Although utilization of indexes and query processing cost estimation are not discussed in this dissertation, these issues in the context of integration of heterogeneous information sources raise challenging new problems. There are a number of works relevant to these issues. Index scheme for retrieval of structured documents are studied in [CM94] and [SDAMZ94]. Optimization schemes for queries including ADT functions, and cost analysis are studied in [HN96], [HS93] and [YKY$^+$91].

3.4 Visual User Interfaces

The important features of the visual user interface of the integration system are that it models integration activity as combinations of the different two processes, target discovery and data operation, and that it provides them with different visual and interactive supporting tools. Although the two processes (and their supporting tools) are amalgamated in the user interface, they are essentially different in their purposes. Therefore, the two supporting tools differs in their focuses. This section describes related works relevant to each of the two supporting tools.

As the tool for target discovery, the user interface provides interactive information exploration facility which combines browsing and querying of both metadata and data. PESTO [CHMW96] is a graphical user interface which provides seamless combinations of browsing and querying of data instance. VQL[MK93] is a visual query language which can query against not only data instance but also metadata. OPOSSUM [HIL95] is a schema management system which offers schema visualization and exploration. DataGuides [GW97] serves as dynamic schemas, generated from the semistructured databases. Browsing database structure (metadata) is one of applications of DataGuides. All the above works can be considered to provide only particular combinations of browsing and querying of both metadata and data.

As the tool for data operation, the user interface provides a visual data manipulation language named HQBE for restructuring of data objects whose structures are heterogeneous and complicated. HQBE is based on QBE [Zlo77]. Discussion about the expressive power of QBE is given in [ÖW89]. STBE [ÖMÖ89] and VISUAL [BSOO96] are visual query languages which can construct new nesting structures as the query result structures. These languages require all the target data objects for a query to have exactly the same data structure. In contrast, HQBE allows target data objects to have different data structures as long as they have some common structures essential for the data manipulations. Moreover, it is also a unique point that the data manipulation result of HQBE can have various data representations such as structured documents, Web, and relations, to suit purposes.

Chapter 4

An Information Integration System

4.1 Overview

This chapter shows basic architecture of the proposed information integration system. In this chapter, WebNR/SD is used to illustrate the role of the NR/SD integration models. The same discussion can be applied to the other models, NR/SD and NR/SD+, except that these models do not deal with Web.

4.2 Architecture

One of the promising approaches to integrate heterogeneous information sources is to use software modules or agents called *mediators* and *wrappers* [PGMW95] [PGMW96] [Wie92]. The information integration system in the dissertation follows this approach to attain integration of structured document repositories, the Web, and relational databases (Figure 4.1).

The mediator acts as a coordinator, and first it dispatches wrappers to information sources[1]. Wrappers for relational databases and document repositories communicate with local information sources directly, while the wrapper for the Web communicates with a number of Web servers to get Web pages. Wrappers provide the mediator with schema information on local information sources. In particular, wrappers for the Web and structured document repositories construct relational views over them. The mediator provides the visual user interface module with an integrated schema on those information sources. After the mediator gets the user's request from the user interface module, it analyzes the request, decomposes it into local processing requests, and sends them to wrappers. Each wrapper issues local commands to the local information source. Local commands may be simple requests to get particular data items such as Web page fetches, or complicated queries

[1]In this context, we do not assume that wrappers always reside at local information sources as in TSIMMIS [PGMW95].

which utilize querying capability of local information sources, such as SQL commands. The wrapper receives the intermediate result, translates it into WebNR/SD data representation, and sends it back to the mediator. Finally, the mediator collects data from the wrappers and produces the final result, which is returned to the user interface module.

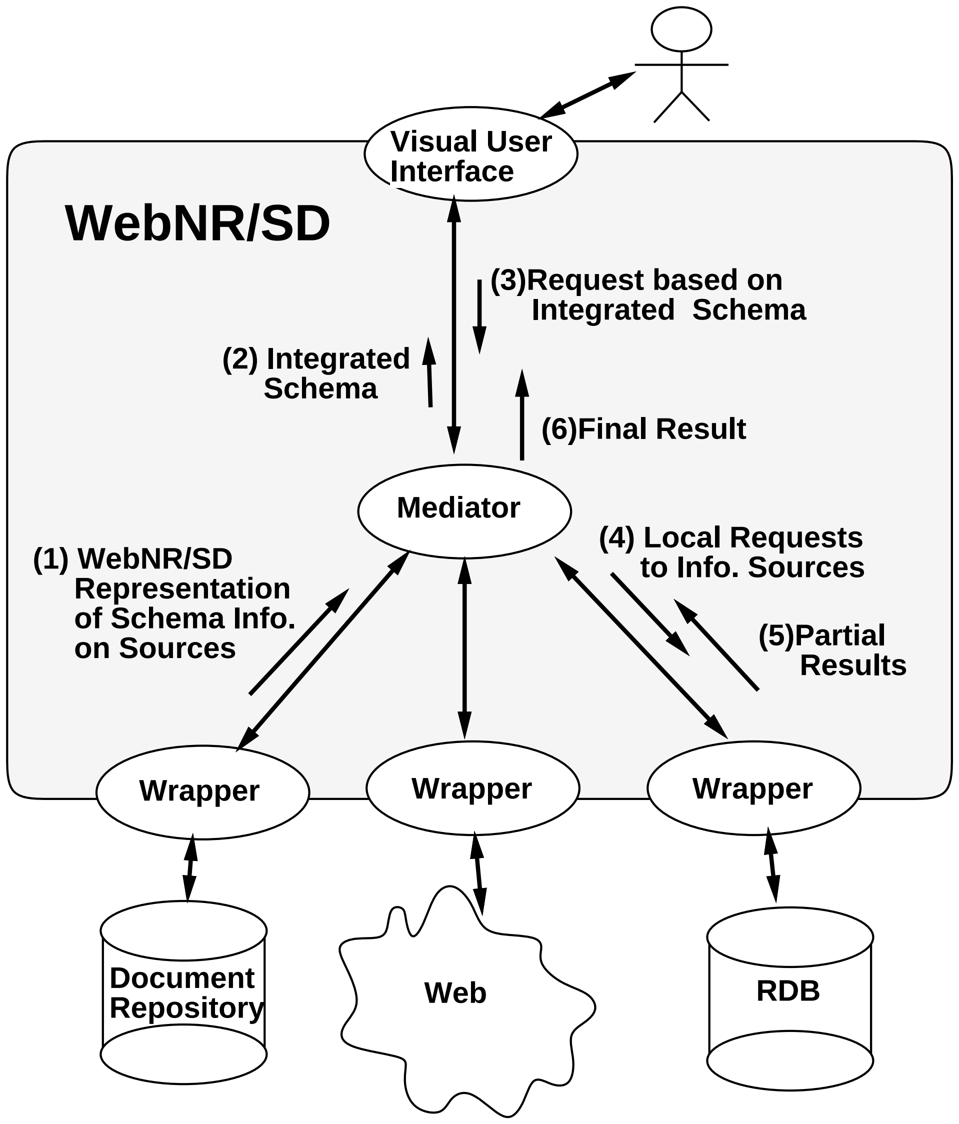


Figure 4.1: Integration environment

Chapter 5

Data Models

5.1 Overview

The integration system first transforms information sources into a pivot data model, and then manipulates them at the uniform representation level. For this purpose, three versions of pivot data models — *NR/SD*, *NR/SD+*, and *WebNR/SD* — were developed. NR/SD and NR/SD+ were designed for integration of structured document repositories and relational databases. WebNR/SD was designed to incorporate the Web as an information source into the integration framework in addition to the two types of information sources. In this chapter, first, the common basic features of the NR/SD integration models are described. Then, each model is more precisely explained.

5.2 Basic Concepts

The NR/SD integration models are hybrid data models which combine nested relational structures and abstract data type concept. More precisely, they introduce *an abstract data type for structured documents* (named *SD type*) into nested relational structures. The NR/SD integration models treat raw structured documents as atomic values of SD type. They provide the nested relational algebra operators, and a number of functions associated with SD type to retrieve text elements contained in structured documents. In addition, they achieve symmetric integration through operators named *converters*. The converters dynamically transform nested relational structures into structured documents and vice versa.

In this section, first, the data structures of the NR/SD integration models are defined. Then, the concept of converters are explained.

5.2.1 NR/SD Data Structures

The nested relational structures of the NR/SD integration models are defined here following the formalism of Fischer and Thomas [FT83] [TF86]. A *relation scheme* S is a set of rules of the form $A_i = (A_1^i, \ldots, A_n^i)$. An example of relation scheme T is as follows.

$$T = \{A = (B, C, D), D = (E, F)\}$$

T has two rules. $A, B, \ldots, F$ are called *attributes*. We call attributes which appear on the left side of some rules, namely A and D, *higher-order attributes*, and the others, namely B, C, E, and F, *zero-order attributes*. Let E_S denote the set of attributes in S, namely $E_T = \{A, B, C, D, E, F\}$. Each attribute can appear at most once on the right side of some rule and also on the left side of another rule. S must have just one *external attribute*, denoted by R_S, which appears only on the left side of some rule, namely $R_T = A$.

Instances are defined for each attribute. If A_i is a zero-order attribute, an instance of A_i is a value from the set $dom(A_i)$, called the *domain* of A_i. As defined in Subsection 5.2.2, $dom(A_i)$ can be the *structured document type* as well as ordinary primitive data types such as Integer and String. Values of the structured document type are called *SD values*, and values in the other domains are called *ordinary values*. If A_i is a higher-order attribute and $A_i = (A^i_1, A^i_2, \ldots, A^i_n)$, then an instance of A_i is a set of tuples such that each component of a tuple is an instance of A^i_j. Instances of higher-order attributes are called *composite values*.

A			
B	C	D	
		E	F
abc	`<table><dep>` `Department...`	1	...
		2	...
def	...	3	...
		4	...

Figure 5.1: Relation $\langle T, r_0 \rangle$

The *relation* $\langle S, r \rangle$ is a pair of the relation scheme S and an instance r of R_S. Figure 5.1 shows relation $\langle T, r_0 \rangle$ in tabular form. Here, $dom(B)$ is String, $dom(E)$ is Integer, and $dom(C)$ and $dom(F)$ are the structured document type defined in Subsection 5.2.2. Often we refer to the relation simply by its instance r when there is no ambiguity.

5.2.2 SD Type

SD type is an abstract data type to handle structured documents. A value of the *structured document type* (or *SD type*) is a pair of a *DTD* (Document Type Definition) and text in which tags are embedded according to the DTD. We call a value of SD type an *SD value*.

Figure 5.2 shows the example SD value corresponding to the XML document in Figure 2.1. The DTD is in the upper box. Inside the lower box is the tagged text. Repetition, sequence, optional, hypertext link structures, and #PCDATA in Figure 2.1 is represented as **rep**, **seq**, **opt**, **hlink**, and **text** in Figure 5.2, respectively. If the definition of an element type is omitted as one of "para," the element type is considered to have such a definition as "para=**text**." It is important that all SD values belong to the same domain even if they have different DTDs.

Although all the NR/SD integration models deal with SD type, permitted element structures in the DTD vary among those models. The element structures in NR/SD are restricted to **seq**, **rep**, **or**, and recursive structures. NR/SD+ allows all the element structures which appear in SGML documents. WebNR/SD augments the element structures in NR/SD+ by a special element structure named **hlink** structure to represent hypertext links between Web pages.

From now on, linear representation of DTDs is used for notational convenience. For example, DTD { a=**seq**(b,c), b=**rep**(d), d=**seq**(e,f), c:=**or**(g,h) } is represented as "a: **seq**(b: **rep**(d: **seq**(e,f)), c: **or**(g,h))."

Note that linear representation of DTDs has several restrictions in expressiveness. For example, it allows no recursive structure. However, the following discussions can be applied to SD values having general DTDs.

Functions Associated with SD Type

SD type has a number of associated functions. They are based on the region algebra [Bur91] [CCB95] [CM95], and are used to retrieve elements contained in SD values. In addition to ordinary text retrieval, they facilitate element retrieval which depends on document structures and information embedded in tags.

A *region* is a contiguous part of text. In this context, we only consider regions which correspond to elements. The region algebra is a set-at-a-time algebra. An expression e of the region algebra is generated by the following rule:

$$\begin{aligned} e \rightarrow\ & R_i | e \cup e | e \cap e | e - e \\ & | e \supset e | e \subset e | e < e | e > e | \sigma[w](e) | (e) \end{aligned}$$

where R_i is a generic identifier (e.g. "para"), "**A**," or "**I**." If R_i is a generic identifier, the region algebra expression "R_i" returns the set of elements whose generic identifiers are R_i. If R_i is "**A**," it returns the set of all elements which appear in the tagged text. If R_i is "**I**," it returns a singleton set which contains one element corresponding to the whole tagged text. Union ($\cup$), intersection ($\cap$), difference ($-$) are ordinary set operators. The *including* ($\supset$), *included* ($\subset$), *follows* ($>$), and *precedes* ($<$) operators are defined as follows:

$$R \supset S = \{r \in R : \exists s \in S, r \supset s\}$$

$$R \subset S = \{r \in R : \exists s \in S, r \subset s\}$$

$$R > S = \{r \in R : \exists s \in S, r > s\}$$

$$R < S = \{r \in R : \exists s \in S, r < s\}$$

where $r \supset s$ holds when element r strictly includes element s, $r > s$ holds when r follows s (i.e. the begin position of r is after the end position of s), and $r \subset s$ and $r < s$ are defined in a similar way. The selection (σ) operator selects elements which include occurrences of the word "w."

For example, for the tagged text "`<c> <a> <b> w1 w2 </b> <b> w3 </b> <b> w4 w1 </b> </a> <b> w1 w5 </b> </c>`", the region algebra expression "a" returns the set of elements {"`<a> <b> w1 w2 </b> <b> w3 </b> <b> w4 w1 </b> </a>`"}, "b" returns {"`<b> w1 w2 </b>`," "`<b> w3 </b>`," "`<b> w4 w1 </b>`," "`<b> w1 w5 </b>`"}, "**A**" returns { "`<b> w1 w2 </b>`," "`<b> w3 </b>`," "`<b> w4 w1 </b>`," "`<b> w1 w5 </b>`, " "`<a> <b> w1 w2 </b> <b> w3 </b> <b> w4 w1 </b> </a>`," "`<c> <a> <b> w1 w2 </b> <b> w3 </b> <b> w4 w1 </b>`

`</a> <b> w1 w5 </b> </c>`"}, "**I**" returns {"`<c> <a> <b> w1 w2 </b> <b> w3 </b> <b> w4 w1 </b> </a> <b> w1 w5 </b> </c>`"}, and "σ["w1"](b) $\subset$ a" returns {"`<b> w1 w2 </b>`," "`<b> w4 w1 </b>`" }.

Text retrieval functions associated with SD type are defined as they return only outer-most elements in the results of region algebra expressions. For example, although the results of "σ["w1"](**A**)" is { "`<b> w1 w2 </b>`," "`<b> w4 w1 </b>`," "`<b> w1 w5 </b>`, " "`<a> <b> w1 w2 </b> <b> w3 </b> <b> w4 w1 </b> </a>`," "`<c> <a> <b> w1 w2 </b> <b> w3 </b> <b> w4 w1 </b> </a> <b> w1 w5 </b> </c>`"}, the corresponding text retrieval function returns {"`<c> <a> <b> w1 w2 </b> <b> w3 </b> <b> w4 w1 </b> </a> <b> w1 w5 </b> </c>`"} since the element contains all the other elements in the result of the expression.

5.2.3 Converters

The NR/SD integration models have operators named *converters*. Converters realize *dynamic*, *bidirectional*, and *partial* transformation between SD values and nested relational structures. First, "*dynamic* transformation" means that the transformation is performed dynamically, when converters are applied. The NR/SD integration models do not statically map text elements in structured documents to nested relational structures. Second, "*bidirectional*

memo	=	**seq**(prolog, body)
prolog	=	**seq**(from, to)
from	=	**seq**(name, **opt**(homepage))
to	=	**or**(faxno, email)
body	=	**rep**(**or**(para,list))
list	=	**rep**(item)
item	=	**or**(para, list)
homepage	=	**hlink**

```
<memo>
<prolog>
<from>
<name>Lee</name>
<homepage href="http://...xml">homepage</homepage>
</from>
<to><faxno>0298-12-3456</faxno></to>
<prolog>
<body>
<para> ...  </para> ...
<list><item>...</item><item><list>...</list></item></list>
</body>
</memo>
```

Figure 5.2: Sample SD value

transformation" means that the transformation is performed not only from SD values to nested relational structures, but from nested relational structures to SD values. Finally, "*partial* transformation" means that the target of transformation can be some part of data structure. For example, in Figure 5.3, converters transform the data structure only under attribute C. Therefore, some parts of data can be represented as nested relational structures, and the other parts as SD values. This allows us to control to what degree data structure appears in the nested relational schema. Namely, the only nested relational structures are visible in the schema of nested relations, and document structures are embedded inside SD values, and invisible from the schema of nested relations.

The combination of converters and the other operators allows us the data manipulations which use advantages of both nested relational structures and structured documents as follows.

1. By transforming SD values into nested relational structures, the nested relational algebra operators can be used to manipulate structured documents. Manipulations of a set of documents and data restructuring are possible.

2. By transforming nested relational structures into SD values, text retrieval functions can be used to retrieve information from nested rela-

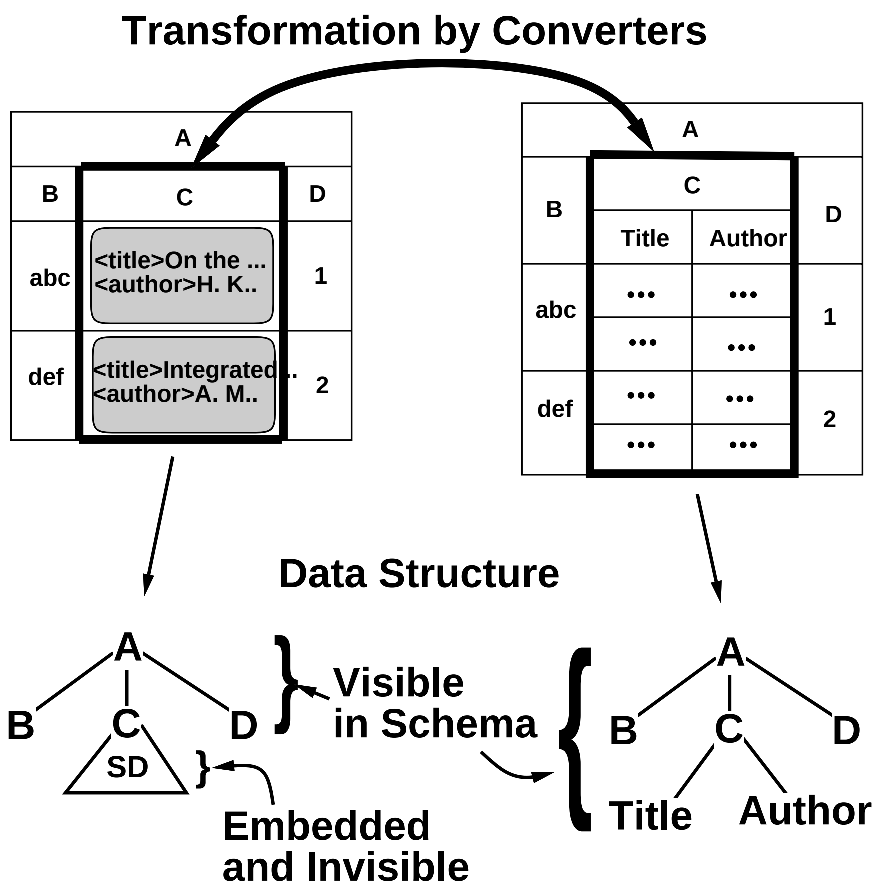


Figure 5.3: Concept of converters

tional structures.

3. The NR/SD integration models allow uniform operations of structurally heterogeneous objects. As mentioned before, the data structure only of nested relations is visible in the nested relational schema, and that of SD values is invisible. Therefore, structurally heterogeneous objects could have the same nested relational schema, with the common structure appearing in the schema, and the other structure embedded in SD values.

5.3 NR/SD

NR/SD [MK97a] [MK97b] is the initial version of the NR/SD integration models. The design principle of converters in NR/SD is to give mapping rules between data constructs in nested relational structures and element structures in structured documents. As mentioned before, the elements structures in NR/SD are restricted to **seq**, **rep**, **or**, and recursive structures.

5.3.1 Converters

Converters of NR/SD are *Rep-unpack*, *Seq-unpack*, *Rep-pack*, *Seq-pack*, *Or-append*, and *Or-remove*. The converters transform SD values into nested relational structures and vice versa. XX-unpacks extract the top-level structures embedded in SD values and represent them in the nested relational structures. XX-packs attain the conversions in the opposite direction. Or-append and Or-remove are mainly used to make some preparations for XX-unpacks and XX-packs.

Rep-unpack

Rep-unpack (**RU**) takes a relation containing SD values, and extracts the repetition (**rep**) structures at their roots. For example, relation r_1 (Figure 5.4) is transformed into relation r_2 (Figure 5.4) by the following Rep-unpack:

$$r_2 := \mathbf{RU}_{B:(C,D),E}(r_1).$$

Definition 1. Let $\langle S, r\rangle$ be a relation. Assume that S has the rule $R_S = (A_1, \ldots, A_m)$, $dom(A_i) = SD$ for some $1 \leq i \leq m$, and $\forall t \in r \exists g, d, c(t[A_i] =$

$\langle g : \mathbf{rep}(d), c\rangle)$. Then, $\mathbf{RU}\ _{A_i:(O,B),G}(\langle S, r\rangle) = \langle S', r'\rangle$, where B, O and G are new attributes,

$$\begin{array}{rcl} S' & = & (S - \{R_S = (A_1, \ldots, A_m)\}) \\ & & \quad \cup\{R_S = (A_1, \ldots, A_m, G), A_i = (O, B)\}, \\ r' & = & \{t \mid \exists u \in r, \exists g, d, c, n(u[A_i] = \langle g : \mathbf{rep}(d), c\rangle \\ & & \quad \wedge n = \#\mathit{sub\text{-}el}(c) \wedge t = u \text{ except } t[G] = g \text{ and} \\ & & \quad t[A_i] = \{(1, \langle d, \mathit{sub\text{-}el}(c, 1)\rangle), \ldots, \\ & & \qquad\qquad\qquad (n, \langle d, \mathit{sub\text{-}el}(c, n)\rangle)\})\}, \end{array}$$

#sub-el(c) is the number of the direct sub-elements of c, and *sub-el(c, i)* is the i-th direct sub-element of c. $\square$

Seq-unpack

Seq-unpack (**SU**) takes a relation containing SD values and extracts the sequence (**seq**) structures at their roots. For example, relation r_3 (Figure 5.5) is transformed into relation r_4 (Figure 5.5) by the following Seq-unpack:

$$r_4 := \mathbf{SU}_{B=(C,D),E}(r_3).$$

Definition 2. Let $\langle S, r\rangle$ be relation. Assume that S has the rule $R_S = (A_1, \ldots, A_m)$, $dom(A_i) = SD$ for some $1 \le i \le m$, and $\exists k \forall t \in r \exists g, d_1, \ldots, d_k, c(t[A_i] = \langle g : \mathbf{seq}(d_1, \ldots, d_k), c\rangle)$. Then,

r_1:

A	B
1	⟨ a:**rep**(b:**or**(c:**seq**(d,e),f)) , “`<a><b><c><d>T1</d><e>T2</e></c></b> <b><f>T3</f></b> <b><c><d>T4</d><e>T5</e></c></b></a>`”⟩
2	⟨ g:**rep**(h:**seq**(i,j,k)) , “`<g><h><i>T6</i><j>T7</j><k>T8</k></h> <h><i>T9</i><j>T10</j><k>T11</k></h></g>`”⟩

r_2:

A	B		E
	C	D	
1	1	⟨ b:**or**(c:**seq**(d,e),f) , “`<b><c><d>T1</d><e>T2</e></c></b>`”⟩	a
	2	⟨ b:**or**(c:**seq**(d,e),f) , “`<b><f>T3</f></b>`”⟩	
	3	⟨ b:**or**(c:**seq**(d,e),f) , “`<b><c><d>T4</d><e>T5</e></c></b>`”⟩	
2	1	⟨ h:**seq**(i,j,k) , “`<h><i>T6</i><j>T7</j><k>T8</k></h>`”⟩	g
	2	⟨ h:**seq**(i,j,k) , “`<h><i>T9</i><j>T10</j><k>T11</k></h>`”⟩	

Figure 5.4: Examples of Rep-unpack and Rep-pack

$\mathbf{SU}_{A_i=(B_1,\ldots,B_k),G}(\langle S,r\rangle) = \langle S',r'\rangle$, where $B_1,\ldots,B_k$ and G are new attributes,

$$\begin{aligned} S' = (&S - \{R_S = (A_1,\ldots,A_m)\}) \\ &\cup\{R_S = (A_1,\ldots,A_{i-1},B_1,\ldots,B_k,A_{i+1},\ldots,A_m,G)\}, \end{aligned}$$

and

r_3:

A	B
1	⟨ a:**seq**(b:**rep**(c),d:**seq**(e,f)), "`<a><b><c>T1</c><c>T2</c></b>` `<d><e>T3</e><f>T4</f></d></a>`"⟩
2	⟨ g:**seq**(h:**or**(i,j),k:**rep**(l)), "`<g><h><j>T5</j></h>` `<k><l>T6</l><l>T7</l></k></g>`"⟩

r_4:

A	C	D	E
1	⟨ b:**rep**(c) , "`<b><c>T1</c>` `<c>T2</c></b>`"⟩	⟨ d:**seq**(e,f) , "`<d><e>T3</e>` `<f>T4</f></d>`"⟩	a
2	⟨ h:**or**(i, j) , "`<h><j>T5</j></h>`"⟩	⟨ k:**rep**(l) , "`<k><l>T6</l>` `<l>T7</l></k>`"⟩	g

Figure 5.5: Examples of Seq-unpack and Seq-pack

$$\begin{aligned} r' = \{t | \exists u \in r, \exists g, d_1, \ldots, d_k, c(\\ & u[A_i] = \langle g : \mathbf{seq}(d_1, \ldots, d_k), c\rangle \wedge t[G] = g \\ & \wedge t[A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_m] \\ & \qquad = u[A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_m] \\ & \wedge 1 \leq \forall l \leq k(t[B_l] = \langle d_l, \mathit{sub\text{-}el}(c, l)\rangle))\}. \qquad \Box \end{aligned}$$

Rep-pack

Rep-pack (**RP**) takes a relation containing SD values, and embeds sub-relation structures into SD values as repetition (**rep**) structures. For example, relation r_2 (Figure 5.4) is transformed into relation r_1 (Figure 5.4) by the following Rep-pack:

$$r_1 := \mathbf{RP}_B(r_2).$$

Definition 3. Let $\langle S, r\rangle$ be a relation. Assume that S has the rules $R_S = (A_1, \ldots, A_m, G)$ and $A_i = (O, B)$, $dom(B) = SD$, $\leq$ is a total order relation over $dom(O)$, and $\forall t \in r \exists d \forall v \in t[A_i] \exists c (v[B] = \langle d, c\rangle)$. Then, $\mathbf{RP}_{A_i}(\langle S, r\rangle) = \langle S', r'\rangle$, where

$$\begin{aligned}
S' = &(S - \{R_S = (A_1, \ldots, A_m, G), A_i = (O, B)\}) \\
&\cup \{R_S = (A_1, \ldots, A_m)\}, \\
r' = &\{t | \exists u \in r, \exists g, d, i_1, \ldots, i_n, c_1, \ldots, c_n (\\
&u[G] = g \wedge u[A_i] = \{(i_1, \langle d, c_1\rangle), \ldots, (i_n, \langle d, c_n\rangle)\} \\
&\wedge i_1 \leq \ldots \leq i_n \wedge t = u \text{ except} \\
&t[A_i] = \langle g : \mathbf{rep}(d), add\_tag(concat(c_1, \ldots, c_n), g)\rangle)\},
\end{aligned}$$

$concat(c_1, \ldots, c_n)$ is the concatenation of the tagged texts $c_1, \ldots, c_n$, and add_tag([tagged_text], g) is the tagged text "<g>[tagged_text]</g>." For example, add_tag("T1", a) is "<a>T1</a>." □

In the above definition, the generic identifier g is given by the corresponding value of attribute G. We can explicitly specify the generic identifier g as a parameter instead of specifying attribute G. In this case, the expression would be $\mathbf{RP}_{B,g}(r_2)$, where g is a given generic identifier. We omit the formal definition of this version of Rep_pack.

Seq-pack

Seq-pack (**SP**) takes a relation containing SD values, and embeds attribute sub-sequences into SD values as sequence (**seq**) structures. For example, relation r_4 (Figure 5.5) is transformed into relation r_3 (Figure 5.5) by the following Seq-pack:

$$r_3 := \mathbf{SP}_{B=(C,D)}(r_4).$$

Definition 4. Let $\langle S, r\rangle$ be a relation. Assume that S has the rule $R_S = (A_1, \ldots, A_m, G)$, and $dom(A_i) = SD, \ldots, dom(A_j) = SD$ for some $1 \leq i \leq j \leq m$. Then, $\mathbf{SP}_{B=(A_i,\ldots,A_j)}(\langle S, r\rangle) = \langle S', r'\rangle$, where B is a new attribute,

$$\begin{aligned} S' = (&S - \{R_S = (A_1, \ldots, A_m, G)\}) \\ &\cup \{R_S = (A_1, \ldots, A_{i-1}, B, A_{j+1}, \ldots, A_m)\}, \end{aligned}$$

and

$$
\begin{aligned}
r' = \{t | \exists u \in r, \exists g, d_i, \ldots, d_j, c_i, \ldots, c_j(\\
\quad u[G] = g \wedge u[A_i] = \langle d_i, c_i \rangle \wedge, \ldots, \wedge u[A_j] = \langle d_j, c_j \rangle\\
\quad \wedge t[A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_m] =\\
\qquad\qquad u[A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_m]\\
\quad \wedge t[B] =\\
\quad \langle g : \mathbf{seq}(d_i, \ldots, d_j), \mathit{add\_tag}(\mathit{concat}(c_i, \ldots, c_j), g) \rangle)\}. \qquad \Box
\end{aligned}
$$

As in Rep-pack, the generic identifier g can be explicitly specified as a parameter to Seq-pack.

Or-remove

Or-remove (**OR**) takes a relation containing SD values, and removes the top "or" (**or**) structures at their roots. This operator can be used to prepare for further applications of Rep-unpack and Seq-unpack operators, which require the root structure of target SD values to be repetition (**rep**) or sequence (**seq**). For example, the following Or-remove transforms relation r_5 (Figure 5.6) into relation r_6 (Figure 5.6), to which we can apply Seq-unpack:

$$r_6 := \mathbf{OR}_{B,C}(r_5).$$

Definition 5. Let $\langle S, r \rangle$ be a relation. Assume that S has the rule $R_S = (A_1, \ldots, A_m)$, $dom(A_i) = SD$ for some $1 \le i \le m$, and $\forall t \in$

r_5:

A	B
1	⟨ a:**or**(b:**seq**(c:**rep**(d),e), f:**seq**(g,h)), "`<a><b><c><d>T1</d><d>T2</d></c><e>T2</e></b></a>`"⟩
2	⟨ i:**or**(f:**seq**(g,h), j:**seq**(k,l:**or**(m,n))) , "`<i><j><k>T3</k><l><m>T4</m></l></j></i>`"⟩
3	⟨ o:**or**(p:**seq**(q,r), f:**seq**(g,h)), "`<o><p><q>T5</q><r>T6</r></p></o>`"⟩

r_6:

A	B	C
1	⟨ b:**seq**(c:**rep**(d),e) , "`<b><c><d>T1</d><d>T2</d></c><e>T2</e></b>`"⟩	a
2	⟨ j:**seq**(k,l:**or**(m,n)) , "`<j><k>T3</k><l><m>T4</m></l></j>`"⟩	i
3	⟨ p:**seq**(q,r) , "`<p><q>T5</q><r>T6</r></p>`"⟩	o

Figure 5.6: Example of Or-remove

$r \exists g, d_1, \ldots, d_k, c(t[A_i] = \langle g : \mathbf{or}(d_1, \ldots, d_k), c\rangle)$. Then, $\mathbf{OR}_{A_i,G}(\langle S, r\rangle) = \langle S', r'\rangle$, where G is a new attribute,

$$S' = (S - \{R_S = (A_1, \ldots, A_m)\}) \\ \cup\{R_S = (A_1, \ldots, A_m, G)\},$$

and

$$r' = \{t | \exists u \in r, \exists g, g_1, \ldots, g_k, d_1, \ldots, d_k, c(\\ u[A_i] = \langle g : \mathbf{or}(g_1 : d_1, \ldots, g_k : d_k), c\rangle \\ \wedge c = \text{"}\texttt{<}g\texttt{><}g_i\texttt{>}\ldots\texttt{</}g_i\texttt{></}g\texttt{>}\text{"} \wedge t = u \text{ except} \\ t[G] = g \text{ and } t[A_i] = \langle g_i : d_i, \textit{sub-el}(c, 1)\rangle)\}. \quad \square$$

Or-append

Or-append (**OA**) takes a relation containing SD values, and add the "or" (**or**) structures at their roots. This operator can be used to prepare for Rep-pack operator, which requires the target SD value set to have the same element type at their roots. For example, we cannot directly apply Rep-pack to relation r_7 (Figure 5.7). However, the following Or-append yields relation r_8 (Figure 5.7), to which we can apply Rep-pack:

$$r_8 := \mathbf{OA}_D(r_7).$$

Definition 6. Let $\langle S, r\rangle$ be a relation. Assume that S has the rules $R_S = (A_1, \ldots, A_m, G)$ and $A_i = (B_1, \ldots, B_n)$, and $dom(B_j) = SD$ for some $1 \leq j \leq n$. Then, $\mathbf{OA}_{B_j}$ $(\langle S, r\rangle) = \langle S', r'\rangle$, where

$$\begin{aligned} S' = (S - \{R_S = (A_1, \ldots, A_m, G)\}) \\ \cup\{R_S = (A_1, \ldots, A_m)\} \end{aligned}$$

and

r_7:

A	B		E
	C	D	
1	1	⟨ c:**seq**(d,e), "`<c><d>T1</d><e>T2</e></c>`"⟩	b
	2	⟨ f:**or**(g,h), "`<f><g>T3</g></f>`"⟩	
	3	⟨ c:**seq**(d,e), "`<c><d>T4</d><e>T5</e></c>`"⟩	
2	1	⟨ j:**rep**(k), "`<j><k>T6</k><k>T7</k></j>`"⟩	i
	2	⟨ l:**seq**(m,n), "`<l><m>T8</m><n>T9</n></l>`"⟩	

r_8:

A	B	
	C	D
1	1	⟨ b:**or**(c:**seq**(d,e), f:**or**(g,h)), "`<b><c><d>T1</d><e>T2</e></c></b>`"⟩
	2	⟨ b:**or**(c:**seq**(d,e), f:**or**(g,h)), "`<b><f><g>T3</g></f></b>`"⟩
	3	⟨ b:**or**(c:**seq**(d,e), f:**or**(g,h)), "`<b><c><d>T4</d><e>T5</e></c></b>`"⟩
2	1	⟨ i:**or**(j:**rep**(k), l:**seq**(m,n)), "`<i><j><k>T6</k><k>T7</k></j></i>`"⟩
	2	⟨ i:**or**(j:**rep**(k), l:**seq**(m,n)), "`<i><l><m>T8</m><n>T9</n></l></i>`"⟩

Figure 5.7: Example of Or-append

$$\begin{aligned}
r' = \{t | \exists u \in r, \exists g, d, d_1, \ldots, d_k (u[G] = g \\
&\wedge d = g : \mathbf{or}(d_1, \ldots, d_k) \\
&\wedge \{d_1, \ldots, d_k\} = \{d' | \exists w \in u[A_i], \exists c (w[B_j] = \langle d', c \rangle)\} \\
&\wedge t = u \text{ except } t[A_i] = \\
&\quad \{v | \exists w \in u[A_i], \exists d', c (w[B_j] = \langle d', c \rangle \\
&\quad\; \wedge v = w \text{ except } v[B_j] = \langle d, add\_tag(c, g) \rangle)\})\}. \qquad \Box
\end{aligned}$$

As in Rep-pack, the generic identifier g can be explicitly specified as a parameter to Or-append.

5.3.2 NR/SD Algebra

NR/SD algebra consists of the converters, nested relational algebra operators, *Rename* operator [Mai83], *Apply* operator, and *Domain translator*, as primitive operators.

Nested Relational Algebra Operators

NR/SD algebra contains operators in Figure 5.8. Selection takes selection condition p for selecting tuples. In NR/SD, applicability of some converters depends on the DTD structures of SD values. For this reason, Selection here is extended to be able to select tuples based on equality of DTDs of SD values in tuples. For example, Selection $\sigma_{\mathrm{DTD}(B)=a:\mathbf{or}(b,c)}(r_1)$ selects tuples whose DTDs of attribute B ($dom(B) = SD$) values are $a : \mathbf{or}(b, c)$.

Apply operator

NR/SD provides the *Apply* operator (denoted by α) to extract elements from SD values contained in relations. Region algebra expressions are used to give the extraction specification. Suppose that r is a relation which has an SD type attribute $attr_1$, $expr$ is a region algebra expression, and $attr_2$ is a new attribute name. Then, $\alpha_{attr_1,expr,attr_2}(r)$ adds the new attribute $attr_2$ to the original relation r. The attribute $attr_2$ stores the SD values which are returned as the result of applying the region algebra expression $expr$ to the SD value in the attribute $attr_1$. For example, if $expr$ is given as "σ["w1"]$(b) \subset a$," and the $attr_1$ value of a target tuple is ⟨ c:**seq**(a:**rep**(b),b), "`<c> <a> <b> w1 w2 </b> <b> w3 </b> <b> w4 w1 </b> </a> <b> w1 w5 </b> </c>`" ⟩, then the new $attr_2$ value is { ⟨b, "`<b> w1 w2 </b>`"⟩, ⟨ b, "`<b> w4 w1 </b>`"⟩}.

Domain Translators

In the manipulation of data in structured documents and relational databases, it is sometimes necessary to translate ordinary values such as integers and strings into primitive SD values (namely, SD values which have only one start tag and end tag.) and vice versa. For example, we need to translate a string "abc" into an SD value ⟨g, "`<`g`>abc</`g`>`"⟩ and an integer 1 into ⟨g, "`<`g`>1</`g`>`"⟩. Domain translator $\gamma_{Attr,type}(r)$ changes the domain of attribute $Attr$ into $type$. When $type$ is SD type, a generic identifier must be also specified. For example, $\gamma_{A,SD(para)}(r)$ changes the domain of attribute A into SD type and value "v" in the attribute A into SD value ⟨ para, " `<para>v</para>` "⟩.

Composite Operators

In addition to the above primitive operators, *composite* operators are defined as their combinations. Join operator $\bowtie_p$ is among them. Here, extended selection operator is defined as composite operators. It selects tuples whose SD values satisfy the given selection condition. The selection condition is specified by a region algebra expression $expr$. The extended selection $\sigma_{\rho(attr,expr)}(r)$ is defined as a composite operator $\pi_{\neg A}(\sigma_{A\neq\phi}(\alpha_{attr,expr,A}(r)))$, where $\neg A$ denotes all attributes in r other than A. The pseudo predicate $\rho(attr, expr)$ holds if $expr$ returns a non-empty set of SD values for the SD value in attribute $attr$.

For example, let r be a unary relation with attribute A and contains the set of SD values { ⟨ a:**rep**(b:**or**(c, d)), "`<a> <b> <c> w1 </c> </b> </a>`" ⟩, ⟨ a:**rep**(b:**or**(c, d)), "`<a> <b> <d> w2 </d> </b> </a>`" ⟩}. Then, $\sigma_{\rho(A,b\supset c)}(r)$ returns the singleton set of an SD value { ⟨ a:**rep**(b:**or**(c, d)), "`<a> <b> <c> w1 </c> </b> </a>`" ⟩}.

5.3.3 Basic Properties of Converters

The main concern about the converters is whether the original structures changed by a converter can be recovered again by other converters. The

Selection	$\sigma_p(r)$
Projection	$\pi_{A_{i1},\ldots,A_{im}}(r)$
Cartesian product	$r_1 \times r_2$
Nest	$\nu_{A=(B_1,\ldots,B_m)}(r)$
Unnest	$\mu_A(r)$
Union	$r_1 \cup r_2$
Difference	$r_1 - r_2$

Figure 5.8: Nested relational algebra operators

following propositions show basic properties of converters. The proofs are omitted because they can be derived from the definitions of the converters without difficulty.

Propositions 1 and 2 assure that XX-packs are reversible with XX-unpacks. Proposition 3 assures that Or-append is reversible with Or-remove.

Proposition 1. Let $\langle S, r\rangle$ be a relation. Assume that S has the rules $R_S = (A_1, \ldots, A_m, G)$ and $A_i = (O, B)$, $dom(B) = SD$, and $\forall t \in r \exists d \forall v \in t[A_i] \exists c(v[B] = \langle d, c\rangle)$. Then,

$$\mathbf{RU}_{A_i:(O,B),G}(\mathbf{RP}_{A_i}(\langle S, r\rangle)) = \langle S, r\rangle. \qquad \square$$

Proposition 2. Let $\langle S, r\rangle$ be a relation. Assume that S has the rule $R_S = (A_1, \ldots, A_m, G)$, $dom(A_i) = SD, \ldots, dom(A_j) = SD$ for some $1 \leq i \leq j \leq$

m, and B is a new attribute. Then,

$$\mathbf{SU}_{B=(A_i,\ldots,A_j),G}(\mathbf{SP}_{B=(A_i,\ldots,A_j)}(\langle S,r\rangle)) = \langle S,r\rangle. \qquad \square$$

Proposition 3. Let $\langle S,r\rangle$ be a relation. Assume that S has the rules $R_S = (A_1,\ldots,A_m,G)$ and $A_i = (B_1,\ldots,B_n)$, and $dom(B_j) = SD$ for some $1 \leq j \leq n$. Then,

$$\pi_{A_1,\ldots,A_m,G}(\nu_{A_i=(B_1,\ldots,B_n)}(\mathbf{OR}_{B_j,G}(\mu_{A_i}(\tau_{A_i}(\\ \mathbf{OA}_{B_j}(\langle S,r\rangle)))))) = \langle S,r\rangle. \quad \square$$

Propositions 4 and 5 assure that XX-unpacks are reversible with XX-packs. However, Proposition 6 says Or-remove is not always reversible with Or-append.

Proposition 4. Let $\langle S,r\rangle$ be a relation. Assume that S has the rule $R_S = (A_1,\ldots,A_m)$, $dom(A_i) = SD$ for some $1 \leq i \leq m$, $\forall t \in r\exists g,d,c(t[A_i] = \langle g : \mathbf{rep}(d),c\rangle)$, and B, O, G are new attributes. Then,

$$\mathbf{RP}_{A_i}(\mathbf{RU}_{A_i:(O,B),G}(\langle S,r\rangle)) = \langle S,r\rangle. \qquad \square$$

Proposition 5. Let $\langle S, r\rangle$ be a relation. Assume that S has the rule $R_S = (A_1, A_2, \ldots, A_m)$, $dom(A_i) = SD$ for some $1 \leq i \leq m$, $\exists k \forall t \in r \exists g, d_1, \ldots, d_k, c(t[A_i] = \langle g : \mathbf{seq}(d_1, \ldots, d_k), c\rangle)$, and $B_1, \ldots, B_k, G$ are new attributes. Then,

$$\mathbf{SP}_{A_i=(B_1,\ldots,B_k)}(\mathbf{SU}_{A_i=(B_1,\ldots,B_k),G}(\langle S, r\rangle)) = \langle S, r\rangle. \qquad \Box$$

Proposition 6. Let $\langle S, r\rangle$ be a relation. Assume that S has the rule $R_S = (A_1, \ldots, A_m)$, $dom(A_i) = SD$ for some $1 \leq i \leq m$, and G is a new attribute. Then,

$$\mu_A(\mathbf{OA}_{A_i}(\nu_{A=(A_1,\ldots,A_m)}(\mathbf{OR}_{A_i,G}(\langle S, r\rangle)))) = \langle S, r\rangle$$

does not always hold. $\Box$

5.4 NR/SD+

Although NR/SD realizes dynamic transformation between structured documents and nested relational structures, it is weak in practical applications for the following reasons.

(1) The design principle of converters in NR/SD is to give mapping rules between data constructors (such as set of tuples) in nested relational structures and element structures (such as **rep** structure) in structured documents. However, there are many possible mappings between element structures and data constructors in nested relational structures. For example, there is no reason why transformation between a **seq** element structure and a set of tuples is prohibited. The design principle of converters is not suitable if we want to make such transformations directly expressible. The reason is that we will be overwhelmed by too many kinds of converters, since practical structured documents such as SGML have a lot of element structures.

(2) When converters of NR/SD transform structured documents into nested relational structures and vice versa, the values in nested relational structures must correspond to the text elements at the second level of text element hierarchy in structured documents. Therefore, if we want to embed nested relational values into a deeper level of structured documents, or, if we want to extract text elements at a deeper level of structured documents, we need to apply converters and other operators to relations in a complicated way according to DTDs.

In order to solve the problems, NR/SD+ [MK98] was developed. NR/SD+ differs from NR/SD in the design of converters. It introduces different design principle of converters. The converters of NR/SD+ provide the means of DTD-independent (instance-based) transformation of documents and nested relational structures, according to user-specified mapping rules. When transformation of structured documents into nested relational structures is performed, the user uses instance-based *text element specification* as parameters of converters, to decide what elements are contained in the result nested relational structures. In the opposite transformation, the user specifies template documents, named *masters*, into which the values in nested relational structures are embedded. Utilization of the instance-based text element specification and the masters contributes to overcoming the problems.

5.4.1 SD Type

Because converters of NR/SD+ perform DTD-independent transformation, NR/SD+ allows SD type to have all the element types which appears in SGML documents, without introducing many converters. For example, optional structures and exception structures such as inclusion and exclusion are permitted.

5.4.2 Converters

NR/SD+ has only two primitive converters: *Unpack* and *Pack*. The converters transform SD values into nested relational structures and vice versa. *Unpack* constructs sub-relation structures which store text elements originally contained in SD values. Unpack takes *text element specification*, which is based on the region algebra, as its parameter in order to specify which elements of the SD values are to be contained in the result sub-relations. Conversely, *Pack* constructs SD values from sub-relation structures containing text elements.

Unpack

Unpack (**U**) constructs sub-relation structures which store text elements originally contained in SD values. Figure 5.9 gives an example of Unpack, where

$$r_{10} := \mathbf{U}_{B \rightarrow C(O, D[\sigma['T1'](c \subset (b \cup f))]) \ as \ x}(r_9).$$

This example constructs sub-relation structures for attribute C (with sub-attributes O and D) in the result relation r_{10}. Attribute D includes SD values representing text elements which are extracted from SD values of attribute B in r_9, according to the text element specification $\sigma['T1'](c \subset (b \cup f))$. Thus, SD values in attribute D represent such c-type text elements that they are contained in text elements of type b or f and they contain the word "T1."

SD values in attribute B of relation r_{10} contain *SD references*, denoted by "`&x.`n`;`." SD references refer to SD values stored in the sub-relation structures. The header of SD reference (i.e. "`x`" in this example) is specified as a parameter of Unpack. In r_{10}, we call SD values in attribute B *master SD values* (or *masters*), and those in attribute D *derivative SD values* (or *derivatives*) (Figure 5.10).

Definition 7. Let $\langle S, r\rangle$ be a relation. Assume that S has the rule $R_S = (A_1, \ldots, A_m)$, $dom(A_i) = SD$ for some $1 \leq i \leq m$, Then, $\mathbf{U}_{A_i \rightarrow B(O,C[e])\ as\ x}(r) = \langle S', r'\rangle$, where B, O and C are new attributes, x is a string,

$$\begin{aligned}
S' &= (S - \{R_S = (A_1, \ldots, A_m)\}) \\
&\qquad \cup\{R_S = (A_1, \ldots, A_m, B), B = (O, C)\}, \\
r' &= \{t|\exists u \in r, \exists d, c, n(u[A_i] = \langle d, c\rangle \\
&\quad \wedge n = \#rl(c,e) \wedge t = u \text{ except} \\
&\quad t[A_i] = \langle d, c^{rl(c,e)}_{SDref\_list(x,\#rl(c,e))}\rangle \\
&\quad \text{and } t[B] = \{(1, SD(\langle d, c\rangle, rl(c,e), 1)), \ldots, \\
&\qquad\qquad (n, SD(\langle d, c\rangle, rl(c,e), n))\}\}\},
\end{aligned}$$

r_9:

A	B
1	⟨ b:**rep**(c), "`<b><c>T1 T2</c><c>T3 T4</c> <c>T1 T5</c></b>`" ⟩
2	⟨ a:**seq**(d:**seq**(c,c),b:**rep**(c)), "`<a><d><c>T1</c> <c>T2</c></d><b><c>T6</c><c>T1</c></b></a>`" ⟩
3	⟨ e:**seq**(b:**rep**(c), f:**seq**(c,g)), "`<e><b><c>T1</c><c>T4 T6</c></b><f><c>T7 T1</c><g>T3</g></f></e>`" ⟩

r_{10}:

A	B	C	
		O	D
1	⟨ b:**rep**(c),"`<b>&x.1;` "`<c>T3 T4</c>&x.2;</b>`" ⟩	1	⟨ c, "`<c>T1 T2</c>`" ⟩
		2	⟨ c, "`<c>T1 T5</c>`" ⟩
2	⟨ a:**seq**(d:**seq**(c,c),b:**rep**(c)), "`<a><d><c>T1</c><c>T2</c> </d><b><c>T6</c>&x.1;</b> </a>`" ⟩	1	⟨ c, "`<c>T1</c>`" ⟩
3	⟨ e:**seq**(b:**rep**(c), f:**seq**(c,g)), "`<e><b>&x.1;<c>T4 T6</c> </b><f>&x.2;<g>T3</g></f> </e>`" ⟩	1	⟨ c, "`<c>T1</c>`" ⟩
		2	⟨ c, "`<c>T7 T1</c>`" ⟩

Figure 5.9: Examples of Unpack and Pack

$rl(c, e)$ is the region list which consists of the result regions of application of region algebra expression e to tagged text c, $\#rl(c, e)$ is the length of $rl(c, e)$, $SDref\_list(x, n)$ is a list of n SD references each having the header x, $[x.1, x.2, \ldots, x.n]$, $c^{[x_1,\ldots,x_p]}_{[y_1,\ldots,y_p]}$ is the result of replacement of x_i in text c with y_i, and $SD(v, l, i)$ is a SD value corresponding to the element which is the i th element of region list l for SD value v. □

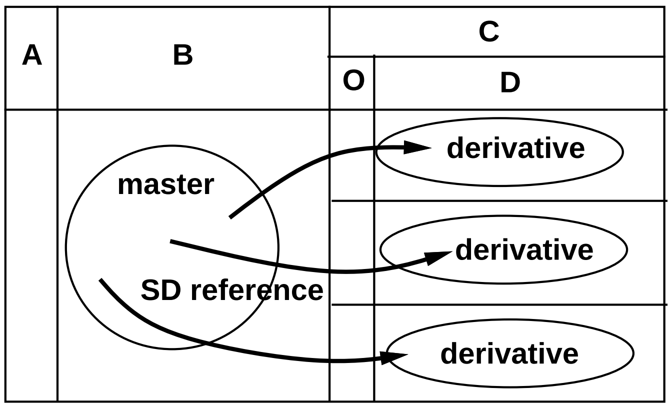


Figure 5.10: Master and derivatives

Pack

Pack (**P**) constructs SD values from sub-relation structures containing text elements. Figure 5.9 gives an example of Pack, too, where

$$r_9 := \mathbf{P}_{C(O,D)\ as\ x \rightarrow B}(r_{10}).$$

This Pack restores original SD values in attribute B from sub-relations in attribute C of r_{10} and masters in attribute B. The masters are used as templates, and SD references are replaced with text elements in derivatives.

Definition 8. Let $\langle S, r\rangle$ be a relation. Assume that S has the rules $R_S = (A_1, \ldots, A_m)$ and $dom(A_i) = SD$, $A_j = (O, B)$, $dom(O) = Integer$, and $dom(B) = SD$. Then, $\mathbf{P}_{A_j(O,B)\ as\ x \rightarrow A_i}(r) = \langle S', r'\rangle$, where x is a string, and

$$\begin{array}{rcl} S' & = & (S - \{R_S = (A_1, \ldots, A_m), A_j = (O, B)\}) \\ & & \quad \cup \{R_S = (A_1, \ldots, A_{j-1}, A_{j+1}, \ldots, A_m)\}, \\ r' & = & \{t \mid \exists u \in r, \exists d, c, d_1, c_1, \ldots, d_n, c_n (u[A_i] = \langle d, c\rangle \\ & & \qquad \wedge u[A_j] = \{(1, \langle d_1, c_1\rangle), \ldots, (n, \langle d_n, c_n\rangle)\} \\ & & \qquad \wedge t = u \text{ except } t[A_i] = \langle d, c^{SDref\_list(x,n)}_{[c_1, \ldots, c_n]}\rangle)\}. \qquad \Box \end{array}$$

5.4.3 Master Constructors

Pack operator requires masters in constructing SD values from sub-relation structures. Although masters could be created by Unpack, as shown above, sometimes we need other masters in order to get SD values which have other inner document structures. Also, if we want to construct SD values from sub-relations which are *not* directly created by Unpack, it is necessary to explicitly provide masters. *Master constructors* can be used to create masters which have DTDs and SD references consistent with existing derivatives. Here, *Sequence master constructor* (**SC**), *Repetition master constructor* (**RC**), and *Or master constructor* (**OC**) are explained. **SC**, **RC**, and **OC** provide **seq**-structured masters, **rep**-structured masters, and **or**-structured masters, respectively.

Figure 5.11 gives examples of **SC** and **RC**, where

$r_{12} := \mathbf{SC}_{C(O,D)\ as\ x,B,G}(r_{11})$, and

$r_{14} := \mathbf{RC}_{C(O,D)\ as\ x,B,G}(r_{13})$.

Figure 5.12 give example of **OC**, where

$r_{16} := \mathbf{OC}_{C(O,D)\ as\ x,B,G}(r_{15})$.

In the above examples, attribute G serves to designate generic identifiers of the top text elements of new masters in attribute B.

5.4.4 NR/SD+ Algebra

NR/SD+ algebra includes converters, master constructors, ordinary nested relational algebra operators, *Domain translator*, *Numbering operator*, and

r_{11}:

A	C		G
	O	D	
1	1	⟨ a, “<a>T1</a>” ⟩	f
	2	⟨ b, “<b>T2</b>”⟩	
	3	⟨ c, “<c>T3</c>”⟩	

r_{12}:

A	C		B
	O	D	
1	1	⟨ a, “<a>T1</a>” ⟩	⟨ f:**seq**(a,b,c), “<f>&x.1;&x.2;&x.3;</f>”⟩
	2	⟨ b, “<b>T2</b>”⟩	
	3	⟨ c, “<c>T3</c>”⟩	

r_{13}:

A	C		G
	O	D	
1	1	⟨ a, “<a>T1</a>” ⟩	f
	2	⟨ a, “<a>T2</a>”⟩	
	3	⟨ a, “<a>T3</a>”⟩	

r_{14}:

A	C		B
	O	D	
1	1	⟨ a, “<a>T1</a>” ⟩	⟨ f:**rep**(a), “<f>&x.1;&x.2;&x.3;</f>”⟩
	2	⟨ a, “<a>T2</a>”⟩	
	3	⟨ a, “<a>T3</a>”⟩	

Figure 5.11: Examples of master constructors **SC** and **RC**

r_{15}:

A	C		G
	O	D	
1	1	⟨ a, "`<a>T1</a>`" ⟩	f
	2	⟨ b, "`<b>T2</b>`"⟩	
	3	⟨ c, "`<c>T3</c>`"⟩	

r_{16}:

A	C		B
	O	D	
1	1	⟨ a, "`<a>T1</a>`" ⟩	⟨ f:**or**(a,b,c), "`<f>&x.1</f>`"⟩
	2	⟨ b, "`<b>T2</b>`"⟩	
	3	⟨ c, "`<c>T3</c>`"⟩	

Figure 5.12: Example of master constructor **OC**

Tag extractor, as primitive operators. Note that *Apply* operator is not included as primitive operators. It can be defined as a composite operator combining Unpack and other operators.

Numbering Operator

Numbering Operator $o_{C(O,D),\leq}(r)$ adds a new attribute O to relations in attribute C. Values of attribute O are natural numbers according a total order relation $\leq$ over values of attribute D. If $\leq$ is omitted, the numbers are assigned in any order.

Tag extractor

Tag extractor $\tau_{A \to G}(r)$ extends relation r by adding a new attribute G, which contains the top (outer-most) tags of SD values in attribute A.

Composite Operators

Here, some composite operators in NR/SD+ are defined. *Apply* operator $\alpha_{A,expr,B(C)}(r)$ is defined as

$$\nu_{B=(C)}(\pi_{\neg(O,B')}(\mu_D(\mathbf{U}_{B' \to D(O,C[expr])\ as\ x}(\pi_{Attr(r),B'}(r \bowtie_{B=B'} \delta_{B \to B'}(r)))))),$$

where $\pi_{\neg(O,B')}(r)$ denotes projection which removes attributes O and B' of r, and $Attr(r)$ stands for the attributes of relation r.

The followings are variations of converters defined as composite operators. They have different parameter specification syntax. They are named *composite converters.*

- $\mathbf{U}_{A_i \to (B_1[e_1]\ as\ x_1, \ldots, B_n[e_n]\ as\ x_n)}(r)$ is an extended version of Unpack. It represents a sequence of primitive Unpacks. Figure 5.13 shows an example, where $r_{18} := \mathbf{U}_{B\rightarrow(C[month]\ as\ x, D[year]\ as\ y)}(r_{17})$. Masters are in attribute B in the result relation r_{18}, and derivatives are in new attributes C and D.

- $\mathbf{P}_{(A_i,\ldots,A_j)\rightarrow B:SC,G}(r)$ involves master constructor **SC** and Pack. It creates new SD values in attribute B using SD values in attributes $A_i,\ldots,A_j$ as derivatives. $\mathbf{P}_{(C,D)\rightarrow B:SC,G}(r_{19})$ to the relation r_{19} in Figure 5.13 yields relation r_{17} in Figure 5.13.

- $\mathbf{P}_{A_i(D,\leq)\rightarrow B:RC,G}(r)$ involves master constructor **RC** and Pack. It creates new SD values in attribute B using SD values in attribute D as derivatives. If we do not care about the order of derivatives in new SD values, ordering specification $\leq$ can be omitted. Figure 5.13 shows an example, where $r_{21} := \mathbf{P}_{C(D)\rightarrow B:RC,G}(r_{20})$.

Formal definitions of composite operators including the above composite converters are in Figure 5.14.

Moreover, for all operators which require attribute names as their parameters, we can explicitly give constant values instead of attribute names. They are defined as composite operators. For example, $\mathbf{P}_{(A_i,\ldots,A_j)\rightarrow B:SC,'gi'}(r)$ specifies generic identifier $'gi'$ directly instead of attribute name G. This is defined as $\mathbf{P}_{(A_i,\ldots,A_j)\rightarrow B:SC,G}(r \times gi_{(G)})$ where $gi_{(G)}$ denotes a unary relation with attribute G, containing only one value $'gi'$.

r_{17}:

A	B
1	⟨ date:**seq**(month, year), "`<date><month>June</month><year>1970</year></date>`"⟩

r_{18}:

A	B	C	D
1	⟨ date:**seq**(month, year), "`<date>&x.1;&y.1;</date>`"⟩	⟨ month, "`<month>June</month>`"⟩	⟨ year, "`<year>1970</year>`"⟩

r_{19}:

A	C	D	G
1	⟨ month, "`<month>June</month>`"⟩	⟨ year, "`<year>1970</year>`"⟩	date

r_{20}:

A	C / D	G
1	⟨ member, "`<member>T1</member>`" ⟩	members
	⟨ member, "`<member>T2</member>`"⟩	
	⟨ member, "`<member>T3</member>`"⟩	

r_{21}:

A	B
1	⟨ members:**rep**(member), "`<members><member>T1</member><member>T2</member><member>T3</member></members>`"⟩

Figure 5.13: Example of composite operators

$$\begin{aligned}
&\mathbf{AS}_{C(O,D)\to(A_1,A_2,\ldots A_n)}(r)\\
&\quad = \pi_{\neg O}(\delta_{D\to A_1}(\sigma_{O=1}(\mu_C(r))) \bowtie \delta_{D\to A_2}(\sigma_{O=2}(\mu_C(r)))\\
&\qquad\qquad \bowtie \ldots \bowtie \delta_{D\to A_n}(\sigma_{O=n}(\mu_C(r))))\\
&\mathbf{TS}_{(A_1,A_2,\ldots A_n)\to C(O,D)}(r)\\
&\quad = \nu_{C=(O,D)}(\delta_{A_1\to D}(\pi_{\neg(A_2,\ldots,A_n)}(r)) \times 1_{(O)} \cup\\
&\qquad\qquad \ldots \cup \delta_{A_n\to D}(\pi_{\neg(A_1,\ldots,A_{n-1})}(r)) \times n_{(O)})
\end{aligned}$$

where $i_{(O)}$ denotes a unary relation with attribute O, containing only one value i.

$$\begin{aligned}
&\mathbf{U}_{A_i\to(B_1[e_1]\ as\ x_1,\ldots,B_n[e_n]\ as\ x_n)}(r)\\
&\qquad = \pi_{\neg(O_1,\ldots,O_n)}(\mu_{E_n}(\mathbf{U}_{A_i\to E_n(O_n,B_n[e_n])\ as\ x_n}(\\
&\qquad\qquad \ldots, \mu_{E_1}(\mathbf{U}_{A_i\to E_1(O_1,B_1[e_1])\ as\ x_1}(r))\ldots)))
\end{aligned}$$

where $\pi_{\neg(O_1,\ldots,O_n)}(r)$ denotes projection which removes attributes $O_1,\ldots,O_n$ of r.

$$\begin{aligned}
&\mathbf{P}_{(A_i,\ldots,A_j)\to B:SC,G}(r)\\
&\quad = \mathbf{P}_{C(O,D)\ as\ x\to B}(\mathbf{SC}_{C(O,D)\ as\ x,B,G}(\mathbf{TS}_{(A_i,\ldots,A_j)\to,C(O,D)}(r)))\\
&\mathbf{P}_{A_i(D,\leq)\to B:RC,G}(r)\\
&\quad = \mathbf{P}_{A_i(O,D)\ as\ x\to B}(\mathbf{RC}_{A_i(O,D)\ as\ x,B,G}(o_{A_i(O,D),\leq}(r)))
\end{aligned}$$

Figure 5.14: Definitions of composite operators

5.4.5 Expressive Power of NR/SD+ Algebra

The expressive power of NR/SD+ algebra contains that of NR/SD. The followings are NR/SD+ expressions equivalent to converters of NR/SD.

$$\begin{aligned}
\mathbf{RU}_{B:(O,D),G}(r) &= \delta_{E\to B}(\pi_{\neg B}(\tau_{B\to G}(\mathbf{U}_{B\to E(O,D[\mathbf{A}\subset_d \mathbf{I}])\ as\ x}(r))))\\
\mathbf{SU}_{B=(B_1,\ldots,B_n),G}(r) &= \pi_{\neg B}(\tau_{B\to G}(\mathbf{AS}_{E(O,D)\to(B_1,\ldots,B_n)}(
\end{aligned}$$

$$\mathbf{U}_{B \to E(O,D[\mathbf{A} \subset_d \mathbf{I}])\ as\ x}(r))))$$

$$\mathbf{RP}_{B:(O,D),G}(r) = \delta_{E \to B}(\mathbf{P}_{B(O,D)\ as\ x \to E}(\mathbf{RC}_{B(O,D)\ as\ x,E,G}(r)))$$

$$\mathbf{SP}_{B=(B_1,\ldots,B_n),G}(r) = \mathbf{P}_{E(O,D)\ as\ x \to B}(\mathbf{SC}_{E(O,D)\ as\ x,B,G}(\mathbf{TS}_{(B_1,\ldots,B_n) \to E(O,D)}(r)))$$

$$\mathbf{OR}_{B,G}(r) = \delta_{D \to B}(\pi_{\neg(B,O)}(\tau_{B \to G}(\mu_E(\mathbf{U}_{B \to E(O,D[\mathbf{A} \subset_d \mathbf{I}])\ as\ x}(r)))))$$

$$\mathbf{OA}_{D,G}(r) = \nu_{C=(O,D)}(\delta_{E \to D}(\mathbf{P}_{F(O_1,H)\ as\ x \to E}(\mathbf{TS}_{(D) \to F(O_1,H)}(\mu_C(\mathbf{OC}_{C(O,D)\ as\ x,E,G}(r))))))$$

5.5 WebNR/SD

WebNR/SD [MK97c] is an extension of NR/SD+. Although the Web contains a collection of structured documents as conventional structured document repositories, NR/SD+ is insufficient to deal with the Web. This is because the Web differs from conventional document repositories in the following points: (1) Web pages usually have hyper-text links to relevant pages. (2) The number of Web pages is virtually innumerable and they are autonomously changing. Therefore, it is impractical to manage all the Web pages in the world with a static database scheme.

To cope with this first point, WebNR/SD introduces a new type named the *Hlink type*. To address the second point, it provides *Navigate* and *Import* operators. They enable us to fetch Web pages from the outside Web world on demand. Navigate gets hyper-text links to the Web pages which meet user-specified conditions. Import is to obtain the contents of the Web pages as SD type values.

In addition to Navigate and Import, WebNR/SD features *Export* operator. It exports SD type values as Web pages to the outside Web world. Thus, in addition to querying different information sources, we can construct *hyper-text views* over the underlying information sources.

In the following discussion, Web pages are assumed to be written in XML, and that Web pages have user-defined inner document structures. This enables us to use tag information more effectively taking care of user-specified data semantics.

5.5.1 Hlink Type

First, WebNR/SD introduces a new element structure **hlink** into SD type in addition to the element structures permitted in NR/SD+. Elements with **hlink** structure are called *linking elements* and represent hyper-text links to Web pages just as anchor elements in HTML documents [1]. An element with **hlink** structure is assumed to have no sub-elements, like elements with **text** structure. The **hlink** structure is associated with the attribute "href," whose value designates a URL. For example, "`<a href="http://T.ac.jp/p1.xml">T Univ</a>`" is an element with **hlink** structure. A value of *Hlink type* (an Hlink value) is defined as an SD value which consists of only one element with **hlink** structure. Since Hlink values are also SD values, operators applicable to SD values can also be applied to Hlink values.

5.5.2 WebNR/SD Algebra

WebNR/SD algebra consists of NR/SD+ algebra operators and four additional operators: Export, Import, Navigate, and URL generator.

Export and Import

Export and *Import* incorporate the Web into our framework. Export (**E**) exports SD values stored in a relation to the Web world as Web pages. Import (**I**) imports Web pages residing in the Web world into a relation as SD values.

[1] Actually, linking elements are defined as elements having "xml-link" attribute in XML. Here, we assume that **hlink** structure is used to represent linking elements for simplicity.

Figure 5.15 gives an example of Export and Import, where

$r_{23} := \mathbf{E}_{B,U,L,G}(r_{22})$, and

$r_{22} := \mathbf{I}_{B,U,L,G}(r_{23})$.

$\mathbf{E}_{B,U,L,G}(r_{22})$ creates Web pages whose URLs are given in attribute U. The Web pages' contents correspond to SD values stored in attribute B of r_{22}. Attribute B in the result relation r_{23} has Hlink values referring to those pages and containing character strings originally stored in attribute L. $\mathbf{I}_{B,U,L,G}(r_{23})$ works in the opposite direction. Attribute B in the result relation r_{22} obtains SD values corresponding to Web pages which are referred to by Hlink values stored in attribute B of r_{23}.

Navigate operator

Navigate (**N**) navigates the Web according to a *path regular expression* specified as a parameter of this operator. In path regular expressions, hyper-text links are represented by $\rightarrow$ (a link whose destination and source pages reside in the same Web server) and $\Rightarrow$ (a link whose destination and source pages

r_{22}:

A	B	U	L	G
1	⟨ b:**seq**(c:**rep**(d),e) , “`<b><c><d>T1 T2</d></c><e>T3</e></b>`” ⟩	http://.../index.xml	Page A	a

r_{23}:

A	B
1	⟨ a:**hlink**, “`<a href="http://.../index.xml">Page A</a>`” ⟩

Figure 5.15: Examples of Export and Import

reside in different Web servers), while Web pages are represented by character strings (*labels*) and a period. For example, $B \rightarrow . \Rightarrow C \rightarrow D$ is a path regular expression that represents the set of paths that start a Web page B in a Web server X, followed by a page in the same server X and a page C in a different server Y($\neq X$), and end with a page D in the server Y.

If there is hyper-text link structure shown in Figure 5.16 in the Web, Figure 5.17 gives the result of Navigate under the above path regular expression, where

$$r_{25} := \mathbf{N}_{B \rightarrow . \Rightarrow C \rightarrow D, E}(r_{24}).$$

Labels in path regular expressions are used to associate Web pages with

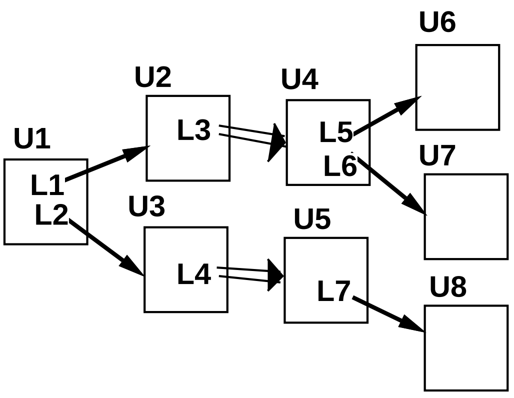


Figure 5.16: Example hyper-text link structure (Un are URLs and Ln are character strings in linking elements)

relational attributes. Figure 5.17 shows two important points: (1) Attribute E of the result relation r_{25} contains the set of paths specified by the path regular expression, where each page on the paths is represented by an Hlink value referring to the Web page instead of the contents of the page itself. (2) Links to those pages corresponding to the period do not appear in sub-relations of attribute E.

Path regular expressions can represent alternation and repetition structures. For example, $A(\rightarrow\ .)* \rightarrow B \ \mid \ A \Rightarrow B$ represents the set of paths which start with a page A, followed by one or more local links leading to B, or from a page A to B via a global link. '$*/n_1..n_2/$' can be used as syntactic sugar. For example, $A(\rightarrow\ .) * /2..3/ \rightarrow B$ is equivalent to $A \rightarrow . \rightarrow . \rightarrow B \mid A \rightarrow . \rightarrow . \rightarrow . \rightarrow B$.

We can also specify selection conditions on Web pages in path regular expressions. The selection condition is given by a region algebra expression and placed just after a label or a period. The selection condition holds if the region algebra expression returns a non-empty set of text elements for the Web page bound to the preceding label or period. For example, assume that the contents of the Web pages at URLs "U4" and "U5" in Figure 5.16 are $\langle$ e:**seq**(a:**hlink**,b:**or**(c, d),a:**hlink**), "`<e> <a href="U6"> L5 </a> <b> <c> w1 </c> </b> <a href="U7"> L6 </a> </e>`" $\rangle$, and $\langle$ f:**seq**(b:**or**(c, d), a:**hlink**), "`<f> <b> <d> w2 </d> </b> <a href="U8"> L7 </a></f>`" $\rangle$, respectively. Then, $\mathbf{N}_{B\to.\Rightarrow C[b\supset c]\to D,E}(r_{24})$ returns the relation which is same as r_{25} except that its attribute E contains only the first two subtuples of r_{25}.

Figure 5.18 shows the BNF specification of path regular expressions [2].

URL generator

URL generator (**URL**) generates new unique URLs. It can be used before Export operation in creating new Web pages. An example of **URL** is shown

[2]It is assumed that every path specified by a path regular expression gives non-empty binding to all the labels. $A \to B \mid A \to C$ is an example which violates the restriction.

r_{24}:

A	B
1	⟨ a:**hlink**, “`<a href="U1"></a>`” ⟩

r_{25}:

A	B	E	
		C	D
1	⟨ a:**hlink**, “`<a href="U1"></a>`” ⟩	⟨ a:**hlink**, “`<a href="U4">L3</a>`” ⟩	⟨ a:**hlink**, “`<a href="U6">L5</a>`” ⟩
		⟨ a:**hlink**, “`<a href="U4">L3</a>`” ⟩	⟨ a:**hlink**, “`<a href="U7">L6</a>`” ⟩
		⟨ a:**hlink**, “`<a href="U5">L4</a>`” ⟩	⟨ a:**hlink**, “`<a href="U8">L7</a>`” ⟩

Figure 5.17: Example of Navigate

in Figure 5.19, where

$$r_{27} := \mathbf{URL}_U(r_{26}).$$

$\mathbf{URL}_U(r_{26})$ extends relation r_{26} by adding a new attribute U, which contains generated URLs.

5.6 Query Specification Example

In this section, WebNR/SD is applied to the example scenario described in Chapter 2. First, I briefly outline how to deal with the problems **P1** through **P4** mentioned in Section 2.3 in WebNR/SD.

```
path_reg_expr ::= label p_expr { '|' label p_expr }
p_expr        ::= link page [ '[' condition ']' ]
                | p_expr { p_expr }
                | p_expr { '|' p_expr }
                | p_expr '*' [ '/' num '..' num '/' ]
                | '(' p_expr ')'
link          ::= '->' | '=>'
page          ::= label | '.'
```

Figure 5.18: Path regular expression syntax

r_{26}:

A	B
1	b1
2	b2

r_{27}:

A	B	U
1	b1	http://.../new/1.xml
2	b2	http://.../new/2.xml

Figure 5.19: Example of URL generator

P1 In WebNR/SD, the Web is modeled as a collection of Hlink values. The Hlink values work as anchor points to fetch necessary Web pages with Navigate and Import operators. In the example scenario, we use "CS department home page" as a starting point to fetch Web pages. The Web wrapper provides a unary relation "WWW" which only contains an Hlink value referring to the "CS department home page" (Figure 5.20). Relation "Faculty" in the relational database is represented as it is. The set of structured documents stored in the document repository is represented as a unary relation "DR" with an SD type attribute

"Doc." The relations "WWW," "Faculty," and "DR" form the integrated schema.

P2 Structured documents in document repositories are represented as SD values. Also, Once Web pages are imported, they are represented as SD values, too. Thus, amalgamation of data in the document repository, Web pages, and the relational database is achieved by converters, Domain translator, and nested relational algebra operators.

P3 In spite of structural difference among the original PL pages, Unpack allows us to extract publication items from the PL pages with a text element specification. The operation would be $\mathbf{U}_{A \rightarrow B(O,C[p\text{-}item]) \ as \ x}(r)$ if attribute A stores the original PL pages.

P4 Construction of new Web pages can be attained by a combination of converters, nested relational algebra operators, and master constructors. First, we construct (sub-)relational structures which store text elements originally contained in PL pages by using Unpack operator. Next, we restructure them using operators such as join and nest. Finally, we transform the relational structures into SD values using master constructors and Pack. Export exports them to the outside Web world and also obtains Hlink values referring to those pages. The Hlink values are used as sub-elements of the index page. Thus, new hyper-text link structures are created in a similar way.

Page
⟨ a:**hlink**, “`<a href="http://.../cs/index.xml"></a>`”⟩

Figure 5.20: Relation WWW

Figure 5.21 shows the element type definitions for publication items shown in Figure 2.4, contained in original PL pages. We assume that every PL page has a text element “page-title” which contains the word “publication-list.” Figure 5.22 shows an example SD value which correspond to a paper in the document repository. Figures 5.23 and 5.24 show DTDs for the index page and new PL pages, respectively. Then, the data manipulation request in Section 2.3 can be specified as follows. In the following WebNR/SD algebra expressions, we omit the domain translators for notational simplicity. Also, we may omit text element specifications, headers of SD references, and generic identifiers, in parameters, when there is no possibility of confusion. For example, $\mathbf{U}_{B\rightarrow(Authors[authors]\ as\ Authors, Title[title]\ as\ Title)}(r)$ is simply represented as $\mathbf{U}_{B\rightarrow(Authors,Title)}(r)$. Also, $\mathbf{P}_{(Author,Pubs)\rightarrow P\text{-}List:SC,'p\text{-}list'}(r)$ is represented as $\mathbf{P}_{(Author,Pubs)\rightarrow P\text{-}List:SC}(r)$.

(1) First, we import all the PL pages residing in the same Web server as that of the CS department home page.

$$r_{28} := \mu_C(\mathbf{I}^*_{D,U,L,G}(\mathbf{N}_{Page(\rightarrow.)*\rightarrow D[\sigma['publication\text{-}list'](page\text{-}title)],C}(WWW)))$$

p-item	=	**seq**(authors, title, pub-info, hp)
pub-info	=	**or**(proc-info, j-info)
proc-info	=	**seq**(proc-title, **opt**(venue), date)
j-info	=	**seq**(j-title, vol, no, date)
authors	=	**rep**(author)
date	=	**seq**(month, year)

Figure 5.21: Element definitions for publication items

paper	=	**seq**(title, authors, pub-info, abstract, body, ref)
authors	=	**rep**(author)
author	=	**seq**(a-name, affiliation)
pub-info	=	**or**(proc-info, j-info)
proc-info	=	**seq**(proc-title, **opt**(venue), date)
j-info	=	**seq**(j-title, vol, no, date)
date	=	**seq**(month, year)
body	=	**rep**(section)
ref	=	**rep**(ref-item)
...		

```
<paper>
<title>Integration of the Web and Heterogeneous
Information Repositories</title><authors>
<author><a-name>T. Johnson</a-name>
<affiliation>A-Univ</affiliation></author>
<author><a-name>G. Mark</a-name><affiliation>
B-Univ</affiliation></author></authors>
<pub-info><proc-info><proc-title>X Conf.
</proc-title><date><month>June</month><year>1997
</year></date></proc-info></pub-info>
<abstract> ...
```

Figure 5.22: Sample SD value which corresponds to a paper

(2) Then, relevant text elements are extracted by unpacking the imported Web pages.

$$\begin{aligned}r_{29} := {} & \mu_{As}(\mathbf{U}_{Authors \rightarrow As(O_2, Author)}(\\ & \quad \mathbf{U}_{P\text{-}Item \rightarrow (Authors, Title, Pub\text{-}Info, Hp)}(\mu_P(\mathbf{U}_{D \rightarrow P(O_1, P\text{-}Item)}(r_{28})))))\end{aligned}$$

(3) Relevant text elements contained in papers in the document repository are also extracted by Unpack operator. Then, papers whose titles contain the word "Web" and whose authors involve people belonging to T university are selected.

$$\begin{aligned}r_{30} := {} & \sigma_{Title \ni ''Web'' \wedge Affiliation = ''T-Univ''}(\mathbf{U}_{Author \rightarrow (A\text{-}Name, Affiliation)}(\\ & \quad \mu_{As}(\mathbf{U}_{Authors \rightarrow As(O_3, Author)}(\\ & \qquad \mathbf{U}_{Doc \rightarrow (Authors, Title, Pub\text{-}Info, Abstract)}(DR)))))\end{aligned}$$

(4) The relation r_{29} is joined with relations r_{30} and "Faculty," and unnecessary attributes are dropped.

$$\begin{aligned}r_{31} := {} & \pi_{Name, Addr, Tel, E\text{-}Mail, Author, Title, Pub\text{-}Info, Hp, Abstract}(\\ & \qquad \sigma_{A\text{-}Name = Author}(r_{30} \bowtie r_{29}) \bowtie_{Author = Name} Faculty)\end{aligned}$$

(5) New SD values are constructed following the DTD shown in Figure 5.24.

$$\begin{aligned}r_{32} := {} & \mathbf{P}_{P(New\text{-}P\text{-}Item) \rightarrow Pubs:RC}(\nu_{P = (New\text{-}P\text{-}Item)}(\\ & \quad \mathbf{P}_{(Title, Pub\text{-}Info, Hp, Abstract) \rightarrow New\text{-}P\text{-}Item:SC}(r_{31}))) \\ r_{33} := {} & \mathbf{P}_{(Author, Pubs) \rightarrow P\text{-}List:SC}(r_{32})\end{aligned}$$

table	=	**rep**(member)
member	=	**seq**(name, addr, tel, e-mail, a)
a	=	**hlink**

Figure 5.23: DTD for the index page

p-list	=	**seq**(author, pubs)
pubs	=	**rep**(new-p-item)
new-p-item	=	**seq**(title, pub-info, hp, abstract)
pub-info	=	**or**(proc-info, j-info)
proc-info	=	**seq**(proc-title, **opt**(venue), date)
j-info	=	**seq**(j-title, vol, no, date)
date	=	**seq**(month, year)
hp	=	**hlink**

Figure 5.24: DTD for new PL pages

(6) The SD values are exported as new PL pages, and Hlink values are set to link to those Web pages.

$$r_{34} := \mathbf{E}_{P\text{-}List,U_1,'publications','a'}(\mathbf{URL}_{U_1}(r_{33}))$$

(7) Finally, the index page is constructed following the DTD shown in Figure 5.23. It is also exported as a new Web page.

$$\begin{aligned} r_{35} &:= \mathbf{P}_{Members(Member)\rightarrow Table:RC}(\nu_{Members=(Member)}(\\ &\qquad \mathbf{P}_{(Name,Addr,Tel,E\text{-}Mail,P\text{-}List)\rightarrow Member:SC}(r_{34}))) \\ r_{36} &:= \mathbf{E}_{Table,U_2,'index\text{-}page','a'}(\mathbf{URL}_{U_2}(r_{35})) \end{aligned}$$

Chapter 6

Query Processing and Optimization

6.1 Overview

This chapter presents a query processing and optimization scheme in the information integration system. Although it is discussed in the context of WebNR/SD, the same discussion applies to the case of NR/SD and NR/SD+. In the query processing and optimizations scheme, the integration system can use *abstract SD values* instead of real SD values to reduce the work space and the intermediate query processing cost of the mediator. Moreover, the query processing and optimization scheme incorporates *query optimization rules* especially designed for the hybrid and symmetric data models, as well as conventional optimization rules. In this chapter, first, abstract SD values is explained, and then the query processing and optimization scheme is presented.

6.2 Abstraction of SD values

In this section, first, the concept of the abstract SD values is explained. Then, two operators for utilization of ASD values are introduced in addition to WebNR/SD algebra operators.

6.2.1 Abstract SD values

Since structured documents could contain a large amount of data, naive transfer of documents from the document repository would require the mediator to have large work space. The notion of *abstract SD values* (ASD values) is introduced to alleviate the problem. Operations of structured documents in WebNR/SD often require only higher-level elements and document structures, and detailed parts are necessary only to obtain the final query results. Thus, we can transfer ASD values containing only partial information required by the mediator to derive intermediate results. The use of ASD values can also reduce the intermediate query processing cost of the mediator.

r_{37}:

A	B
1	⟨ a:**seq**(b:**rep**(c), d:**seq**(e,e)), "`<a><b><c>T1</c><c>T2</c></b>` `<d><e>T3</e><e>T4</e></d></a>`"⟩

r_{38}:

A	B
1	⟨ a:**seq**(b:**rep**(c), d:UNKNOWN, "`<a><b><c>T1</c><c>T2</c></b>` `<d>` [id1, 11, 16] `</d></a>`"⟩

Figure 6.1: Example of transformation between SD values and ASD values

Relation r_{38} in Figure 6.1 contains an ASD value corresponding to the SD value in r_{37}.

In the DTD part of the ASD value, special structure "UNKNOWN" is introduced. The DTD means that the internal structure of element type "d" is unknown. Each "d" element in the tagged text part does not contain actual text data. Instead, each "d" element in the ASD value contains a triplet named *marker*, which specifies the location of actual text data stored in the document repository. The marker consists of the identifier of a document in the document repository, the begin position of the designated element in the document, and its end position.

Note that application of Unpack operator $\mathbf{U}_{B \to C(O, D[c \subset b])\ as\ x}$ to relation r_{38} is also possible even if the SD value in r_{37} is replaced by the ASD value.

This is because the operation requires no contents of “d” type elements.

ASD values can be generated at different abstraction levels. The most abstract ASD value has only one UNKNOWN structure (e.g. $\langle$ a:UNKNOWN , “`<a>` [id1, 2, 17] `</a>`” $\rangle$). However, the result of the above unpack operation cannot be derived with this ASD value. In this case, the ASD value does not contain enough information about internal structures to perform the unpack operation, since the contents corresponding to the marker may contain elements of type “b” or “c.”

6.2.2 Abstraction and Materialization Operators

Abstraction operator

Operator $\mathbf{SDA}_{attr,abs}(r)$ (SD Abstraction) transforms SD values in attribute *attr* into ASD values. We call this process *abstraction.* Abstraction specification *abs* determines the abstraction level. The abstraction specification *abs* has the form of $(g_1, \ldots, g_n; g_{n+1}, \ldots, g_m)$. It specifies that elements of types $g_1, \ldots, g_m$ must be contained in the result ASD values, and that in particular, elements of types $g_1, \ldots, g_n$ must not contain any subelement having UNKNOWN element structures. Element types which will appear in region algebra expressions as part of text element specifications of Unpack operators should be included in $g_1, \ldots, g_m$. In particular, element types whose contents are indispensable for evaluating region algebra expressions (e.g. element type e in $\sigma[w](e)$ operator), should be included in $g_1, \ldots, g_n$.

In application of **SDA** operator, the abstraction specification in its parameter determines the DTDs of result ASD values. The procedure is as follows.

Determination of the DTD of an ASD value

Given an abstract specification $(g_1, \ldots, g_n; g_{n+1}, \ldots, g_m)$, the DTD of an ASD value is derived as follows.

Let A, B, and C be sets of generic identifiers defined as

$$\begin{array}{rcl} A & = & \{g_1, \ldots, g_n\}, \\ B & = & \{g_{n+1}, \ldots, g_m\}, \text{ and} \\ C & = & \{g' | \exists g \in A \cup B(\mathit{may\_contain}(g', g))\}, \end{array}$$

where $\mathit{may\_contain}(g', g)$ holds if DTD of the original SD value defines that g' type elements may contain g type elements as descendants. For example, assume that relation r with attribute D contains an SD value with the DTD "a:**rep**(b:**seq**(c:**seq**(d:**text**,e:**text**),f:**or**(g: **rep**(h:**text**),i:**seq**(j:**text**,k:**text**)

)))." In the case of $\mathbf{SDA}_{D,(f;c)}(r)$, A, B, and C are defined as $A = \{f\}$, $B = \{c\}$, and $C = \{a, b\}$, respectively.

Then, the DTD of a result ASD value is determined to contain the element type definitions which satisfies the following conditions: (1) They are included in the DTD of the original SD value. (2) Element types they define are included in

$$C \cup A \cup SE(A),$$

where $SE(A)$ are a set of all element type definitions necessary to define element types in A. In the case of the above example, $SE(A) = \{g, h, i, j, k\}$. The other element types will have UNKNOWN structures. Therefore, the DTD of the result ASD values is "a:**rep**(b:**seq**(c:UNKNOWN, f:**or**(g:**rep**(h:**text**), i:**seq**(j:**text**, k:**text**))))."

Materialization operator

Operator $\mathbf{SDM}_{attr}(r)$ (SD Materialization) transforms ASD values in attribute *attr* into SD values. We call this process *materialization*. For example, application of **SDM** operator to r_{38} results in r_{37}.

6.3 Outline of Query Processing

This section shows the outline of the query processing and optimization scheme. It is assumed that the relational database can execute relational algebra expressions, and that the document repository has text retrieval capability based on the region algebra and can directly execute $\sigma_{\rho(attr,expr)}(r)$. The mediator is assumed to maintain the schema information of the relational database and the relational views of the document repository and Web. In the following discussion, for simplicity, It is also assumed that given queries just once refer to one relation in the document repository, another relation in Web, and yet another relation in the relational database. However, the discussion can be extend to more general cases without difficulty.

Figure 6.2 shows the basic query processing framework. The mediator receives a query expressed in WebNR/SD algebra and transforms it into the form $E(F, G, H)$[3].

The subexpression F has the form $\mathbf{SDA}_{attr,abs}(\sigma_{\rho(attr,expr)}(R))$, where R is a unary relation with attribute *attr* which models the set of structured documents stored in the document repository. Wrapper 1 submits $\sigma_{\rho(attr,expr)}(R)$ to the document repository, receives the result from the document reposi-

[3]Actually, $E(F, G, H)$ includes SD abstraction and materialization operators in addition to WebNR/SD algebra operators.

tory, and translates it into a unary relation consisting of a set of SD values. Then, wrapper 1 executes the **SDA** operator to transform the SD values into abstract SD values according to the abstraction specification *abs*. After that, the wrapper 1 transfers the result relation $Ans1$ to the mediator.

The subexpression G has the form $\mathbf{SDA}_{attr_1,abs}(\mathbf{I}^*_{atrr_1,attr_2,attr_3,attr_4}(\mathbf{N}_{pre,attr}(W)))$[1], where W is a unary relation with an attribute which contains a set of Hlink values referring to Web pages. Unlike the other two types of information sources, Web itself has no capability of query processing. Therefore, it is wrapper 2 that executes the operators in the expression. After that, it transfers the result relation $Ans2$ to the mediator.

The subexpression H consists of only relational algebra operators applied to a relation stored in the relational database. Wrapper 3 is responsible for processing H. The wrapper submits H to the relational database, and transfers the result $Ans3$ to the mediator.

The mediator executes $E(Ans1, Ans2, Ans3)$ to obtain the final result Ans. In the process, it may fetch some SD values from the wrapper 1 and wrapper 2 when it executes **SDM** operators contained in E. Also, it tells the wrapper 2 to execute main parts of Navigate, Import, and Export operators

[1]$\mathbf{I}^*$ is an extended version of Import operator which can import SD values into internal attributes inside relations. It is defined as a composite operator.

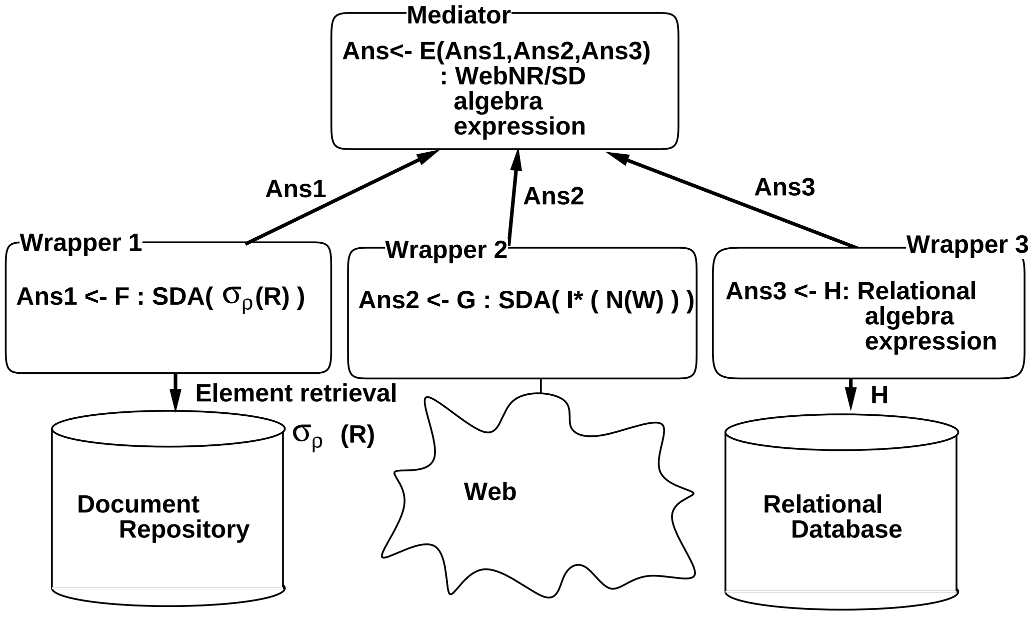


Figure 6.2: Query processing framework

contained in E.

The mediator derives $E(F, G, H)$ in the following two steps.

[Step 1] The mediator decomposes the given query expression into $E'(F', G', H)$, where E', F' and G' are bases for E, F, and G, respectively. As mentioned before, the subexpression H contains only relational algebra operators applied to a relation in the relational database. F' has the form $\sigma_{\rho(attr,expr)}(R)$, where R is a unary relation which models the document repository. The subexpression G' has the form $\mathbf{I}^{*}_{atrr_1,attr_2,attr_3,attr_4}(\mathbf{N}_{pre,attr}(W))$, where W is a unary relation with an attribute which contains a set of Hlink values referring to Web pages. E' consists of WebNR/SD operators applied to data obtained from the three information sources. For example, suppose that we have a binary relation $R_1(A_1, A_2)$ in the relational database, that we view the document repository as a unary relation $R_2(B)$, and that we view the Web as a unary relation $R_3(C)$. Then, the expression $\sigma_{A_2=c}(R_1) \bowtie_{A_1=B_1} \mathbf{U}_{B\to(B_1[b_1],B_2[b_2])}(\sigma_{\rho(B,expr)}(R_2)) \bowtie_{A_1=C_1} \mathbf{U}_{C'\to(C_1[c_1],C_2[c_2])}(\mu_D(\mathbf{I}^{*}_{C',U,L,G}(\mathbf{N}_{C\to C',D}(R_3))))$is decomposed into E', F', G', and H as follows.

$$\begin{array}{rlll}
E': & Ans & := & Ans1 \bowtie_{A_1=B_1} \mathbf{SU}_{B=(B_1[b_1],B_2[b_2]),G}(Ans2) \\
 & & & \qquad\qquad \bowtie_{A_1=C_1} \mathbf{U}_{C'\to(C_1[c_1],C_2[c_2])}(\mu_D(Ans3)) \\
F': & Ans1 & := & \sigma_{\rho(B,expr)}(R_2) \\
G': & Ans2 & := & \mathbf{I}^{*}_{C',U,L,G}(\mathbf{N}_{C\to C',D}(R_3)) \\
H: & Ans3 & := & \sigma_{A_2=c}(R_1).
\end{array}$$

The example here is very simple. Actually, in the decomposition of the query expression into $E'(F', G', H)$, the mediator tries to push locally executable operators down into F', G', and H, and to make E' include as few operators as possible. This process is carried out based on a number of rewriting rules. In addition to well-known algebraic optimization techniques such as selection push-down, they include rules to utilize local filtering capability of the document repository to support nested relational algebra operators, and conversely, those to use the algebraic capability of the relational database to support region-algebra-based data manipulation. Some of the rules are discussed in Section 6.4.

[Step 2] The mediator transforms E', F', and G' into E, F, and G, respectively. First, F and G are derived as $\mathbf{SDA}_{attr_1,abs_1}(F')$ and $\mathbf{SDA}_{attr_2,abs_2}(G')$, respectively. The abstraction specifications abs_1 and abs_2 are determined based on E'. Consider the case of the expressions F' shown in Step 1. E' implies that application of Unpack operator to the result of F' requires only elements of type b_1 and b_2. In particular, E' requires no actual contents of SD values in attribute B_2 (elements of type b_2) until the join operation is finished. Because the values in B_2 are parts of SD values contained in R_2, the abs_1 is determined to be "$(b_1; b_2)$" and F becomes

$$Ans1 := \mathbf{SDA}_{B,(b_1;b_2)}(\sigma_{\rho(B,expr)}(R_2)).$$

The same discussion is applied to G'. In this case, G becomes

$$Ans2 := \mathbf{SDA}_{C',(c_1;c_2)}(\mathbf{I}^*_{C',U,L,G}(\mathbf{N}_{C\to C',D}(R_3))).$$

E is obtained by inserting **SDM** operators into E' at places where materialized SD values are needed. In the above example, **SDM** is only required to obtain the final Ans. Thus, E is constructed as follows.

$$\begin{aligned} Ans \ := \ & \mathbf{SDM}_{B_2}(\mathbf{SDM}_{C_2}(\\ & \qquad Ans1 \bowtie_{A_1=B_1} \mathbf{SU}_{B=(B_1[b_1],B_2[b_2]),G}(Ans2) \\ & \qquad\quad \bowtie_{A_1=C_1} \mathbf{U}_{C'\rightarrow(C_1[c_1],C_2[c_2])}(\mu_D(Ans3)))) \end{aligned}$$

In the above explanation, we have shown the use of ASD values in the simplest way. Actually, it is necessary to tune the abstraction level based on the available work space in the mediator, the local query processing cost, and the data transfer cost.

6.4 Query Processing Example

This section shows query processing steps for the WebNR/SD algebra expressions given in Section 5.6.

[Step 1] As mentioned before, the mediator has a number of rewriting rules. Figure 6.3 shows some rules specific to the WebNR/SD algebra and used in this example. Rules RL1 and RL2 push selection down based on the commutativity of Selection and composite converter $\mathbf{P}_{(A_1,\ldots,A_n)\rightarrow A:SC,gi}(r)$. RL3 and RL4 are rules to screen SD values in advance to discard those which cannot satisfy the outer-most selection condition. These rules are mainly used to

(RL1) $\sigma_{\rho(A,expr)}(\mathbf{P}_{(A_1,\ldots,A_n)\rightarrow A:SC,gi}(r))$

$\quad\quad \rightsquigarrow \mathbf{P}_{(A_1,\ldots,A_n)\rightarrow A:SC,gi}(\sigma_{\rho(A_{i1},expr)\vee\ldots\vee\rho(A_{im},expr)}(r))$

where $expr$ includes no gi, $\mathbf{I}$, $<$ and $>$. $\{A_{i1},\ldots,A_{im}\}$ are a subset of $\{A_1,\ldots A_n\}$ such that the predicate $\rho(A_{ij},expr)$ may hold.

(RL2) $\sigma_{\rho(A,\sigma[w](gi))}(\mathbf{P}_{(A_1,\ldots,A_n)\rightarrow A:SC,gi}(r))$

$\quad\quad \rightsquigarrow \mathbf{P}_{(A_1,\ldots,A_n)\rightarrow A:SC,gi}(\sigma_{\rho(A_1,\sigma[w](\mathbf{I}))\vee\ldots\vee\rho(A_n,\sigma[w](\mathbf{I}))}(r))$

(RL3) $\sigma_{\rho(A_i,expr)}(\mathbf{U}_{A\rightarrow(A_1[e_1]\ as\ x_1,\ldots A_n[e_n]\ as\ x_n)}(r))$

$\quad\quad \rightsquigarrow \sigma_{\rho(A_i,expr)}(\mathbf{U}_{A\rightarrow(A_1[e_1]\ as\ x_1,\ldots A_n[e_n]\ as\ x_n)}(\sigma_{\rho(A,expr)}(r)))$

(RL4) $\sigma_{\rho(A_1,expr)}(\mu_A(\mathbf{U}_{A\rightarrow(O,A_1[e]\ as\ x),G}(r)))$

$\quad\quad \rightsquigarrow \sigma_{\rho(A_1,expr)}(\mu_A(\mathbf{U}_{A\rightarrow(O,A_1[e]\ as\ x)}(\sigma_{\rho(A,expr)}(r))))$

(RL5) $\sigma_{\rho(A,\sigma[w](\mathbf{I}))}(\gamma_{A,SD(gi)}(r)) \rightsquigarrow \gamma_{A,SD(gi)}(\sigma_{A\ni w}(r))$

where attribute A of relation r is of String type.

(RL6) $\sigma_{A=w}(\gamma_{A,String}(r)) \rightsquigarrow \sigma_{A=w}(\gamma_{A,String}(\sigma_{\rho(A,\sigma[w](\mathbf{I}))}(r)))$

where attribute A of relation r is of SD type.

Figure 6.3: Sample rewriting rules

make the document repository sift structured documents before data transfer to the mediator. RL5 pushes Selection down through Domain translator. Note that, by this rule, the selection condition becomes a word containment predicate "$\ni$" on String type values. RL6 facilitates further application of other rewriting rules related to $\sigma_{\rho(attr,expr)}$.

The result of Step 1 is shown below. Note that the selection on attributes "Title" and "Affiliation" in the original expression to yield r_{30} is pushed down into F as $\sigma_{\rho(Doc,\sigma[''Web''](\mathbf{I})\cap\sigma[''T\text{-}Univ''](\mathbf{I}))}(DR)$ [2] in order to discard documents

[2] $\rho(Doc,\sigma[''\mathrm{Web}''](\mathbf{I}) \cap \sigma[''T\text{-}Univ''](\mathbf{I}))$ is equivalent to $\rho(Doc,\sigma[''\mathrm{Web}''](\mathbf{I})) \wedge \rho(Doc,\sigma[''T\text{-}Univ''](\mathbf{I}))$.

not including the words "Web" and "T-Univ," which reduces data transfer from the document repository to the mediator. In addition, H shows that the relational database in advance performs project out unnecessary attributes to reduce data transfer.

F':

$$Ans1 \;:=\; \sigma_{\rho(Doc, \sigma[''Web''](\mathbf{I}) \cap \sigma[''T\text{-}Univ''](\mathbf{I}))}(DR)$$

G':

$$Ans2 \;:=\; \mathbf{I}^{*}_{D,U,L,G}(\mathbf{N}_{Page(\rightarrow .)* \rightarrow D[\sigma['publication\text{-}list'](page\text{-}title)],C}(WWW)))$$

H:

$$Ans3 \;:=\; \pi_{Name,Addr,Tel,E\text{-}Mail}(Faculty)$$

E' consists of the following expressions:

$$z_1 \;:=\; \sigma_{Title \ni ''Web'' \wedge Affiliation = ''T-Univ''}($$
$$\mathbf{U}_{Author \rightarrow (A\text{-}Name, Affiliation)}(\mu_{As}(\mathbf{U}_{Authors \rightarrow As(O_3, Author)}($$

$$
\begin{aligned}
&\qquad\qquad \mathbf{U}_{Doc \to (Authors, Title, Pub\text{-}Info, Abstract)}(Ans1)))))\\
z_2 &:= \mu_{As}(\mathbf{U}_{Authors \to As(O_2, Author)}(\\
&\qquad \mathbf{U}_{P\text{-}Item \to (Authors, Title, Pub\text{-}Info, Hp)}(\\
&\qquad\qquad \mu_P(\mathbf{U}_{D \to P(O_1, P\text{-}Item)}(\mu_C(Ans2))))))\\
z_3 &:= \pi_{Name, Addr, Tel, E\text{-}Mail, Author, Title, Pub\text{-}Info, Hp, Abstract}(\\
&\qquad \sigma_{A\text{-}Name = Author}(Ans1 \bowtie Ans2) \bowtie_{Author = Name} Ans3)\\
z_4 &:= \mathbf{P}_{P(New\text{-}P\text{-}Item) \to Pubs:RC}(\nu_{P=(New\text{-}P\text{-}Item)}(\\
&\qquad \mathbf{P}_{(Title, Pub\text{-}Info, Hp, Abstract) \to New\text{-}P\text{-}Item:SC}(z_3)))\\
z_5 &:= \mathbf{P}_{(Author, Pubs) \to P\text{-}List:SC}(z_4)\\
z_6 &\to \mathbf{E}_{P\text{-}List, U_1, 'publications', 'a'}(\mathbf{URL}_{U_1}(z_5))\\
z_7 &:= \mathbf{P}_{Members(Member) \to Table:RC}(\\
&\quad \nu_{Members=(Member)}(\\
&\qquad \mathbf{P}_{(Name, Addr, Tel, E\text{-}Mail, P\text{-}List) \to Member:SC}(z_6)))\\
Ans &:= \mathbf{E}_{Table, U_2, 'index\text{-}page', 'a'}(\mathbf{URL}_{U_2}(z_7))
\end{aligned}
$$

[Step 2] E' implies that the text elements of element type "title," "authors," "author," "a-name," "affiliation," "pub-info," and "abstract" in the papers stored in the document repository are necessary. It also implies that the text elements of element type "title," "authors," "author," "pub-info," and "hp" in the publication list Web pages are necessary. Moreover, the contents of "title," "a-name," "affiliation," and "pub-info" type elements in the papers, and that of "title," "author," and "pub-info" type elements in the Web pages are indispensable because they are used as join or selection attributes. No other text elements are required for the query processing.

Thus, abstraction specifications in F and G are determined to be (Title, Author, Pub-Info; Authors, Abstract) and (Title, Author, Pub-Info; Authors, Hp), respectively. The result expressions are shown below.

F:

$$\begin{aligned} Ans1 \;:=\; & \mathbf{SDA}_{Doc,(Title,A\text{-}Name,Affiliation,Pub\text{-}Info;Authors,Author,Abstract)}(\\ & \qquad \sigma_{\rho(Doc,\sigma[''Web''](\mathbf{I})\cap\sigma[''T\text{-}Univ''](\mathbf{I}))}(DR)) \end{aligned}$$

G:

$$\begin{aligned} Ans2 \;:=\; & \mathbf{SDA}_{Doc,(Title,Author,Pub\text{-}Info;Authors,Hp)}(\\ & \quad \mathbf{I}^{*}_{D,U,L,G}(\mathbf{N}_{Page(\rightarrow .)*\rightarrow D[\sigma['publication\text{-}list'](page\text{-}title)],C}(\\ & \qquad WWW))) \end{aligned}$$

Finally, **SDM** operators are inserted into E' to obtain the final query result, and we get the following E.

E:

$$z_1 \;:=\; \sigma_{Title\ni''Web''\wedge Affiliation=''T-Univ''}($$

$$\mathbf{U}_{Author \to (A\text{-}Name, Affiliation)}(\mu_{As}(\mathbf{U}_{Authors \to As(O_3, Author)}(\mathbf{U}_{Doc \to (Authors, Title, Pub\text{-}Info, Abstract)}(Ans1)))))$$

$$\begin{aligned}
z_2 &:= \mu_{As}(\mathbf{U}_{Authors \to As(O_2, Author)}(\\
&\qquad \mathbf{U}_{P\text{-}Item \to (Authors, Title, Pub\text{-}Info, Hp)}(\\
&\qquad\qquad \mu_P(\mathbf{U}_{D \to P(O_1, P\text{-}Item)}(\mu_C(Ans2))))))\\
z_3 &:= \pi_{Name, Addr, Tel, E\text{-}Mail, Author, Title, Pub\text{-}Info, Hp, Abstract}(\\
&\qquad \sigma_{A\text{-}Name=Author}(Ans1 \bowtie Ans2) \bowtie_{Author=Name} Ans3)\\
z_4 &:= \mathbf{P}_{P(New\text{-}P\text{-}Item) \to Pubs:RC}(\nu_{P=(New\text{-}P\text{-}Item)}(\\
&\qquad \mathbf{P}_{(Title, Pub\text{-}Info, Hp, Abstract) \to New\text{-}P\text{-}Item:SC}(z_3)))\\
z_5 &:= \mathbf{P}_{(Author, Pubs) \to P\text{-}List:SC}(z_4)\\
z_6 &\to \mathbf{E}_{P\text{-}List, U_1, 'publications', 'a'}(\mathbf{SDM}_{P\text{-}List}(\mathbf{URL}_{U_1}(z_5)))\\
z_7 &:= \mathbf{P}_{Members(Member) \to Table:RC}(\\
&\qquad \nu_{Members=(Member)}(\\
&\qquad \mathbf{P}_{(Name, Addr, Tel, E\text{-}Mail, P\text{-}List) \to Member:SC}(z_6)))\\
Ans &:= \mathbf{E}_{Table, U_2, 'index\text{-}page', 'a'}(\mathbf{SDM}_{Table}(\mathbf{URL}_{U_2}(z_7)))
\end{aligned}$$

Chapter 7

Visual User Interface

7.1 Overview

This chapter explains a visual user interface for the information integration system for structured documents, Web, and databases. In case of traditional database utilization, usually, we first browse metadata such as the schema information of a database, and then submit query statements. But this simple procedure causes problems in the context of heterogeneous information integration. The user interface here provides visual and interactive supporting tools in order to overcome the problems.

7.2 Features of the Visual User Interface

In case of traditional database utilization, usually, we first browse meta-data such as the schema information of a database, and then submit query statements. This simple procedure causes problems in the context of heterogeneous information integration, because of the following reasons:

1. It is difficult to identify target data objects from a sea of data objects in information sources. The reasons are as follows: First, the information sources include structured documents and Web. Their data structures are more complicated compared to those of relational databases. Second, in general, there is no way to bundle data objects of the same type like the extension in ODBs. Moreover, target data objects may be collected from different information sources. Without identifying target data objects, it is impossible to manipulate data objects.

2. Even if target objects of operation have been identified, the complexity of target data structure makes query construction more difficult for end users. For example, the example scenario shown in Section 2.3 requires the hypertext view to be obtained by restructuring of structured documents, Web, and relational databases.

The visual user interface is designed to overcome the problems. The user interface models integration activity as combinations of *target discovery* and *data operation*, and provides them with visual and interactive supporting tools. Target discovery is defined as the process of discovering and extracting target data objects from a number of information sources. In this process, only data objects relevant to user requests are identified. Data operation is defined as the process of manipulating the identified target data objects to construct the final answer for user requests.

Target discovery and data operations in the example scenario in Section 2.3 are as follows.

Target Discovery A set of papers, a set of original publication list Web pages, and a set of data objects containing information about faculties, are necessary to construct the result hypertext view. Therefore, in the target discovery, they must be identified and extracted from a sea of data objects in the information sources having heterogeneous and complicated data structures.

Data Operation The result hypertext view is obtained by joining and restructuring of the sets of data objects. Construction of new publication lists for authors and the index page is performed there.

Because the two processes are essentially different in their purposes, the visual user interface provides different supporting tools for them. Target discovery is supported by interactive information exploration facility which combines browsing and querying of both metadata and data. The reason for introducing metadata query is that browsing alone is insufficient to identify target data objects in accordance with large and complicated metadata in a variety of information sources. The reason for introducing browsing and querying instances into target discovery is that structured documents written in SGML and XML may have different instance-level data structures even if their metadata (DTDs) are the same, since SGML and XML allow their documents to have variant, optional, and exception structures.

Data operation is supported by a visual data manipulation language named *HQBE* (Query By Example for Heterogeneous information sources). HQBE is a QBE [Zlo77] style language, and realizes declarative specification of a subset of WebNR/SD algebra expressions. Its features are as follows. (1) HQBE can manipulate various and complicated data structures appearing in structured documents and Web pages as nesting structures and hypertext link structures. (2) HQBE allows target data objects to have heterogeneous data structures. That is, all the data objects do not need to have the same structures as long as they have some common structures essential for manipulation. (3) The manipulation result may have various data representations such as structured documents, Web pages, and relations, to suit purposes.

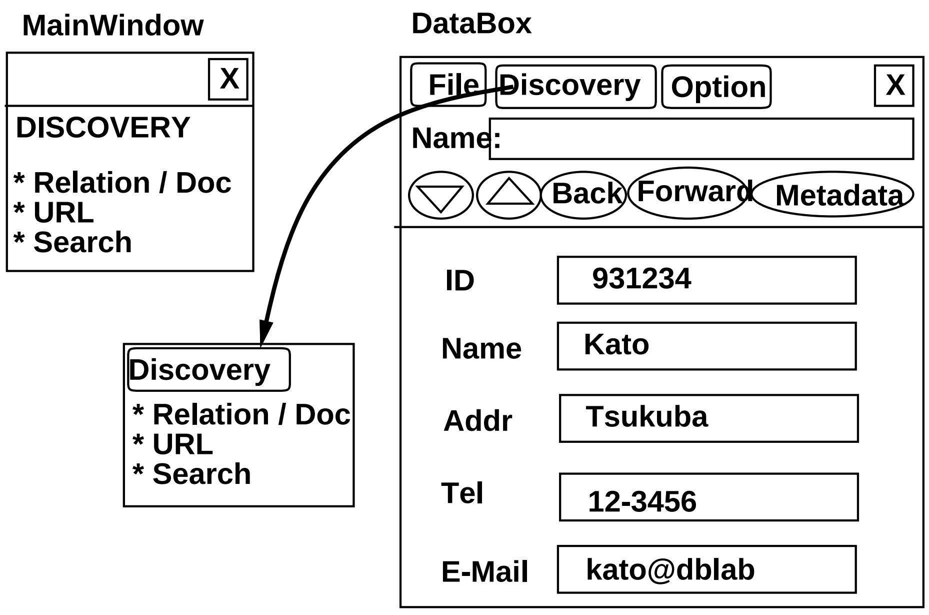


Figure 7.1: Windows in the visual user interface

7.3 Components of the Visual User Interface

Main components of the user interface are a *main window*, which gives a means of the target discovery, and *data boxes*, each of which shows a set of data objects identified and extracted in the target discovery process (Figure 7.1).

The main window provides the following means of target discovery.

- The user selects from the menu the name of an information source and the name of a relation or a document collection. Then, a data box is created to show the set of tuples or documents in the relation or document collection (Relation/Doc).
- The user specifies an URL. Then, a data box containing the Web page is created (URL).
- The user specifies query conditions on metadata and data. Then, a data box is created whose data objects satisfy the conditions (Search).

The triangle and inverted triangle buttons of a data box are used to browse the set of data objects in the data box. The "Discovery" menu of a data box provides the same means of target discovery as the main window does. However, their functions differ from those of the main window in that they replace the data objects previously stored in a data box with new ones, instead of creating a new data box. "Back" and "Forward" buttons can be used to walk back and forth through the sets of data objects the data box has shown before.

The "Search" function of data boxes is extended in that it allows the data objects currently stored in a data box to be used for target discovery. The detailed explanation is given in the next section.

7.4 Interactive Information Exploration

As a means of the target discovery, the user interface provides interactive information exploration facility which combines browsing and querying of both metadata and data. The example in Section 2.3 requires three data boxes which correspond to a set of papers, a set of original publication list Web pages, and a set of data objects containing faculty information.

In the visual user interface, target discovery is the process of creating data boxes which correspond to target data objects. This section illustrates the facility by showing the process of creating data boxes which correspond to the above three sets of data objects. Browsing of both metadata and data is illustrated by creation of data boxes for faculty information and papers. Querying of metadata and data is illustrated by creation of the data box for publication list Web pages.

7.4.1 Creation of Data Box for Faculty Information

When a user selects the "Relation/Doc" menu item in the main window, information sources and their contents (such as the relation names) are displayed. He selects a relation which he expects to be the faculty relation.

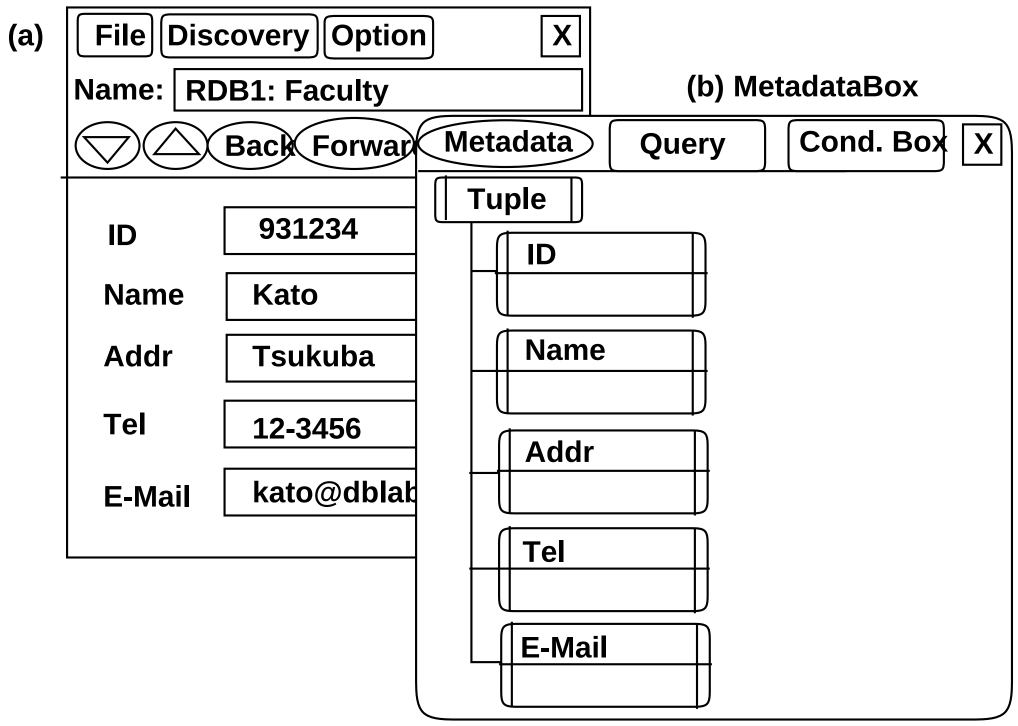


Figure 7.2: Data box (D1) and metadata box for faculty information

Then, a data box corresponding to the relation is created (Figure 7.2 (a)). The "Metadata" button in the data box is used to create a metadata box (Figure 7.2 (b)) which displays the metadata (schema information) of the relation. The two windows give the user the means of browsing of both metadata and data. If he found the relation wrong one, he would select again the "Relation/Doc" menu item in the main window, and continue to browse data and metadata until he find the right one. In the following discussion, the data box created here is referred to as D1.

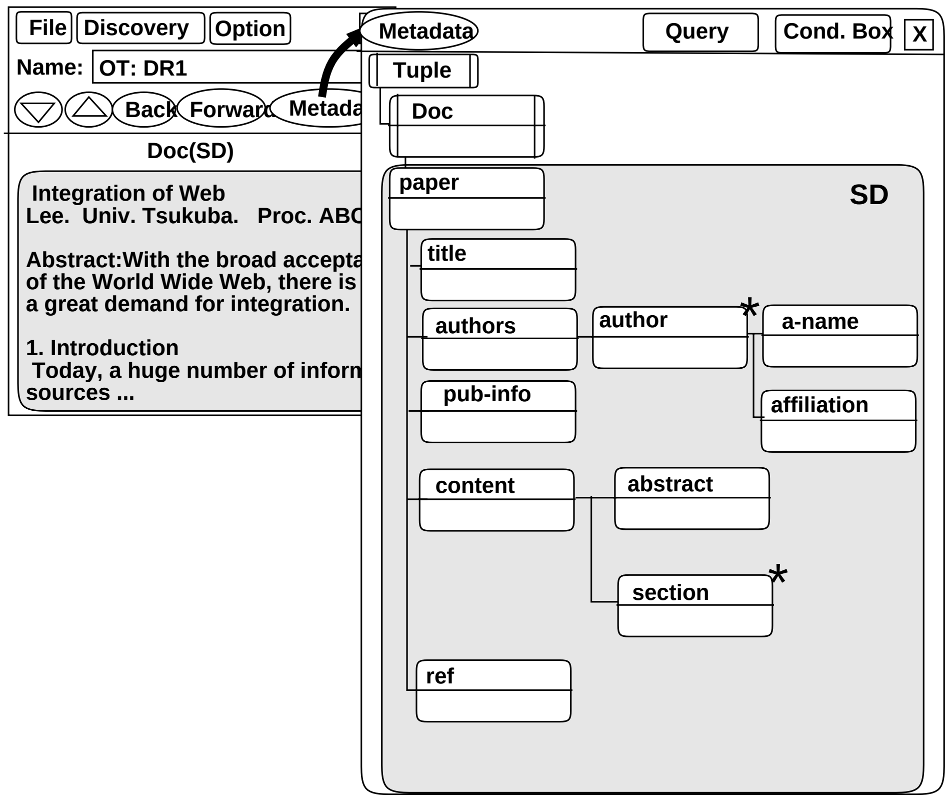
7.4.2 Creation of Data Box for Papers

Creation of the data box for papers is achieved in the same way as in the case of the faculty information. The differences are that each data object in the data box is not a tuple but a document, and that the data structure in the metadata box contains DTD of the document (Figure 7.3). In the following discussion, the data box is referred to as D2.

7.4.3 Creation of Data Box for Publication List Web Pages

The user clicks the "URL" menu item in the main window, and specifies the URL of the CS department home page. Then, a data box containing the Web page is created. In the following discussion, it is referred to as D3. He can traverse hypertext links in the Web in the same way as he uses an Web browser, in order to find publication list Web pages. Figure 7.4 shows a publication list Web page found in the process. The metadata box shows the document structure of this page.

In actual applications, the user is often encountered by structural variation or anomalies of data structure. In this example, if he traverses and examines the original publication list Web pages, he will find the following

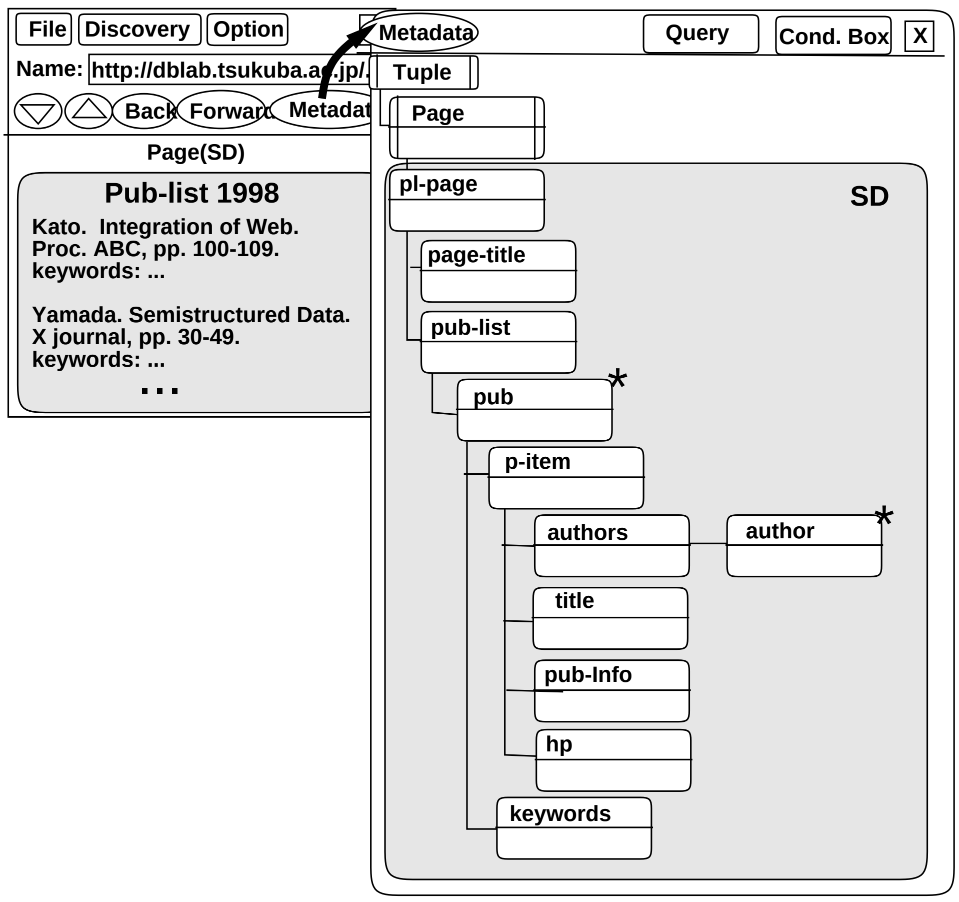


Integration of Web
Lee. Univ. Tsukuba. Proc. ABC

Abstract:With the broad accepta
of the World Wide Web, there is
a great demand for integration.

1. Introduction
Today, a huge number of inform
sources ...

Figure 7.3: Data box (D2) and metadata box for papers



Pub-list 1998

Kato. Integration of Web.
Proc. ABC, pp. 100-109.
keywords: ...

Yamada. Semistructured Data.
X journal, pp. 30-49.
keywords: ...

...

Figure 7.4: Browsing publication list Web pages for years (D3)

two facts (see Section 2.3):

1. PL pages for different years have different inner page structures. For example, the PL page for 1996 categorizes publication items by projects, while that for 1998 annotates each publication item with some key words.
2. However, every page has a “page-title” element containing the word “pub-list.” In addition, even though the pages have minor differences in their structures, they can be considered to have the similar structure in that each page contains a collection of publication items with “p-item” tags.

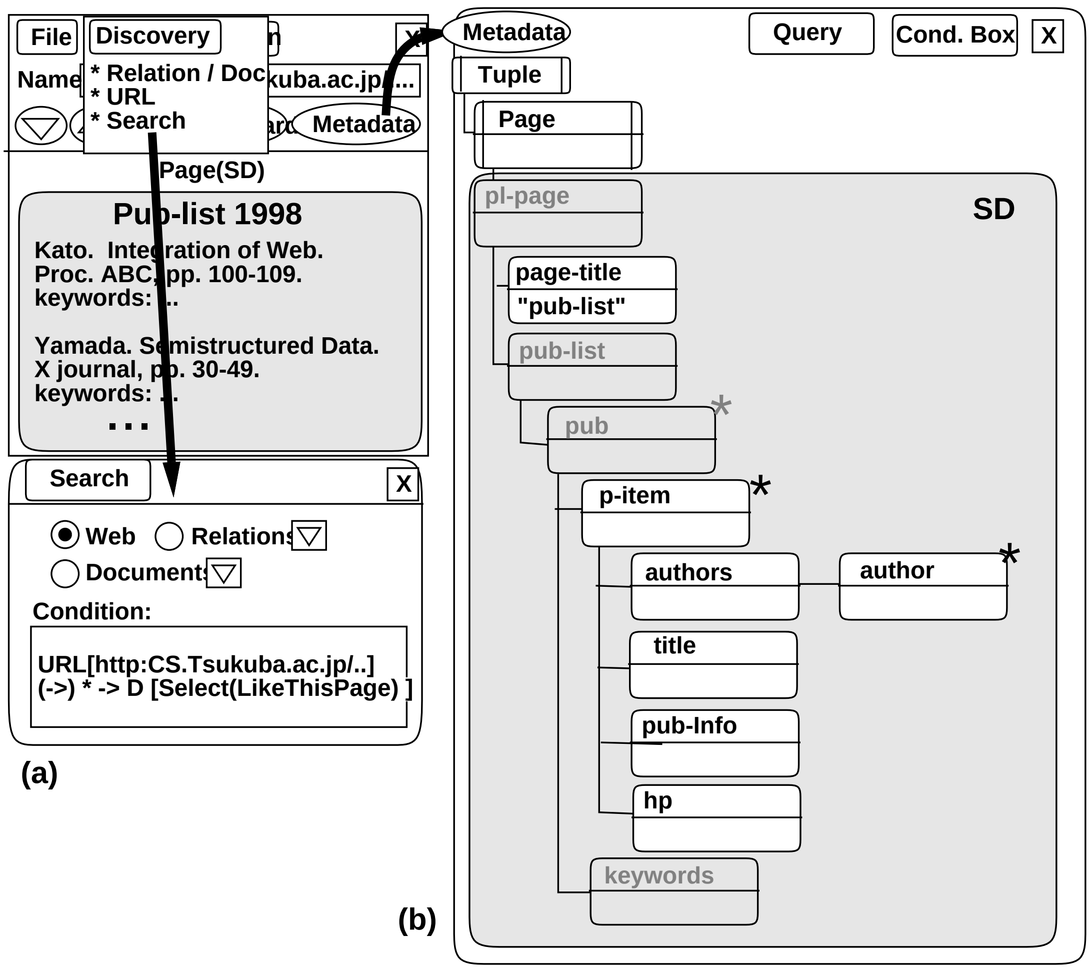
In spite of existence of such variations in publication list Web pages, the user can fetch all the pages by using the common properties of them.

First, the user selects “Search” item in “Discovery” menu of the data box D3. Then, a window for search condition appears (Figure 7.5(a)). He specifies query conditions to get all the publication Web pages. In Figure 7.5(a), he specifies that he wants such Web pages that they reside in the same Web server as that of the CS department home page, that they are accessible from the home page, and that they satisfy the condition specified in the metadata box.

Then, he edits the data structure of the original publication list Web page already shown in the metadata box, in order to construct the condition (Figure 7.5 (b)). The condition says that the page should contain the word "pub-list" in the "page-title" element, and have a collection of publication items. As shown in the figure, the condition about the metadata is specified by erasing irrelevant data structures in the metadata box.

Finally, he clicks "Search" button of the window (a) (in Figure 7.5). Then, all the qualified Web pages are stored in the data box D3. As shown in this example, target discovery can be achieved by querying both metadata and data. Note that the data objects whose structures are almost the same but different in part can be identified and extracted.

This section has shown how to construct data boxes with interactive information exploration combining browsing and querying of both metadata and data. However, they are only examples of data box construction. In fact, browsing and querying can be combined and intertwined in any ways to construct data boxes.



Pub-list 1998

Kato. Integration of Web.
Proc. ABC, pp. 100-109.
keywords: ...

Yamada. Semistructured Data.
X journal, pp. 30-49.
keywords: ...
...

URL[http:CS.Tsukuba.ac.jp/..]
(->) * -> D [Select(LikeThisPage)]

Figure 7.5: Querying metadata and data to relate D3 with all the publication Web pages

7.5 Visual Data Manipulation Language: HQBE

The user interface provides a visual data manipulation language named *HQBE* (Query By Example for Heterogeneous information sources) as a means of data operation. Figures 7.6 and 7.7 comprise the HQBE description which specifies the data operation for the example scenario in Section 2.3. Assume that the data boxes D1, D2, and D3 already exist because the target discovery has been done. The query is constructed in the following way.

First, the user makes entries in the metadata boxes of D1, D2, and D3 (Figure 7.6(a)(b)(c)). Words starting with "&" (such as "&Tsukuba" in Figure 7.6) are *example elements*, which were introduced by QBE [Zlo77]. Conditions such as " = 'T-Univ' " can be specified in the description. Moreover, condition boxes can be used to express conditions difficult to express in metadata boxes ((d) in Figure 7.6).

Next, the user opens the fourth data box D4 (Figure 7.7). He tells the user interface that the data box should be a empty data box, which has no corresponding set of data objects. Therefore, the metadata box of D4 shows a blank space at first. He draws the data structure of the required result there and enters example elements (Figure 7.7(e)). In Figure 7.7, the description of required new publication lists and that of the required index pages are (g) and (f), respectively. Hypertext links among them are represented as dotted arrows.

Finally, he clicks the “Query” button in the metadata box of D4. The answer appears in D4. In this example, a hypertext link to the index page is stored in D4 so that he can browse all the result Web pages including the index page and new publication lists.

HQBE descriptions are translated into WebNR/SD algebra expressions to compute data manipulation results. In addition to simple manipulations such as selection and projection against a set of data objects in a data box, HQBE is able to express complex restructuring manipulations such as the above example. Also, HQBE can manipulate uniformly the data objects which partially differ in their structures, but have some common structures essential for the manipulation. In the example, structural difference of the publication list Web pages can be managed without difficulty. Moreover, manipulation results can be in the form of structured documents, Web, and relations, to suit purposes. In the example, the result is obtained in the form of Web pages. Namely, the point is that HQBE allows users to express, in a declarative way, data manipulations that take advantage of WebNR/SD features.

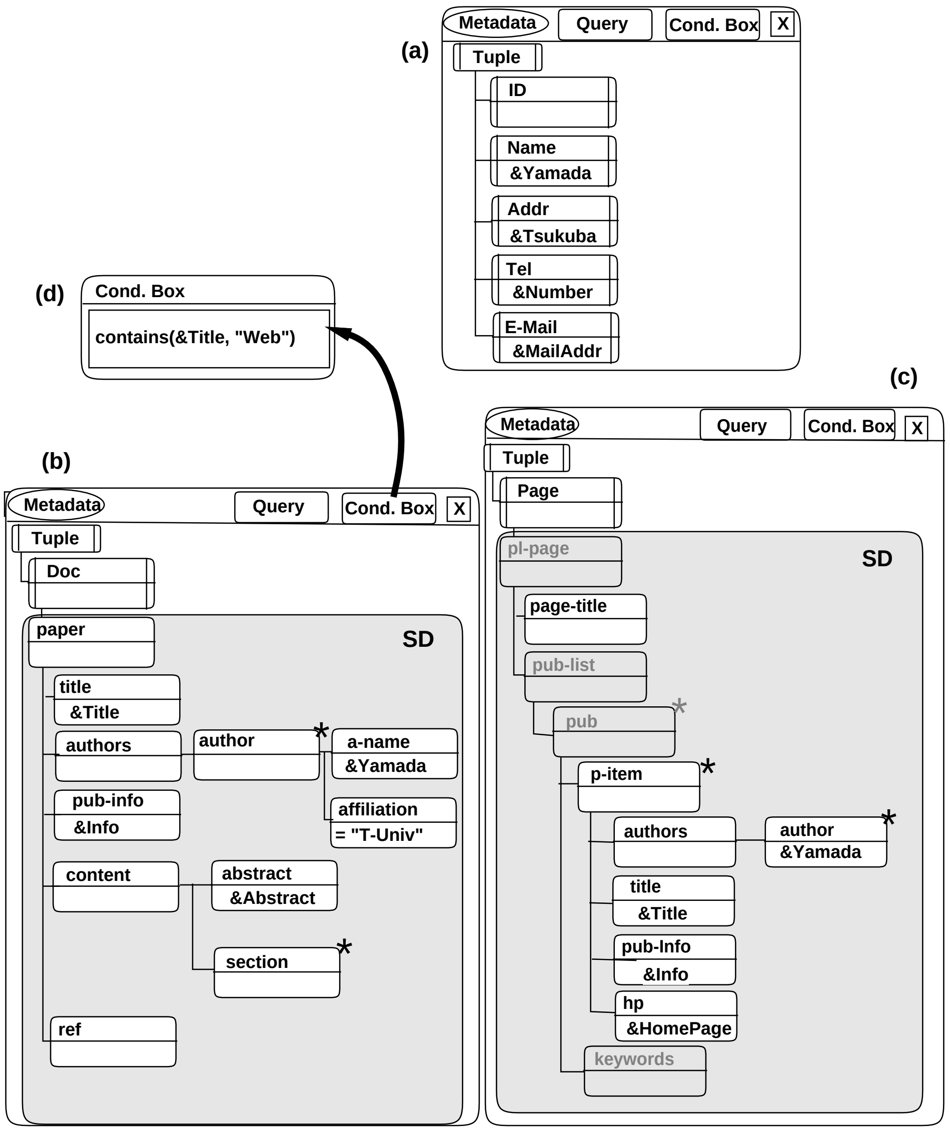


Figure 7.6: HQBE description for data operation (1)

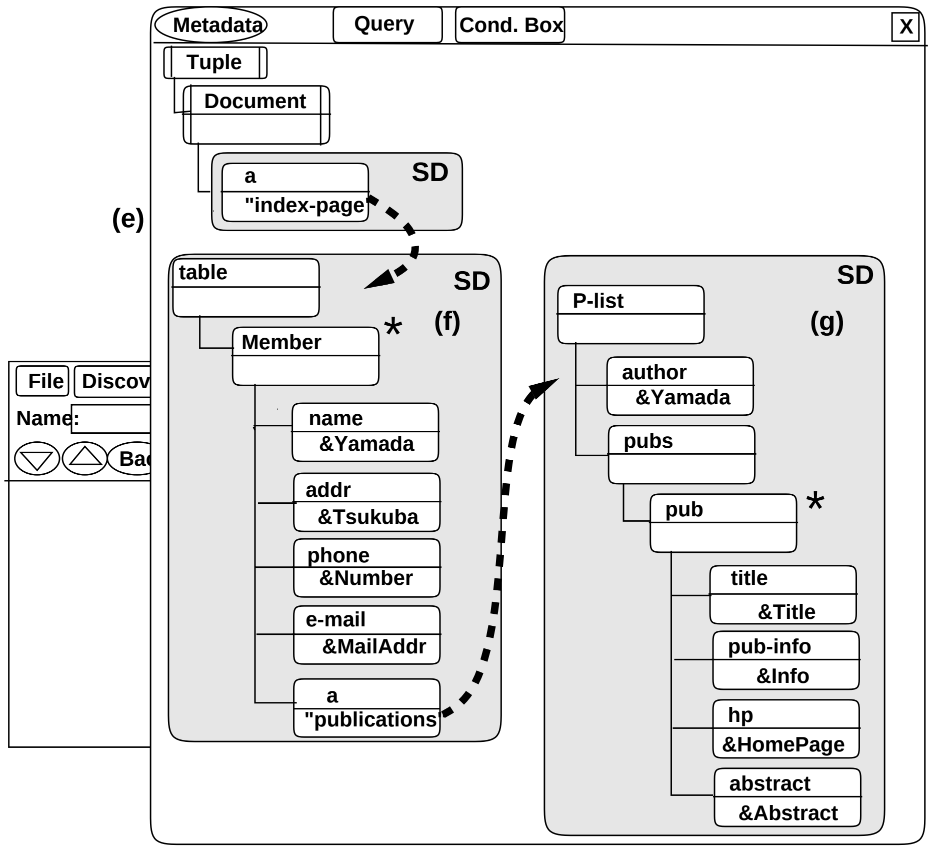


Figure 7.7: HQBE description for data operation (2)

Because data manipulation results are stored in data boxes again, they can be used for further target discovery by the interactive information exploration faculty, and/or for further data manipulation by HQBE.

7.6 Translation of HQBE Descriptions into WebNR/SD Algebra Expressions

This section explains translation of HQBE descriptions into WebNR/SD algebra expressions. The HQBE description in Section 7.5 is used as an example. In this section, the metadata boxes associated with sets of target data objects (the metadata boxes of D1, D2, and D3 in the example) are referred to as *target metadata boxes.* The metadata box which represents the data structure of the manipulation result (one of D4) is called the *output metadata box.*

In the following discussion, we do not deal with the pictorial HQBE description in Section 7.5 directly. We express the HQBE description in a text format. In this case, description in each target metadata box is expressed by *a target expression*, and description in the output metadata box is expressed by an *output expression.* In general, an HQBE description can be expressed by a number of target expressions, an output expression, and conditions specified in condition boxes.

```
NULL:R(
    ID:Integer,
    Name:String[&Yamada],
    Addr:String[&Tsukuba],
    Tel:String[&Number],
    E-Mail:String[&MailAddr]
)
```

Figure 7.8: Target expression for D1

7.6.1 HQBE Description in a Text Format and Its Semantics

Figures 7.8, 7.9, and 7.10 show target expressions which represent the target metadata boxes in Figure 7.6.

Target expressions contain two kinds of information: (1) Data structures of target data objects, and (2) Example elements and conditions specified in target metadata boxes. An important point is that document (SD value) structures are described more loosely in target expressions than they are in DTDs. The descriptions of SD value structures is referred as *loose structure specifications*. Target documents of HQBE manipulation can have different document structures and DTDs as long as they conform to the same loose structure specifications. Figure 7.11 shows the syntax and semantics of target

```
NULL:R(
    Doc:SD(
        {
            $\supset_d$ $title.1[\&Title]$,
            $\supset_d$ $authors.2\{\supset_d$ $author^*$
                                {
                                    $\supset_d$ $a\text{-}name.1[\&Yamada]$,
                                    $\supset_d$ $affiliation.2[=$ "T-Univ"]
                                },
            $\supset_d$ $pub\text{-}info.3[\&Info]$,
            $\supset_d$ $content.4\{\supset_d$ $abstract.1[\&Abstract]\}$
        }
    )
)
```

Figure 7.9: Target expression for D2

```
NULL:R(
    Page:SD(
        {
            $\supset$ $p\text{-}item^*$
                {
                    $\supset_d$ $authors.1\{\supset_d author^*[\&Yamada]\}$,
                    $\supset_d$ $title.2[\&Title]$,
                    $\supset_d$ $pub\text{-}info.3[\&Info]$,
                    $\supset_d$ $hp.4[\&HomePage]$
                }
        }
    )
)
```

Figure 7.10: Target expression for D3

expressions (including loose structure specifications).

According to the semantics explained in Figure 7.11, The loose structure specification contained in the target expression for D3 is interpreted as follows:

“An SD value in attribute Page contains a number of ’p-item’ elements. Each ’p-item’ element has an ’authors’ element, a ’title’ element, a ’pub-info’ element, and a ’hp’ element as the first, second, third, and fourth child elements. The ’authors’ element has ’author’ elements. The ’author,’ ’title,’ ’pub-info,’ and ’hp’ elements are associated with example elements ’&Yamada,’ ’&Title,’

```
TE::= 'NULL:R(' Attr ':' SubSt { ',' Attr ':' SubSt } ')'
                           //The outermost relational structure
SubSt::= 'R' '(' Attr ':' SubSt { ',' Attr ':' SubSt } ')'
                        //Sub relational structure
       | 'SD''('  LooseSt ')'
                        //LooseStructure for SD values
       | OtherType [ '[' Example | Condition ']' ]
                        //An example element or a condition
                          can follow the other types.

LooseSt::= '{' ContainmentSpec { ',' ContainmentSpec } '}'
          // contains the elements specified by ContainmentSpecs
ContainmentSpec::= SpecElem LooseSt
                      // contains an element specified by SpecElem,
                      // whose inner structure is specified by LooseSt.
                 | SpecElem '[' Example | Condition ']'
                      // contains an element specified by SpecElem,
                      // which is associated with an example element
                      // or a condition.

SpecElem::= '⊃' gi //includes a gi-typed element.
       | '⊃_d' gi '.'  Int //includes a gi-typed element
                     //as the i-th child element.
       | '⊃' gi '*' //includes gi-typed elements.
       | '⊃_d' gi '*' //includes gi-typed elements
                    //as children.
```

Figure 7.11: Syntax and semantics of loose structure specifications

```
R_1(Document:SD(
   a:hlink[``indexpage'']( 
     table:rep_1(
         member:seq(
            name[&Yamada],
            addr[&Tsukuba],
            tel[&Number],
            e-mail[&MailAddr],
            a:hlink[``publications''](
               p-list:seq(
                  author[&Yamada],
                  pubs:rep_1(
                     new-p-item:seq(
                         title[&Title],
                         pub-info[&Info],
                         hp[&HomePage],
                         abstract[&Abstract]
                     )
                  )
               )
            )
         )
     )
   )
))
```

Figure 7.12: Output expression corresponding to D4

'&Info,' and '&HomePage,' respectively."

Figure 7.12 shows the output expression for HQBE description in Section 7.5. An output expression specifies data structure of the manipulation result. It differs from target expressions in the following two points: (1) Document (SD value) structures are specified by their DTDs, not by loose structure specifications. The reason is that construction of the final result requires rigid description of data structures of the result. (2) Nesting structures such as subrelation structures (represented as 'R') and repetition structures of elements in SD values ('rep') require natural numbers to be associated with them. The numbers are associated with them in such a way that the child nesting structures which has the same parent are enumerated by natural numbers. The numbers specify the order of Nest operations. Namely, in the process of data manipulation, Nest operators are applied in the specified order for construction of nesting structures which appear in the output expression. The reason why the numbers are necessary is that difference in the order of Nest operations can result in different manipulation results, since Nest operator has no commutativity.

The condition specified in the condition box of the example HQBE description is as follows:

$$Contains(\&Title, \text{“Web”}).$$

7.6.2 WebNR/SD Algebra Expressions Obtained by the Translation

This subsection explains the WebNR/SD algebra expressions obtained by the translation of HQBE descriptions. In the WebNR/SD algebra expressions, a set of data objects in each data box is regarded as a relation of WebNR/SD. In the example in Section 7.5, relations D_1, D_2, and D_3 are used to represent data objects in <u>D1</u>, <u>D2</u>, and <u>D3</u>, respectively. The expressions include the following three main parts:

Extraction Part This part extracts relevant data from nested relations and SD values. Namely, it transforms each relation associated with its target expression into a flat relation. Attribute values of the result relation correspond to those attribute values and text elements which are associated with example elements and conditions in the target expression. The extraction part consists of expressions each of which extracts data from a target relation. In this example, it consists of three expressions for D_1, D_2, and D_3. Here, we call the expressions EP_1, EP_2, and EP_3.

Selection and Join Part First, the part selects tuples from each relation according to the conditions that the corresponding target expression and condition box specify. Then, it constructs a new relation by joining the results of the selection operations together.

Restructuring Part This part constructs the required data structures by restructuring the result of the Selection and Join part according to an output expression.

Not all the above parts are necessary for HQBE descriptions. For example, an HQBE description which selects tuples satisfying some conditions, is translated into a simple expression which applies only a selection operator to a relation. In this case, the expression is considered to have only the selection and join part containing only a selection operator.

The followings are WebNR/SD algebra expressions for the HQBE description in Section 7.5.

Extraction Part

EP_1:

$$\begin{aligned} r_{37} := \pi_{\&Yamada_1,\&Tsukuba_1,\&Number_1,\&MailAddr_1}(\\ \delta_{Addr\rightarrow\&Yamada_1}(\delta_{Name\rightarrow\&Tsukuba_1}(\delta_{Tel\rightarrow\&Number_1}(\\ \delta_{E\text{-}Mail\rightarrow\&MailAddr_1}(D_1))))) \end{aligned}$$

r_{37}:

&Yamada$_1$	&Tsukuba$_1$	&Number$_1$	&MailAddr$_1$

EP_2:

$$\begin{aligned}
r_{38} := \pi_{\&Yamada_3,\&Title_2,\&Info_2,\&Abstract_1,Affiliation}(\\
\mathbf{U}_{D\rightarrow(\&Yamada_3[(a\text{-}name\cap 1)\subset_d \mathbf{I}])}(\\
\mathbf{U}_{D\rightarrow(Affiliation[(affiliation\cap 2)\subset_d \mathbf{I}])}(\\
\mathbf{U}_{B\rightarrow(\&Abstract_1[(abstract\cap 1)\subset_d \mathbf{I}])}(\\
\mu_C(\mathbf{U}_{A\rightarrow C(O_1,D[author\subset_d \mathbf{I}])}(\\
\mathbf{U}_{Doc\rightarrow(\&Title_2[(title\cap 1)\subset_d \mathbf{I}])}(\\
\mathbf{U}_{Doc\rightarrow(A[(authors\cap 2)\subset_d \mathbf{I}])}(\\
\mathbf{U}_{Doc\rightarrow(\&Info_2[(pub\text{-}info\cap 3)\subset_d \mathbf{I}])}(\\
\mathbf{U}_{Doc\rightarrow(B[(content\cap 4)\subset_d \mathbf{I}])}(D_2)))))))))))
\end{aligned}$$

r_{38}:

&Yamada$_3$	&Title$_2$	&Info$_2$	&Abstract$_1$	Affiliation

EP_3:

$$\begin{aligned}
r_{39} := \pi_{\&Yamada_2,\&Title_1,\&Info_1,\&HomePage_1}(\\
\mu_D(\mathbf{U}_{C\rightarrow D(O_2,\&Yamada_2[author\subset_d \mathbf{I}])}(\\
\mathbf{U}_{A\rightarrow(C[(authors\cap 1)\subset_d \mathbf{I}])}(\\
\mathbf{U}_{A\rightarrow(\&Title_1[(title\cap 2)\subset_d \mathbf{I}])}(\\
\mathbf{U}_{A\rightarrow(\&Info_1[(pub\text{-}info\cap 3)\subset_d \mathbf{I}])}(\\
\mathbf{U}_{A\rightarrow(\&HomePage_1[(hp\cap 4)\subset_d \mathbf{I}])}(\\
\mu_B(\mathbf{U}_{Page\rightarrow B(O_1,A[p\text{-}item\subset \mathbf{I}])}(D_3))))))))))
\end{aligned}$$

r_{39}:

&Yamada$_2$	&Title$_1$	&Info$_1$	&HomePage$_1$

Selection and Join Part

$$
\begin{aligned}
r_{40} := r_{37} &\bowtie_{\&Yamada_1=\&Yamada_2} \gamma_{\&Yamada_2 \rightarrow String}(r_{39}) \\
&\bowtie_{\&Yamada_2=\&Yamada_3 \wedge \&Title_1=\&Title_2 \wedge \&Info_1=\&Info_2} \\
&\qquad \sigma_{\&Title_2 \ni \text{“Web”}}(\sigma_{Affiliation=\text{“T-Univ”}}(r_{38}))
\end{aligned}
$$

r_{40}:

&Yamada$_1$	&Tsukuba$_1$	&Number$_1$	&MailAddr$_1$	&Yamada$_2$	&Title$_1$	&Info$_1$	&HomePage$_1$	&Yamada$_3$	&Title$_2$	&Info$_2$	&Abstract$_1$	Affiliation

Restructuring Part

(1) Unnecessary attributes are dropped.

$$
\begin{aligned}
r_{41} := \pi_{\&Yamada_1,\&Tsukuba_1,\&Number_1,} \\
\quad {}_{\&MailAddr_1,\&Yamada_2,\&Title_1,\&Info_1,\&HomePage_1,\&Abstract_1}(r_{40})
\end{aligned}
$$

(2) New publication list pages for authors are constructed according to the output expression.

$$
\begin{aligned}
r_{42} &:= \mathbf{P}_{B(A)\rightarrow C:RC,'pubs'}(\nu_{B=(A)}(\\
&\qquad \mathbf{P}_{(\&Title_1,\&Info_1,\&HomePage_1,\&Abstract_1)\rightarrow A:SC,'new\text{-}p\text{-}item'}(r_{41}))) \\
r_{43} &:= \mathbf{P}_{(\&Yamada_2,C)\rightarrow D:SC,'p\text{-}list'}(\gamma_{\&Yamada_2\rightarrow SD(author)}(r_{42}))
\end{aligned}
$$

(3) The new pages are exported as new Web pages.

$$r_{44} := \mathbf{E}_{D,U_1,'publications','a'}(\mathbf{URL}_{U_1}(r_{43}))$$

(4) The index page is constructed and exported.

$$\begin{aligned} r_{45} := {} & \mathbf{P}_{F(E)\rightarrow H:RC,'table'}(\nu_{F=(E)}(\\ & \quad \mathbf{P}_{(\&Yamada_1,\&Tsukuba_1,\&Number_1,\&MailAddr_1,D) \rightarrow E:SC,'member'}(\gamma_{\&MailAddr_1\rightarrow SD(e\text{-}mail)}(\\ & \quad\quad \gamma_{\&Number_1\rightarrow SD(tel)}(\gamma_{\&Tsukuba_1\rightarrow SD(addr)}(\\ & \quad\quad\quad \gamma_{\&Yamada_1\rightarrow SD(name)}(r_{44}))))))) \\ r_{46} := {} & \delta_{H\rightarrow Document}(\mathbf{E}_{H,U_2,'index\text{-}page','a'}(\mathbf{URL}_{U_2}(r_{45}))) \end{aligned}$$

r_{46}:

Document

7.6.3 Translation Procedure

This subsection explains how to translate an HQBE description into WebNR/SD algebra expressions.

Construction of Extraction Part

For every target relation $D_i (1 \leq i \leq n)$, expression EP_i is constructed in the following way.

1. First, according to the target expression for D_i (called TE_i), EP_i' is constructed. It applies Unnest operations to D_i in order to flatten it. In addition, EP_i' involves Rename operators (δ). They rename every attribute name which is not of SD type and is associated with an example variable $\&example$ into $\&example_i$, where i is a number to distinguish it among the attributes associated with the same example variable.

2. Then, EP_i is constructed by adding Unnest and Unpack operators to EP_i', according to all the loose structure specifications $LS_j (1 \leq j \leq m_i)$ in TE_i. In this construction, region algebra expressions for parameters of Unpack operators should be determined according to the loose structure specifications. For example, "$\supset_d$ *title*.2" in the loose structure specification of TE_3 generates region algebra expression "$(title \cap 2) \subset_d$ **I**."

Construction of Selection and Join Part

1. For every result of EP_i, expression SJP_i is constructed. They select tuples according to selection conditions which are specified in target expressions and condition boxes.
2. Selection and join part expression SJP is constructed by joining all the $SJP_i (1 \leq i \leq n)$. The Join condition is conjunction of the predicates each of which holds if a value of attribute $\&example_i$ is equal to that of attribute $\&example_j (i \neq j)$.

Construction of Restructuring Part

1. Let RP' be an expression which applies projection operator to the result of SJP. It projects out attributes irrelevant to the final result.
2. Operators such as Nest, Pack, and Export are necessary to construct the result structures specified in the output expression. Restructuring part expression RP is obtained by adding such operators to RP', according to the output expression. If Nest operators create new nesting structures whose parents are the same, they are applied in the order specified in the output expression.

Chapter 8

Prototype System Development

8.1 Overview

This chapter describes basic design of a prototype information integration system. First, design principles of the prototype system are explained. Second, its architecture is described. Then, interfaces among the mediator and wrappers are explained. Next, data communication among the modules of the system is explained. Finally, some screen shots of the actual system are shown.

8.2 Design Principles

The prototype system is designed according to the following three principles.

Utilization of Existing Tools The integration system uses hybrid data models which combine nested relational structures and abstract data type concept. Both of them are existing and well-known data modeling constructs, and there are many programs and systems available for their manipulation. Therefore, we can utilize such existing software products. In fact, the prototype system uses ORDBMS Illustra [Inf] as a back-end engine to implement the mediator.

Realization of Extensibility It is desirable that the integration system can accept new types of information sources and wrappers in an easy way. Therefore, the prototype system is designed so that it would require little additional hard coding in incorporating new wrappers into the system. To put it concretely, the mediator and wrappers in the prototype system are designed to have the same interface, and to exchange information about metadata of information sources and query processing power in a dynamic fashion.

Data Transfer On Demand In general, integration of information sources involves a large amount of data objects to be processed. Therefore, software modules such as the visual user interface, the mediator, and wrappers require a lot of working space in integration processes. In this prototype system, data objects are transferred among modules in a lazy fashion in order to reduce the working space and intermediate query processing costs. The transfer control is performed at the tuple level and value level.

8.3 Prototype System Architecture

Figure 8.1 shows the architecture of the prototype system. Opentext index server [Ope], Web, Oracle8 [Ora], and mSQL [Hug] are connected as information sources. The main modules of the system are Opentext wrapper, JDBC wrapper, Web wrapper, the mediator, and the visual user interface module.

The Opentext wrapper provides a relational view on top of the document collections stored in the Opentext database. A document collection is regarded as a relation. The Opentext wrapper translates WebNR/SD algebra expressions into PAT expressions [ST92], which Opentext can accept for document retrieval. PAT expressions are based on region algebra. Thus, WebNR/SD's Selection operator with the ρ predicate can be processed by Opentext. In addition, the Opentext wrapper can transform SD values into ASD values and vice versa. Namely, it can execute **SDA** and **SDM** operators contained in WebNR/SD algebra expressions. Also, when the mediator applies **SDM** operator to a relation stored in the mediator, it delegates transformation between ASD values and SD values to the Opentext wrapper if the values are managed by the wrapper.

The JDBC wrapper communicates with a relational database through a JDBC driver. It translates WebNR/SD algebra expressions including only relational algebra operators into SQL queries. It has translation rules to translate relational algebra expressions into SQL92 intermediate level queries.

The Web wrapper is different from the other wrappers in that it provides no means of query translation because the Web has no query processing capability. It provides a relational view on top of Web pages. For each Web page, there exists a corresponding relation which contains only an Hlink value including the page's URL. The Web wrapper plays the main roles in processing Navigate, Import, and Export operators. When the mediator processes those operators, it delegates value-level operations required for the Web-relevant operators to the Web wrapper. In addition, like Opentext wrapper, it can transform SD values into ASD values and vice versa.

The mediator employs Illustra database management system as a backend engine to support WebNR/SD-based operations. Target data objects which the mediator processes and intermediate results are once stored in the

Illustra database. Integrated operations by the mediator are classified into three categories according to their executors.

- Operations executed by Illustra:

  Because WebNR/SD is a hybrid data model which includes relational data model in it, some operations of WebNR/SD can be processed by Illustra alone. For example, Selection, Projection, and Join operators are executed directly by Illustra.

- Operations cooperatively executed by the mediator and Illustra:

  The mediator module and Illustra cooperate to execute some operators. They include Nest, Unnest, Pack, and Unpack operators.

- Operations by the mediator, Illustra, and wrappers:

  Wrappers play the main roles in executing such operators as SDM, Import, Export, and Navigate operators. In this case, all the three types of modules work. The mediator serves as a coordinator in executing the operators.

The visual user interface module is implemented based on AWT (Abstract Window Toolkit), and gives users visual and interactive supporting tools for integrated operations. First, it tells the mediator to give necessary information for users to specify operations. It translates user's directions into WebNR/SD algebra expressions or other method invocations (such as relation renaming method), and submits them to the mediator.

All codes are written in Java except the mediator-Illustra bridge module which is written in C. This is because Illustra has no API for Java at present. Communication among the modules is attained by HORB (Hirano's Object Request Broker), which was developed at ETL [Hir97].

8.4 Interface among the Mediator and Wrappers

Modules of the prototype systems are implemented as java objects. Therefore, the modules have java interfaces (Figure 8.2). As mentioned before, for extensibility, the mediator and wrappers in the prototype system are designed to have the same interface. The common interface is "NRSD Processor." The other interfaces are optional. The "SD Materializer" interface is used when the mediator delegates transformation between ASD values and SD values to wrappers. The "Navigable" interface is used when the mediator delegates value-level operations required for Web-relevant operators to wrappers. Wrappers which create ASD values and Hlink values are supposed to implement "SD Materializer" and "Navigable," respectively. When a wrapper creates ASD values or HLink values, it embeds the wrapper's id

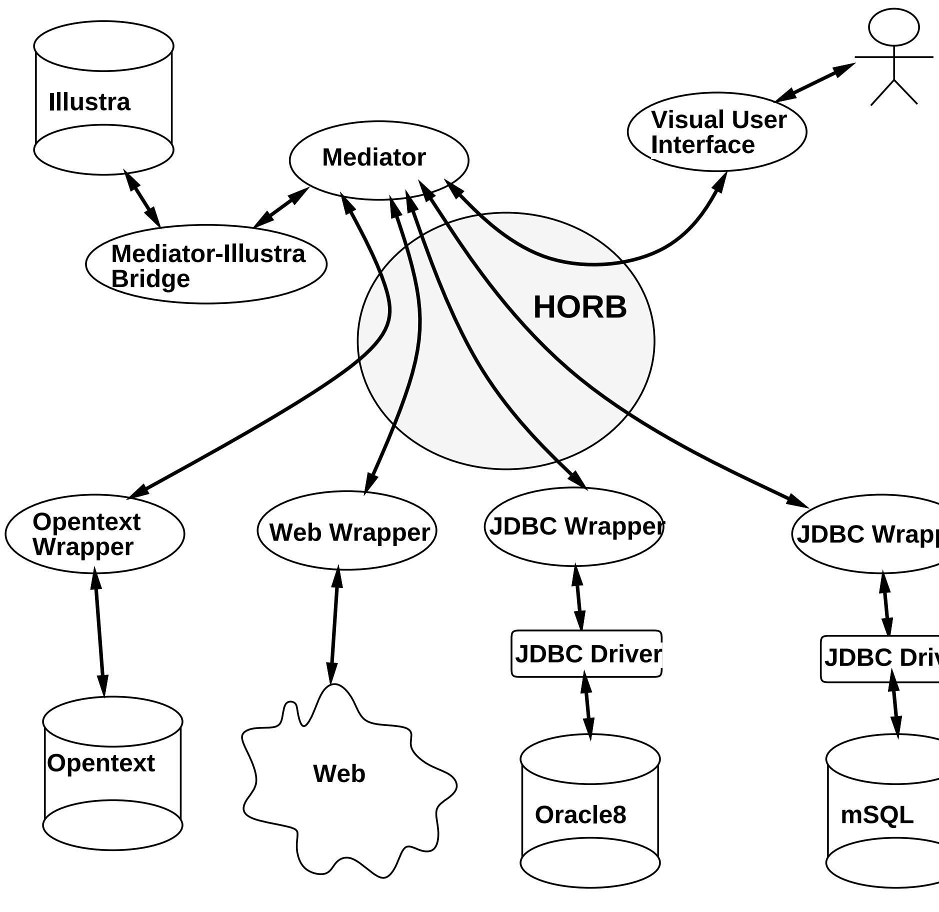


Figure 8.1: Prototype system architecture

into those values. Therefore, the mediator need not know in advance what optional interfaces are implemented by wrappers.

Detailed explanation of the interfaces are given below.

NRSD Processor A module with this interface has capability to process NR/SD algebra expressions. Figure 8.3 shows a part of this interface. Method `getMetaData` returns metadata of all relations which the module can operate. Method `executableExpr` returns information about query processing capability of the module, named *executable expression specification*. For example, The executable expression specification of the mediator tells the module can process any kind of WebNR/SD algebra expressions. That of a JDBC wrapper tells it processes the expressions including only the operators of relational algebra. Method `executeExpr` takes a WebNR/SD algebra expression which is consistent with the executable expression specification. As a return value, it returns a relation handle object for the result relation.

SD Materializer A module with this interface implements the methods which support transformation of ASD values into SD values. Figure 8.4 shows a part of this interface. The `markerToSD` method takes an object which conveys information of a marker (markers are explained in Section 6.2). As a return value, it returns an SD value which contains

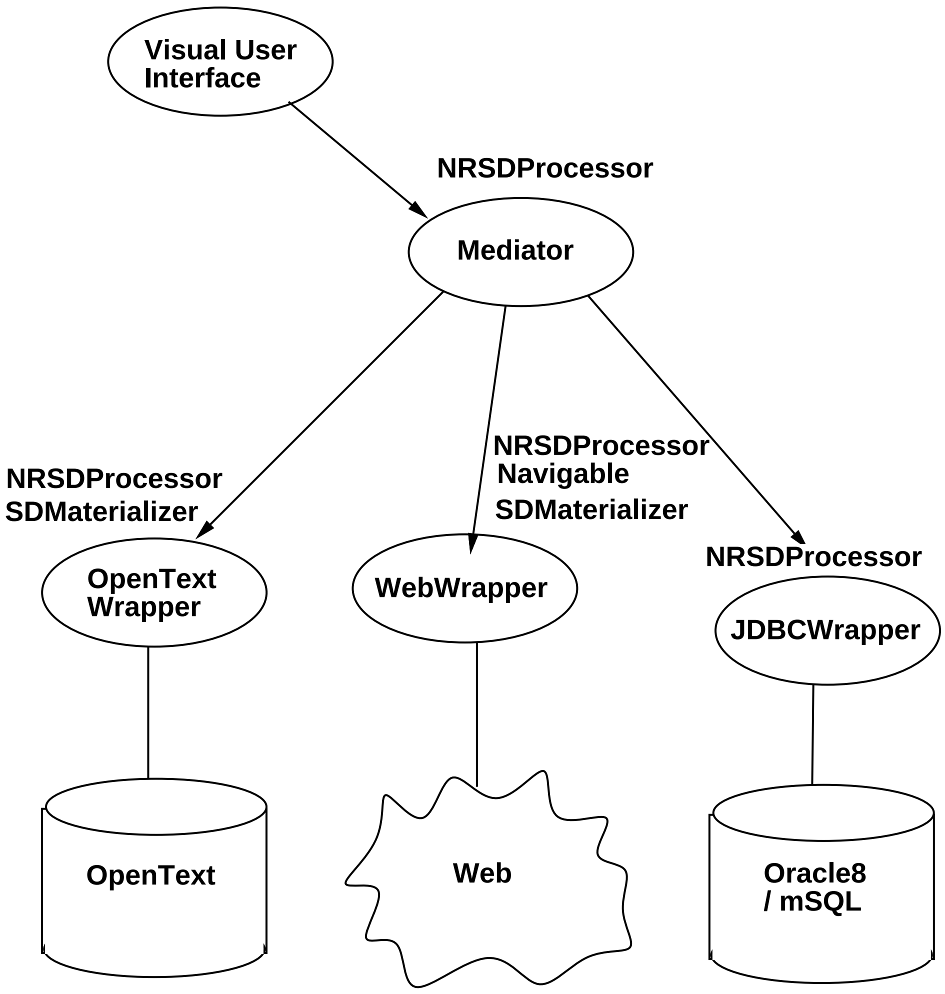


Figure 8.2: Interfaces among modules

the materialized text element body.

Navigable A module with this interface provides a means to process operations relevant to hypertext links. Figure 8.5 shows a part of this interface. Methods `navigate`, `importSD`, and `export` are value-at-a-time versions of Navigate, Import, and Export operators in WebNR/SD algebra. Method `export` returns objects having interface `Anchor`. The `Anchor` is an interface that is implemented by Hlink value objects. It provides methods relevant to hypertext link navigation. Method `importASD` is used to execute $\mathbf{SDA}(\mathbf{Import}(r))$. The method is introduced in order to avoid inefficient execution of the expression. Without this method, execution of $\mathbf{SDA}(\mathbf{Import}(r))$ means that after Import operator fetches the contents of Web pages, **SDA** operator has to discard some part of the contents. The `importASD` is implemented not to fetch unnecessary parts of the contents in advance.

Method `ExecutableExpr` of the NR/SD Processor interface is very important. Although all the wrappers have the common interface "NRSD Processor," wrappers are different in their query processing capability. The `ExecutableExpr` returns description of query processing power of the module at the algebra operator level. Therefore, it allows the mediator to know the difference of query processing capability.

```
public interface NRSDProcessor{

  InfoSourceMetaData getMetaData();
    // returns metadata of the relations the module can operate
  ..

  String   executableExpr();
    // returns executable expression specifications
  RelationHandle executeExpr(NestedList NRSDExpr);
    // returns RelationHandle objects referring to
    // the result relation
  ..

}
```

Figure 8.3: NRSD Processor interface (in part)

```
public interface SDMaterializer {
    SD markerToSD(Marker marker);
}
```

Figure 8.4: SD Materializer interface (in part)

```
public interface Navigable{
  RelationHandle navigate(Anchor start, NestedList pathRegExpr);
  Anchor export(SD sd, Object Locator, String GI, String text)
  SD     importSD(Anchor anchor);
  SD     importASD(Anchor anchor, NestedList as1, NestedList as2);
                  // composite operator equivalent to SDA(Import(R)).
    ...
}
```

Figure 8.5: Navigable interface (in part)

8.5 Data Communication among Modules

Logically, in the integration system, all kinds of data objects — not only relations stored in relational databases, but also document collections and Web pages — are transformed into relations by wrappers. And the unit of data transfer is a relation. However, naive implementation of such a framework requires the modules to spend a lot of working space, and sometime unnecessary data translation and transfer occur. In the prototype system, data objects are transferred on demand. The on-demand policy is realized as follows. (1) It implements the basic spirit of query processing and optimization scheme explained in Chapter 6. It decomposes queries into a number of expressions the wrappers can process, although it does not use query rewriting rules. In addition, it implements methods for ASD values. (2) It transfers relation handle objects instead of relations themselves. Actual tuples are transferred on demand. Thus, unnecessary data objects are not transferred into the visual user interface module and the mediator. For example, the visual user interface module requires one tuple at a time in browsing.

In addition to relations, there are many types of data objects which are transferred among modules. In the following, the major data objects are explained. The modules exchange metadata of information sources, information about query processing capability of the mediator and wrappers, WebNR/SD algebra expression objects, and relation handle objects.

Metadata of information sources Metadata of an information source is represented by an object having `InfoSourceMetaData` interface. Basically, an `InfoSourceMetaData` object consists of a number of objects having `RelationMetaData` interface. Each of them contains metadata of a relation stored in the information source. The `RelationMetaData` interface is designed based on JDBC's `ResultSetMetaData` interface. Figure 8.6 shows a part of `RelationMetaData` interface.

Query processing capability information As described before, modules (i.e. the mediator and wrappers) vary in the degree of the query processing capability. Information about query processing capability of a module is represented by a string-typed data object called an *executable expression specification.* It contains the syntax of executable expressions whose alphabets are WebNR/SD algebra operators. Figure 8.7 shows the executable expression specification of the Opentext wrapper.

WebNR/SD algebra expressions A WebNR/SD algebra expression is represented by a data object of `NestedList` class. For example, expression $\mathbf{U}_{addr\rightarrow(city[city]\ as\ x)}(\pi_{addr}(Faculty \bowtie_{fname=supervisor} (Group \bowtie \sigma_{mname=\text{'Lin'}}(Member))))$ is represented as "(US (addr (city (RIG city) X)) (PROJECTION (addr) (JOIN (= fname supervisor)

(REL Faculty) (NJOIN (REL Group) (SELECTION (= mname 'Lin') (REL member))))))."

Relation handles In the prototype system, relations in WebNR/SD are implemented by relation handle objects. Namely, it is relation handle objects that wrap physical data structures which correspond to logical relations. The actual data structures vary among the mediator and wrappers. When the JDBC wrapper obtains the query result from the JDBC driver, the result is represented by a java object whose type is determined by JDBC API. On the other hand, the Illustra database stores nested relations in the form of a number of flat relations, and the mediator deals with them through cursors. As mentioned before, by transferring relation handle objects instead of objects containing all data in relations, the system realizes on-demand tuple transfer for reduction of working spaces of the modules.

8.6 Screen Shots of the Prototype System

In this section, some screen shots of the prototype system are shown. Figure 8.8 shows a data box for relation "Member" in the Oracle 8 database. It also shows the corresponding metadata box. In this figure, the user is submitting an elementary HQBE description. Figure 8.9 shows a data box for a docu-

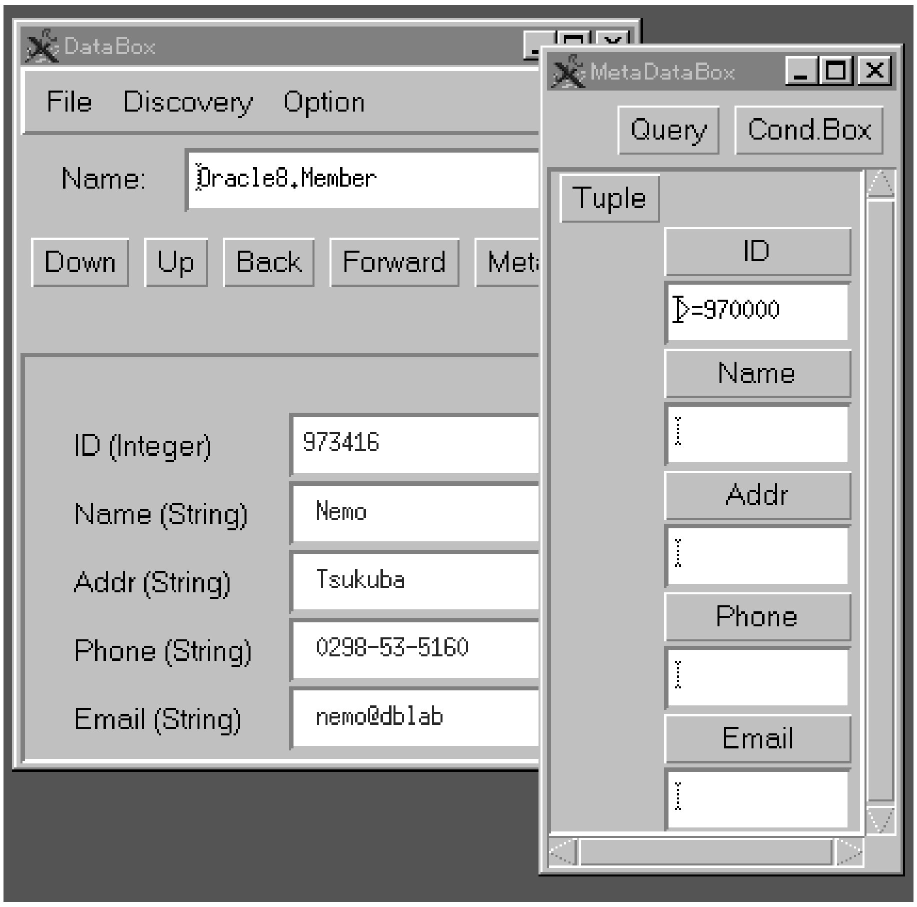
```
public interface RelationMetaData{
  int    getAttCount();  // the number of attributes
  String getAttName(int index); // attribute name of
                                   index-th attribute
  int    getAttIndex(String attName); // inverse of getAttName
  int    getAttType(int index);  // type identifier of
                                    index-th attribute
  String getAttTypeName(int index); // type name of
                                       index-th attribute
  RelationMetaData getMetaData(int index);
                    // RelationMetaData of sub-relations
                       in index-th attribute
}
```

Figure 8.6: RelationMetaData interface (in part)

```
Expr ::= ['SDA'] ['SELECTION_R'] '( REL' RELATION ')' ;
```

Figure 8.7: Executable expression specification for the Opentext wrapper

ment collection in the Opentext database. Figure 8.10 shows a data box for a Web page. The page is written in XML.



Name: Oracle8.Member

ID (Integer) 973416

Name (String) Nemo

Addr (String) Tsukuba

Phone (String) 0298-53-5160

Email (String) nemo@dblab

ID >=970000

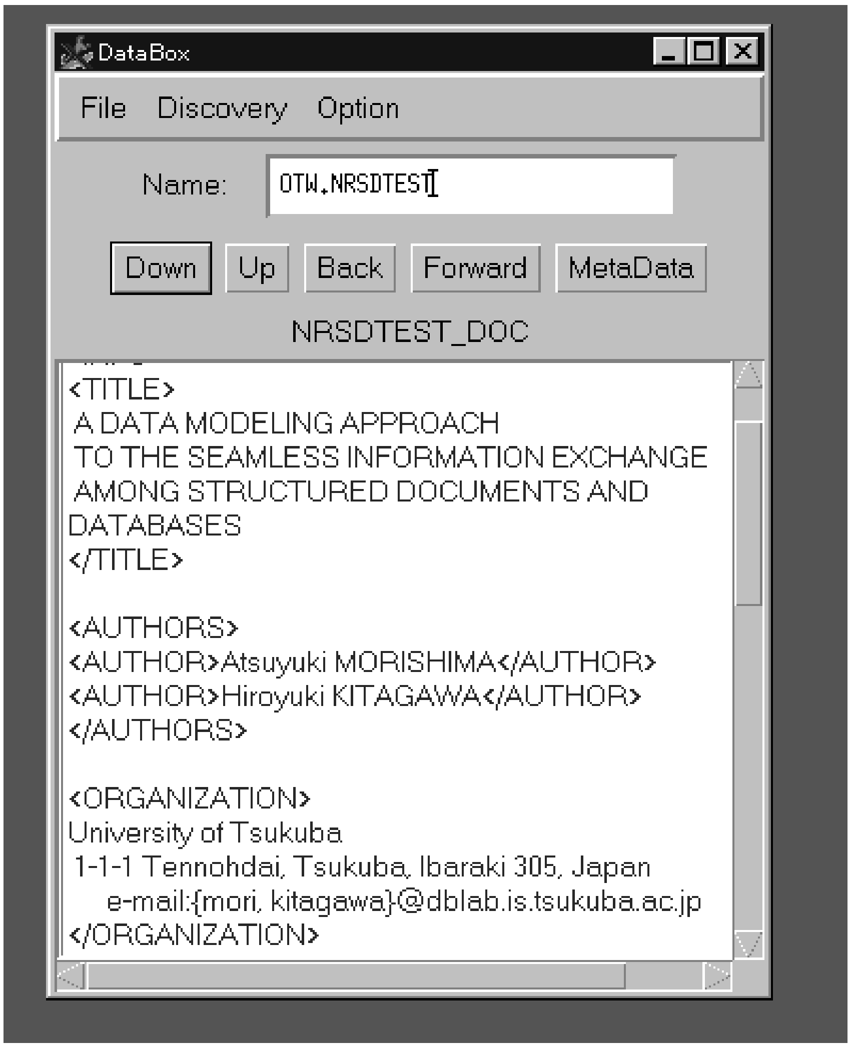
Name

Addr

Phone

Email

Figure 8.8: Data box and metadata box for a relation in the Oracle 8 database

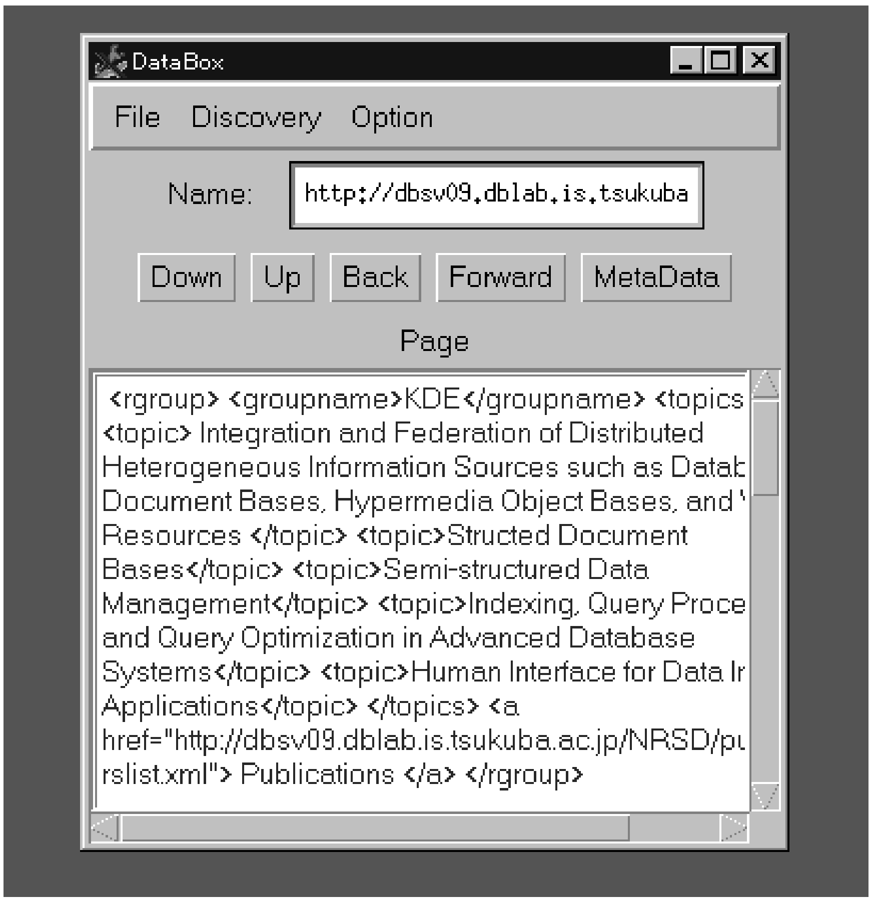


```
<TITLE>
 A DATA MODELING APPROACH
 TO THE SEAMLESS INFORMATION EXCHANGE
 AMONG STRUCTURED DOCUMENTS AND
DATABASES
</TITLE>

<AUTHORS>
<AUTHOR>Atsuyuki MORISHIMA</AUTHOR>
<AUTHOR>Hiroyuki KITAGAWA</AUTHOR>
</AUTHORS>

<ORGANIZATION>
University of Tsukuba
 1-1-1 Tennohdai, Tsukuba, Ibaraki 305, Japan
    e-mail:{mori, kitagawa}@dblab.is.tsukuba.ac.jp
</ORGANIZATION>
```

Figure 8.9: Data box for a document collection in the Opentext database



<rgroup> <groupname>KDE</groupname> <topics
<topic> Integration and Federation of Distributed
Heterogeneous Information Sources such as Datat
Document Bases, Hypermedia Object Bases, and '
Resources </topic> <topic>Structed Document
Bases</topic> <topic>Semi-structured Data
Management</topic> <topic>Indexing, Query Proce
and Query Optimization in Advanced Database
Systems</topic> <topic>Human Interface for Data Ir
Applications</topic> </topics> <a
href="http://dbsv09.dblab.is.tsukuba.ac.jp/NRSD/pu
rslist.xml"> Publications </a> </rgroup>

Figure 8.10: Data box for a Web page

Chapter 9

Conclusions

This dissertation has proposed a new information integration system for structured documents, Web, and relational databases. In order to achieve symmetric and dynamic integration of the information sources, the system employs hybrid and symmetric data models which combine nested relations and SD values with dynamic transformation mechanisms. This dissertation has shown that utilization of the hybrid and symmetric data models is a very promising approach to attain the integration. This is the main contribution of this dissertation. More detailed contributions are summarized below.

Hybrid and Symmetric Data Models

In Chapter 5, three variations of the hybrid and symmetric data models — the NR/SD integration models — have been explained. They are NR/SD, NR/SD+, and WebNR/SD. By providing WebNR/SD algebra expressions that express a complex information integration example, this chapter has shown the applicability of the models to practical integration problems.

Query Processing and Optimization Scheme in the Proposed Integration System

Chapter 6 has shown that the query expressions written in the NR/SD integration models can be processed in the practical system having mediator-based architecture. Also, it has shown that there is a query optimization scheme for more efficient processing in the information integration system based on this approach.

Visual User Interface

Chapter 7 has described design of a visual user interface for integration of structured documents, Web, and relational databases. The user interface gives a means to cope with the largeness and complexity which arise in integration of heterogeneous and distributed information sources. To my knowledge, HQBE is the first visual data manipulation language designed for integration of heterogeneous information sources.

Development of a Prototype System

Chapter 8 has explained the basic design of a prototype system. Currently, the prototype system is working in part. Development of the prototype system has shown that the proposed information integration system is implementable, and that, since this approach is based on existing approaches, the implementation can be effectively supported by existing software products such as Illustra.

I think integration of heterogeneous and distributed information sources continues to be one of the most important issues in practical computer utilization. I hope that in the future the technology of information integration will provide people with the ability of manipulating all kinds of information in the world easily, as they like.

Bibliography

[AB84] Searge Abiteboul and Nicole Bidoit. Non first normal form relations to represent hierarchically organized data. In *Proc. ACM Symposium on Principles of Database Systems*, pages 191–200, Waterloo, Ontario, Canada, 1984.

[AB86] Serge Abiteboul and Nicole Bidoit. Non first normal form relations: An algebra allowing data restructuring. *Journal of Computer and System Sciences*, 33(3):361–393, December 1986.

[Abi97] Serge Abiteboul. Querying semi-structured data. In *Proc. 6th International Conference on Data Theory (ICDT'97)*, pages 1–18, 1997.

[ACM93] Serge Abiteboul, Sophie Cluet, and Tova Milo. Querying and updating the file. In *Proc. 19th VLDB Conference.*, pages 73–84, 1993.

[ACM95] Serge Abiteboul, Sophie Cluet, and Tova Milo. A database interface for file update. In *Proc. ACM SIGMOD Conference.*, pages 386–397, 1995.

[AFS89] Serge Abiteboul, Patrick C. Fischer, and H. J. Scheck. *Nested Relations and Complex Objects in Databases.* no. 361 in Lecture Notes in Computer Science. Springer-Verlag, 1989.

[AQM$^+$97] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet L. Wiener. The Lorel query language for semistructured data. *International Journal on Digital Libraries*, 1(1):68–88, 1997.

[AV97] Serge Abiteboul and Victor Vianu. Queries and computation on the Web. In *Proc. 6th International Conference on Data Theory (ICDT'97)*, pages 262–275, 1997.

[BCK$^+$94] G. Elizabeth Blake, Mariano P. Consens, Pekka Kilpeläinen, P. Larson, T. Snider, and F. Tompa. Text/relational database management systems: Harmonizing SQL and SGML. In *Proc. International Conference on Applications of Databases, no. 819 in Lecture Notes in Computer Science*, pages 267–280, Vadstena, Sweden, 1994.

[BDH$^+$95] Peter Buneman, Susan B. Davidson, Kyle Hart, G. Christian Overton, and Limsoon Wong. A data transformation system for biological data sources. In *Proc. 21st VLDB Conference*, pages 158–169, 1995.

[BDHS96] Peter Buneman, Susan B. Davidson, Gerd G. Hillebrand, and Dan Suciu. A query language and optimization techniques for unstructrured data. In *Proc. ACM SIGMOD Conference*, pages 505–516, 1996.

[BLS$^+$94] Peter Buneman, Leonid Libkin, Dan Suciu, Val Tannen, and Limsoon Wong. Comprehension syntax. *SIGMOD Record*, 23(1), 1994.

[BSOO96] N. H. Balkir, E. Sukan, Gultekin Ozsoyoglu, and M. Ozsoyoglu. VISUAL: A graphical icon-based query language. In *Proc. DE Conference*, pages 524–533, 1996.

[Bun97] Peter Buneman. Semistructured data. In *Proc. 16th ACM Symposium on Principles of Database Systems (PODS'97)*, pages 117–121, 1997.

[Bur91] Forbes J. Burkowski. An algebra for hierarchically organized text-dominated databases. *Information Processing & Management*, 28(3):333–348, 1991.

[CACS94] Vassilis Christophides, Serge Abiteboul, Sophie Cluet, and Michel Scholl. From structured documents to novel query facilities. In *Proc. ACM SIGMOD Conference*, pages 313–324, 1994.

[CCB95] Charles L. A. Clarke, Gordon V. Cormack, and Forbes J. Burkowski. An algebra for structured text search and a framework for its implementation. *The Computer Journal*, 38(1):43–56, 1995.

[CGT75] Donald D. Chamberlin, Jim N. Gray, and Irving L. Traiger. Views, authorization, and locking in a relational database system. In *Proc. National Computer Conference*, pages 425–430, Anaheim, 1975.

[CHMW96] Michael Carey, Laura Haas, Vivek Maganty, and John Williams. PESTO: An integrated query/browser for object databases. In *Proc. VLDB Conference*, pages 203–214, Munbai, India, 1996.

[CM94] Mariano P. Consens and Tova Milo. Optimizing queries on files. In *Proc. ACM SIGMOD Conference*, pages 301–312, Minneapolis, Minnesota, 1994.

[CM95] Mariano P. Consens and Tova Milo. Algebras for querying text regions. In *Proc. ACM Symposium on Principles of Database Systems*, pages 11–22, 1995.

[Cod70] E. F. Codd. A relational model for large shared databank. *Communications of the ACM*, 13(6):377–387, May 1970.

[Col89] Latha S. Colby. A recursive algebra and query optimization for nested relations. In *Proc. ACM SIGMOD Conference*, pages 273–283, Portland, OR, 1989.

[Col90] Latha S. Colby. A recursive algebra for nested relations. *Information Systems*, 15(5):567–582, 1990.

[CSG94] Latha S. Colby, Lawrence V. Saxton, and Dirk Van Gucht. Concepts for modeling and querying list-structured data. *Information Processing & Management*, 30(5):687–709, 1994.

[CST92] W. Bruce Croft, Lisa Ann Smith, and Howard R. Turtle. A loosely-coupled integration of a text retrieval system and an object-oriented database system. In *Proc. ACM SIGIR Conference*, pages 223–232, 1992.

[DKS92] Weimin Du, Ravi Krishnamurthy, and Ming-Chien Shan. Query optimization in heterogeneous DBMS. In *Proc. of the 18th VLDB Conference*, pages 277–291, Vancouver, British Columbia, Canada, 1992.

[EP90] Ahmed K. Elmagarmid and Calton Pu, editors. *Special Issues on Heterogeneous Databases*, volume 20 of *ACM Computing Surveys*. 1990.

[FFK$^+$97] Mary F. Fernandez, Daniela Florescu, Jaewwoo Kang, Alon Y. Levy, and Dan Suciu. STRUDEL: A Web-site management system. In *Proc. SIGMOD Conference*, pages 549–552, Tucson, May 1997.

[FSTG85] Patrick C. Fischer, Lawrence V. Saxton, Stan J. Thomas, and Dirk Van Gucht. Interactions between dependencies and nested relational structures. *Journal of Computer and System Sciences*, 31(3):343–354, 1985.

[FT83] Patrick C. Fischer and Stan J. Thomas. Operators for non-first-normal-form relations. In *Proc. IEEE COMPSAC83*, pages 464–475, Chicago, 1983.

[FWM97] Thorsten Fiebig, Jürgen Weiss, and Guido Moerkotte. RAW: A relational algebra for the Web. In *Proc. of the Workshop on Management of Semi-structured Data*, pages 34–41, May 1997.

[GF88] Dirk Van Gucht and Patrick C. Fischer. Multilevel nested relational structures. *Journal of Computer and System Sciences*, 36(1):77–105, 1988.

[GG88] Marc Gyssens and Dirk Van Gucht. The powerset algebra as a result of adding programming constructs to the nested relational algebra. In *Proc. ACM SIGMOD Conference*, pages 225–232, Chicago, Illinois, 1988.

[GKD97] Michael R. Genesereth, Arthur M. Keller, and Oliver M. Duschka. Infomaster: An information integration system. In *Proc. SIGMOD Conference*, pages 539–542, Tucson, May 1997.

[Guc87] Dirk Van Guch. On the expressive power of the extended relational algebra for the unnormalized relational model. In *Proc. ACM Symposium on Principles of Database Systems*, pages 302–312, San Diego, California, 1987.

[GW97] Roy Goldman and Jennifer Widom. DataGuides: Enabling query formulation and optimization in semistructured databases. In *Proc. VLDB Conference*, Atheans, Greece, 1997.

[GZ89] Ralf H. Güting and Roberto Zicari. An introduction to the nested sequences of tuples data model and algebra. In *Nested Relations and Complex Objects in Databases*, no. 361 in Lecture Notes in Computer Science. Springer-Verlag, 1989.

[GZC89] Ralf H. Güting, Roberto Zicari, and David M. Choy. An algebra for structured office documents. *ACM Trans. Office Information Systems*, 7(4):123–157, April 1989.

[HBP94] A. R. Hurson, M. W. Bright, and Simin H. Pakzad, editors. *Multidatabase Systems: An Advanced Solution for Global Information Sharing*. IEEE Computer Society Press, 1994.

[HIL95] Eben M. Haber, Yannis E. Ioannidis, and Miron Livny. OPOSSUM: Desk-top schema management through customizable visualization. In *Proc. VLDB Conference*, pages 527–538, Zurich, Switzerland, 1995.

[Hir97] Satoshi Hirano. HORB: Distributed execution of java programs. In *Proc. International Conference on Worldwide Computing and Its Applications*, pages 29–42, Tsukuba, Japan, 1997.

[HN96] Joseph M. Hellerstein and Jeffery F. Naughton. Query execution techniques for caching expensive methods. In *Proc. SIGMOD Conference*, pages 423–434, Montreal, Canada, 1996.

[HS93] Joseph M. Hellerstein and Michael Stonebraker. Predicate migration: Optimizing queries with expensive predicates. In *Proc. SIGMOD Conference*, pages 267–276, Washington, DC, USA, 1993.

[Hug] Hughes Technologies, Ltd. Hughes technologies web site. `http://www.Hughes.com.au/`.

[HY84] Richard Hull and Chee K. Yap. The format model: A theory of database organization. *Journal of the ACM*, 31(3):518–537, 1984.

[Inf] Informix Software, Inc. Informix home page. `http://www.informix.com/`.

[ISO86] ISO. Information processing – text and office system – standard generalized markup language (SGML), 1986. ISO 8879.

[ISO92] ISO. Hypermedia/time-based structuring language (Hytime), 1992. ISO/IEC 10744.

[JN95] Kalervo Järvelin and Timo Niemi. An NF^2 relational interface for document retrieval, restructuring and aggregation. In *Proc. SIGIR Conference*, pages 102–110, 1995.

[JS82] G. Jaeschke and H.-J. Schek. Remarks on the algebra of non first normal form relations. In *Proc. ACM Symposiumon on Principles of Database Systems*, pages 124–138, Los Angeles, CA, 1982.

[KD95] Christian Kalus and Peter Dadam. Flexible relations – operational support of variang relational structures. In *Proc. of the 21th VLDB Conference*, pages 539–550, Zurich, Switzerland, 1995.

[KK89] Hiroyuki Kitagawa and Tosiyasu L. Kunii. *The Unnormalized Relational Data Model — For Office Form Processor Design—*. Springer-Verlag, 1989.

[Kra97] Ralf Kramer. Databases on the Web: Technologies for federation architectures and case studies. In *Proc. SIGMOD Conference*, pages 503–506, Tucson, May 1997.

[KS95] David Konopnicki and Oded Shmueli. W3QS: A query system for the world-wide web. In *Proc. of VLDB Conference*, pages 54–65, 1995.

[LSS96] Laks V. S. Lakshmannan, Fereidoon Sadri, and Iyer N. Subramanian. A declarative language for querying and restructuring the Web. In *Proc. of the 6th Int. Workshop on Research Issues in Data Eng. (RIDE'96)*, February 1996.

[Mai83] David Maier. *The Theory of Relational Databases.* Computer Science Press, 1983.

[Mak77] Akifumi Makinouchi. A consideration on normal form of not-necessarily-normalized relation in the relational data model. In *Proc. 3rd VLDB Conference*, pages 447–453, 1977.

[MK93] Lil Mohan and Rangasami L. Kashyap. A visual query language for graphical interaction with schema-intensive databases. *IEEE Trans. Knowledge and Data Engineering*, 5(5):843–858, 1993.

[MK97a] Atsuyuki Morishima and Hiroyuki Kitagawa. A data modeling and query processing scheme for integration of structured document repositories and relational databases. In *Proc. Fifth International Conference on Database Systems for Advanced Applications (DASFAA'97)*, pages 145–154, Melbourne, April 1997.

[MK97b] Atsuyuki Morishima and Hiroyuki Kitagawa. A data modeling approach to the seamless information exchange among structured documents and databases. In *Proc. 1997 ACM Symposium on Applied Computing (ACM SAC'97)*, pages 78–87, San Jose, Feburary 1997.

[MK97c] Atsuyuki Morishima and Hiroyuki Kitagawa. Integrated querying and restructuring of the world wide web and databases. In *Proc. International Symposium on Digital Media Information Base (DMIB'97)*, pages 261–271, Nara, Japan, 1997.

[MK98] Atsuyuki Morishima and Hiroyuki Kitagawa. NR/SD+ data model and its query processing — for integration of structured documents and relational databases. *Transactions of Information Processing Society of Japan*, 39(4):964–967, 1998. (in Japanese, with English Abstract).

[MM97] Alberto O. Mendelzon and Tova Milo. Formal models of Web queries. In *Proc. 16th ACM Symposium on Principles of Database Systems (PODS'97)*, pages 134–143, 1997.

[MMM96] Alberto O. Mendelzon, George A. Mihaila, and Tova Milo. Querying the world wide web. In *Proc. PDIS'96*, pages 80–91, December 1996.

[NS96] Tam Nguyen and V. Srinivasan. Accessing relational databases from the world wide web. In *Proc. SIGMOD Conference*, pages 529–540, Montreal, 1996.

[ÖMÖ89] Gultekin Özsoyoğlu, Victor Matos, and Z. Meral Özsoyoğlu. Query processing techniques in the summary-table-by-example database query language. *ACM TODS*, 14(4):526–573, 1989.

[Ope] Opentext Corporation. Opentext home page. `http://www.opentext.com/`.

[Ora] Oracle Corporation. Oracle home page. `http://www.oracle.com/`.

[ÖW89] Gultekin Özsoyoğlu and Huaqing Wang. A relational calculus with set operators, its safety, and equivalent graphical languages. *IEEE Trans. on Software Engineering*, 15(9):1038–1052, Sept. 1989.

[PBE95] Evaggelia Pitoura, Omran A. Bukhres, and Ahmed K. Elmagarmid. Object orientation in multidatabase systems. *ACM Computing Surveys*, 27(2):141–195, June 1995.

[PGMW95] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object exchange across heterogeneous information sources. In *Proc. 11th Data Engineering Conference*, pages 251–260, 1995.

[PGMW96] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. MedMaker: A mediation system based on declarative specifications. In *Proc. 12th Data Engineering Conference*, pages 132–141, 1996.

[QRS$^+$95] Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey Ullman, and Jennifer Widom. Querying semistructured heterogeneous information. In *Proc. 4th International Conference on Deductive and Object-Oriented Databases (DOOD'95)*, pages 319–344, Singapore, 1995.

[RAH$^+$96] Mary Tork Roth, Manish Arya, Laura M. Haas, Michael J. Carey, William F. Cody, Ronald Fagin, Peter M. Schwarz, Joachim Thomas II, and Edward L. Wimmers. The garlic project. In *Proc. SIGMOD Conference*, page 557, Montreal, Canada, 1996.

[RS97] Mary Tork Roth and Peter M. Schwarz. Don't scrap it, wrap it! a wrapper architecture for legacy data sources. In *Proc. VLDB Conference*, pages 266–275, Athens, Greece, 1997.

[SDAMZ94] Ron Sacks-Davis, T. Arnold-Moore, and J. Zobel. Database systems for structured documents. In *Proc. International Symposium on Advanced Database Technologies and Their Integration*, pages 272–283, Nara, Japan, 1994.

[SDKR$^+$95] Ron Sacks-Davis, A. Kent, K. Ramamohanarao, J. Thom, and J. Zobel. Atlas: A nested relational database system for text applications. *IEEE Trans. Knowledge and Data Engineering*, 7(3):454–470, 1995.

[ST92] Airi Salminen and Frank Wm. Tompa. PAT expressions: an algebra for text search. In *Proc. the 2nd International Conference on Computational Lexicography (COMPLEX '92)*, pages 309–332, 1992.

[TF86] Stan J. Thomas and Patric C. Fischer. Nested relational structures. *Advances in Computing Research*, 3:269–307, 1986.

[Tri91] Phil Trinder. Comprehensions, a query notation for DBPLs. In *Proc. International Workshop on Database Programming Languages*, pages 49–62, 1991.

[VAB96] Marc Volz, Karl Aberer, and Klemens Böhm. Applying a flexible OODBMS-IRS-coupling to structured document handling. In *Proc. 12th Data Engineering Conference*, 1996.

[W3C97] W3C. HTML 4.0 specification, December 1997. World Wide Web Consortium Recommendations, `http://www.w3.org/`.

[W3C98] W3C. Extensible markup language (XML) 1.0, February 1998. World Wide Web Consortium Recommendations, `http://www.w3.org/`.

[Wie92] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, March 1992.

[YA94] Tak W. Yan and Jurgen Annevelink. Integrating a structured-text retrieval system with an object-oriented database system. In *Proc. 20th VLDB Conference*, pages 740–749, Santiago, Chile, 1994.

[YIU96] Masatoshi Yoshikawa, Osamu Ichikawa, and Shunsuke Uemura. Amalgamating SGML documents and databases. In *Proc. 5th International Conference on Extending Database Technology*, March 1996.

[YKY$^+$91] Kenichi Yajima, Hiroyuki Kitagawa, Kazunori Yamaguchi, Nobuo Ohbo, and Yuzuru Fujiwara. Optimization of queries including ADT functions. In *Proc. of International Symposium on Database Systems for Advanced Applications (DASFAA'91)*, pages 366–373, Tokyo, Japan, 1991.

[Zlo77] Mosh M. Zloof. Query by example: a data base language. *IBM Systems Journal*, 16(4):324–343, 1977.