WebDB: A System for Querying Semistructured Data on the Web

Wen-Syan Li Junho Shim[†] K. Selçuk Candan[†]

C&C Research Laboratories
NEC USA, Inc.
110 Rio Robles, M/S SJ100
San Jose, CA 95134, USA
Email:wen@ccrl.sj.nec.com
Tel:(408)943-3008 Fax:(408)943-3099

Abstract

The World-Wide Web can be viewed as a collection of semistructured multimedia documents in the form of Web pages connected through hyperlinks. Unlike most Web search engines, which primarily focus on information retrieval functionality, WebDB aims at supporting a comprehensive database-like query functionality, including selection, aggregation, sorting, summary, grouping, and projection. WebDB allows users to access (1) document level information, such as title, URL, length, keywords, types, and last modified date; (2) intra-document structures, such as tables, forms, and images; and (3) inter-document linkage information, such as destination URLs and anchors. With these three types of information, comprehensive queries can be answered for complex Web-based applications, such as Web mining and Web site management, can be answered. WebDB is based on object-relational concepts: Object-oriented modeling and relational query language. In this paper, we present the data model, language, and implementation of WebDB. We also present, the novel visual query/browsing interface for semistructured Web and Web documents. Our system provides high usability compared with other existing systems.

Key words. WWW, semistructured data, Web database, Web query language, SQL3, object-relational DBMS, visual user interface

1 Introduction

The World-Wide Web can be viewed as a collection of semi-structured multimedia documents (pages) connected through hyperlinks. The Web contains not only the information *explicitly* displayed on each Web page but also intra-document structures and inter-document linkage information. For example, the fact that "owners of Web pages with links pointing to *www.nba.com* are more likely to be NBA fans" would make such a query asking for such pages would be valuable for marketing purposes. Similarly, the result for the query "retrieve the pages in the IRS Web site that contains the keyword *1040 tax form* and an HTML Form" can be used to locate and download the electronic version of the 1040 Tax Form.

Most existing WWW query systems focus on searching document level information. As a result, the queries listed above are not supported. For instance, in the tax form example, the users would specify

[†]This work was performed when the author visited NEC USA, CCRL.

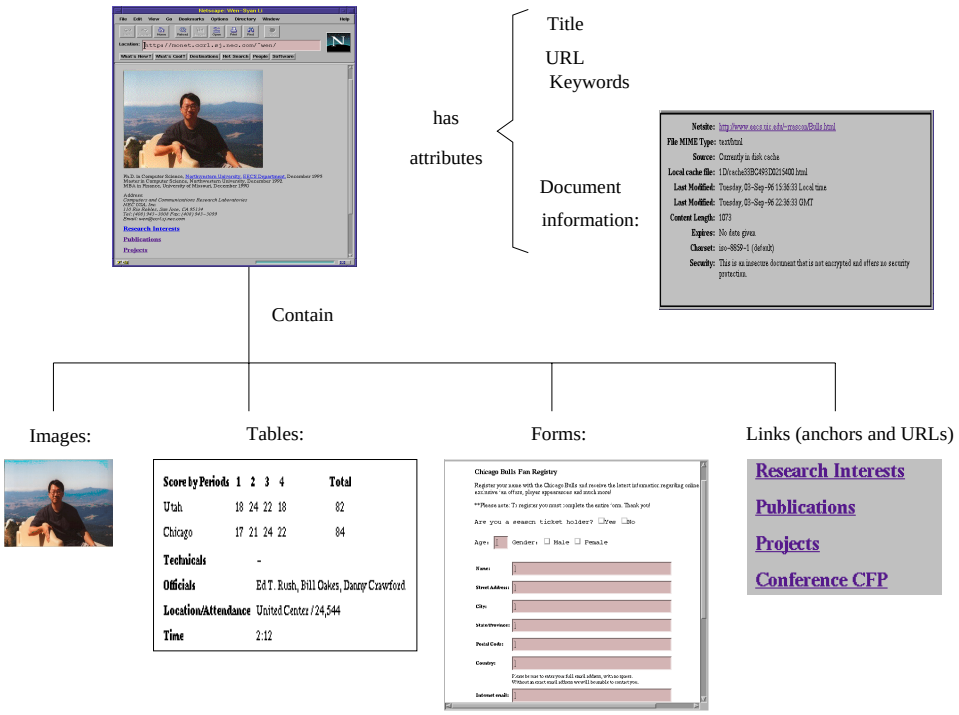

Figure 1: Hierarchical Structure of HTML Pages

the keyword "1040 tax form" but then they would need to browse through the results, including less relevant pages, such as those pages explain the purposes of Form 1040, in order to find the page where an electronic format of the 1040 Tax Form is available for downloading.

We categorize information available on the Web as follows:

- Document information, such as type, size, last modified date, URL, page title, keywords, etc. Most Web search engines support searching only on this information.
- Inter-documentation information including links from/to/within a page and anchor labels. Links within a page are through so-called *labels*.
- Intra-document information, such as forms, images, tables, and links. For some of these, we can further explore their internal structures. For example, there may be forms, tables, images, or links within the cells of a given table.

We illustrate the structure of Web pages in Figure 1. We view the Web as a graph, in which pages are vertices and the links among them are edges. By exploring the Web document internal structures, we view Web pages as hierarchical structures. Most existing Web search engines primarily support simple information retrieval. With all these three types of information, our work aims at supporting functionality beyond existing Web search engines support, such as follows:

- **Extracting the Web structure and applying searches to inter-document linkage information:** For example, results of the query "Retrieve all pages with a link pointing to the NEC Web site" can be used to explore the potential customer base for NEC products. Alternatively,

results of the query "List the top ten sites to which pages link on the NEC Web site, sorted by numbers of links" can be used to find popular sites on the Web.

- **Extracting Web document internal structures and applying searches to intra-document structures:** For example, the query "retrieve all pages that have the keyword "*Michael Jordan*" and have the keyword *statistics* within a table" can be used to retrieve pages that may contain Michael Jordan's statistics. A query "retrieve all pages that have the keywords *Chicago Bulls* and *Fan Registration* and have an HTML form in the pages" can be used to retrieve documents associated with on-line registration information for the Chicago Bulls Fan Club.

- **Multimedia search capabilities:** Since Web pages contain multimedia information, users need to be able to search Web documents based on media contents. An example query of this type is "retrieve all documents with an image similar to a Chicago Bulls logo (bull_logo.gif) within a table".

- **Supporting more comprehensive query capabilities:** Most Web search engines support keyword-based page retrieval. WebDB aims at supporting more comprehensive query capability, including aggregation, sorting, select, and projection on document-level metadata as well as inter- and intra-document information. In other words, WebDB is a Web search system which provides full-fledged database-like query functionality. An example comprehensive query that WebDB can process is "retrieve all pages, modified after 1997, which are linked from www.nba.com with depth of 10, sort the results by their URLs, and remove duplicate pages." This query can be used as a *spider* to collect documents from www.nba.com and to organize the results. The query "retrieve all pages which have links to www.nba.com, group them by country of url locations, and display the numbers of pages for each country" can be viewed as using a query to conduct a market survey for geographic locations of NBA fans. WebDB also supports queries on complex intra-document structures, such as "retrieve documents with a form within a table".

- **Providing computer-human interaction:** Some systems provide database-like Web query functionality. However, these systems are generally not practical since they do not provide their end-users with appropriate tools that can not formulate such complex queries. WebDB provides higher usability through strong emphasis on computer-human interaction design. A visual query interface (WebIFQ) and a query statement generator to assist users in formulating complex queries are essential parts of WebDB.

WebDB aims at provide tools that will allow users to access the data available over the net despite its semi-structured nature. In fact, by providing different levels of access, WebDB allows users to extract and exploit the information captured in the structure itself.

1.1 System Overview

WebDB project, at NEC C&C Research laboratories in San Jose, WebDB project aims at building a hypermedia database system for various types of semistructured data, such as Web documents and SGML documents. Compared with other existing systems, WebDB provides a more comprehensive

approach with focuses on modeling, language design, query processing, and *usability*. Consequently, WebDB has a strong emphasis on query interface design and computer human interaction issues such as query expansion and interactive query reformulation. Let us look an example. A user wants to retrieve all Web pages containing both an HTML form and the keyword *multimedia* (or other terms related by semantic similarity or co-occurrence relationship) which have links to the NEC Web sites in *www.ccrl.neclab.com* within link depth of 3. Users are specifically interested in the URLs of the NEC pages which are accessible through these outside pages.

WebDB is based on the object-relational concept, which can be viewed as an extended ER model. The Web contains documents (entities) and hyperlinks connecting these documents (relations). The intra-document structures are modeled using the object-oriented model. The inter-document linkage is modeled through join operations between source document URLs and destination document URLs. The above query can be specified using the Web Query Language (WQL - in the object-relational implementation we translate this query language to SQL3) that WebDB provides as follows:

Select Doc D2.URL, "$\longrightarrow$", Doc D1.URL
From WebDB
Where
D1.URL like "www.ccrl.neclab.com*"[1]
and D1.Keyword mentions ("multimedia" or s-like("multimedia",5) or cooccurrence("multimedia",5))
and D1 contains Form F1
and F1 mentions "*"
and D2 contains Link L1
and L1.URL = D1.URL Depth 3
and L1.TextualAnchor mentions "NEC"

Note that the projected string "$\longrightarrow$" is for the purpose of output presentation and *mentions* is a string matching function for a set of strings, such as a keyword list. Figure **??** shows that users use a visual query interface, WebIFQ (Web In-Frame-Query), to specify queries, rather than using the complex query language directly. The corresponding query statements are automatically generated by WebIFQ. The query statement in WQL is then processed by the WebDB query processor. The result of the above query may be as follows:

http://www.ece.nwu.edu/$\sim$ shimjh $\longrightarrow$ http://www.ccrl.neclab.com/forum97/
http://www.ece.nwu.edu/$\sim$ shimjh $\longrightarrow$ http://www.ccrl.neclab.com/
http://www.ece.nwu.edu/$\sim$ shimjh $\longrightarrow$ http://www.ccrl.neclab.com/Anecdote
http://www.ece.nwu.edu/$\sim$ shimjh $\longrightarrow$ http://www.ccrl.neclab.com/nec_sj/
http://www.ece.nwu.edu/$\sim$ acura $\longrightarrow$ http://www.ccrl.neclab.com/forum97/
...

Note that the result of this query is not a collection of URLs, but reorganized information: pairs of Web pages. Additional reorganization functionalities are also supported for sorting, aggregating, or filtering out information. The result is presented to the user through a browser, such as Netscape

[1] Please note that URL and domain are different. For example, www.ccrl.neclab.com/forum97/index.html is a URL while www.ccrl.neclab.com is a domain. WebDB stores documents' URLs, rather than domain names, to support queries with fine granularity.

Navigator. The user can click on any of the presented URLs to browse a particular page or can save these URLs as bookmarks for later use. WebDB also supports *slide-show* functionality, i.e. automated display of all pages or selected pages (e.g. first 10 pages).

WebDB supports services beyond existing Web search engines can and features many additional *query supporting services* to provide better usability than existing Web query systems with limited database-like functionality. WebDB features the following novel *query supporting services*: (1) WebDB provides a visual query interface and query generator, WebIFQ. Users are not required to know the underlying complex modeling schemes or language syntax. WebIFQ can be used as a general purpose Web query interface for WebDB and Web related applications built on top of ORDBMSs; (2) Recursively exploring and modeling intra-document structures; (3) WebIFQ visualizes user query specification, including intra-document structures and linkage information; (4) WebDB can be used through existing browsers as browsing interfaces; (5) WebDB supports query relaxation through *semantically relevant term index* and *cooccurring term index*; and (5) it provides database query functionality while maintaining file system-based implementation level response time through the use of advanced indexing techniques.

1.2 Paper Organization

The rest of this paper is organized as follows: We first review related work. In Section 3, we present our Web modeling scheme and the corresponding data model for implementing WebDB. In Section 4, we give the formal definitions and syntax of WQL. In Section 5, we present the design and operations of WebIFQ, the visual query and browsing interfaces in WebDB. In Section We give our conclusions in Section 6.

2 Related Work

In[1], we introduced a hybrid multimedia database system, SEMCOG, to support *content-oriented integration* through the integration of semantics and cognition-based approaches. In SEMCOG, images are modeled based on the object-oriented concept and the query language is based on SQL3: multimedia is modeled as user-defined data types and user-defined functions are defined to extend SQL for manipulating multimedia data types. SEMCOG provides a visual query interface, IFQ (In Frame Query)[2], to support complex SQL3-based multimedia database query statements. WebDB's system architecture is similar to SEMCOG: it is based on object-relational concept and its language, WQL, is based on SQL3. The modeling methodology in WebDB is more complicated than in SEMCOG because both intra-document structures and inter-document linkage are modeled. WebDB also features a visual query interface and a query generator similar to IFQ to assist users in specifying complex SQL3 queries.

Our work integrates image retrieval, semi-structured data modeling and retrieval, information retrieval, and user interfaces. In this section, we review related work in these areas and compare it with WebDB.

2.1 Text-based Web Search

Most information retrieval engines provide search capabilities only by keyword or phrase and criteria combinations using Boolean expressions. This type of system is meant for simple information retrieval from a large amount of information. Examples of these systems include Altavista[3], InfoSeek[4], Yahoo[5], and Excite[6]. They provide information retrieval without considering the Web structure and multimedia components of Web documents. Altavista is distinct as it includes a query refinement interface called *Live Topic*. WebDB supports query refinement as well as query relaxation and query reformulation.

Many search engines rank query results by how well they match the original query. [7] discusses strategies for solving the so-called "meta-ranking" problem and proposes a method for merging ranks from heterogeneous Internet sources, particularly in an integrated environment of multiple digital libraries. Similar to [7], I.SEE[8] also addresses the functionality of searching multiple resources. I.SEE is an integrated interface to search engines on the Internet developed at Rensselaer Polytechnic Institute. It translates user queries and forwards them to multiple search engines including AltaVista, Excite, HotBot, Infoseek, and Yahoo!. These two systems' functionalities are limited by the search engines where queries are forwarded to, while WebDB aims at supporting additional functionalities.

2.2 Multimedia-based Web Search

Yahoo's Image Surfer[9] includes a collection of images that are available on the Web. The images from the Web Site are categorized manually; in these categories, users can retrieve images based on color histograms.

WebSeer[10] connects the textual information and images in HTML documents. By integrating an analysis about the images, WebSeer provides the multimedia search based on the image contents and keywords.

In Yahoo! and WebSeer, the integration is at the level of the whole image. WebDB provides image multimedia searches at the object level, as described in [1], since WebDB's underlying media processing capability is supported through SEMCOG[1].

2.3 Complex Search on the Web or the Intranet

LORE[11] at Stanford University is an effort to develop a lightweight object repository. The query language for LORE, LOREL, is an SQL-like query language designed for querying semi-structured heterogeneous information on the Intranet environment. LOREL extends the concept of column names to information/object path descriptions and provides function calls, such as *grep* to support more flexible string matching in information retrieval. DataGuides[12] is a graphical interface for LORE. LOREL is similar to WQL in syntax and DataGuides is similar to the visual query interface in our system. WebDB supports additional multimedia search functionalities, visual query/browsing interfaces, and a SQL3 language generator. [13] extends LOREL's functionalities by representing changes in semistructured data for users to query.

WebSQL[14, 15] is a project at University of Toronto to develop a Web query facilitation language. It views the Web as a table of documents, in which URL, Title, Type, Last Modified Date are treated as columns. WebSQL extends standard SQL by adding information related to Web documents, such as URL and Title, as column names for queries. Some user-defined functions, such as "mentions", are supported for more fuzzy textual string matching. The query interface provided for WebSQL is form-based, as opposed to the visual query interface and query generator provided by WebDB: we feel it is difficult for end-users to pose these complex queries directly. Additionally, WebDB supports query relaxation. WebOQL[16] is an extension of WebSQL. WebOQL further supports reorganization of Web pages.

Strudel[17, 18, 19] is a Web-site management system. It applies familiar concepts from database management systems to the process of building Web sites. The Web site manager can separate the logical view of information at a Web site, the structure of that information in linked pages, and the graphical presentation of pages in HTML. First it defines the data available at that site as an integrated view from multiple (external) sources. Second, it defines the structure of the Web-site, as a view over the underlying information: different versions of the site can be defined by specifying multiple views.

UnQL[20, 21], developed at AT&T, has been specifically designed to query semistructured data. It has two levels: a declarative and a functional level. The declarative level consists of a Select-Where construct, much like the SQL Select-From-Where, with some additional OQL-like constructs to deal with nested objects, including regular expressions to access attributes which may be nested arbitrarily deep. The functional level consists of a language allowing the definition of mutually recursive functions, based on pattern matching. Substantial amount of work has been focused on query optimization[22, 23, 24]. Compared with UnQL, WebDB uses a language based on SQL3 to take advantages of both OO modeling and relational languages.

WebLog[25], developed at Concordia University, is a intended to be a more complete language to support both query and result rendering formatting. Due to the complexity of this language and lack of query interfaces, WebLog is difficult to use. No implementation of WebLog has been reported.

TSIMMIS[26] is a project at Stanford University to support query heterogeneous information resources. TSIMMIS is similar to WebLog, but also provides many pre-defined queries for information retrieval so that users need not pose complex queries directly. But, this also restricts searches using a limited set of queries.

W3QS (WWW Query System)[27] at Technion (Israel Institute of Technology) is a project to develop a high level SQL-like Web query language, W3QL, which views the Web as an ultra large database. W3QL is capable of addressing both structure and content. W3QS allows users to specify the starting page for the search, the search domain, and the file names using Perl regular expressions and file types (i.e. extensions gif, tiff, ps) for page contents. W3QS allows users to specify the depth of links for search. In comparison, WebDB also allows users to specify queries with arbitrary Web structures; it is not limited to one link-in or one link-out. Moreover, WebDB's query interface is more visual and user-friendly and WebDB also supports query relaxation.

SuperPages[28] discusses a loosely coupled architecture to construct the yellow pages service of

GTE SuperPages. It enables Web users to search through listings of 11 million businesses in over 17000 categories. It uses an Information Retrieval (IR) engine, Verity[29], to search through complex listing objects while the objects themselves are stored in an object database. The complex objects are modeled as a hierarchical structure with various SGML tags for the IR engine to perform query processing.

The Araneus Project[30] addresses the issue of views in the Web context. It introduces a set of languages for managing and restructuring data coming from the Web and presents a specific data model, the Araneus Data Model, to describe the scheme of Web hypertext. Based on the data model, it develops two languages, Ulixes and Penelope, to support a view definition process. Ulixes is used to build database views of the Web, which can then be analyzed and integrated using database techniques. Penelope is used to define derived Web hypertext from relational views, which can be used to generate hypertextual views over the Web. These languages have been used to develop a scheme for the DBLP Bibliography Server[31].

Similar to DBLP Bibliography Server, HyperFile[32] is a data and query model for hypertext documents. It introduces sophisticated modeling scheme and discusses its query processing technique. Compared with HyperFile, WebDB is a query system hypermedia documents on the Web and WebDB supports additional functionalities, such as a visual query interface, multimedia search capability, and query relaxation, to provide higher usability.

3 Modeling World-Wide Web

We view the Web as a collection of HTML documents connected by hyperlinks. To support more comprehensive and complex query functionalities, such as those described in Section 1, WebDB stores not only Web document level metadata, but also intra-document and inter-document information. In this section, we present our modeling scheme for the Web and Web pages followed by the operations on the data model.

3.1 Web Modeling

We view and model the Web as a labeled directed graph $G_{web} = (V_{web}, E_{web})$, where the vertices ($V \in V_{web}$) denote the pages and the edges ($E \in E_{web}$) denote the hyperlinks between these pages. The vertices are labeled by the URLs of the pages and other document level information, including title, URL, content length, data types, last modified date, and keywords. The edges are links from source pages to destination pages and are labeled by the *descriptive text or image*: anchors. Hence, if there is an edge $e_j \in E_{web}$, of the form $n_i \rightarrow n_k$, with an anchor $label(e_j)$, the page corresponding to vertex n_i has a link to the page corresponding to vertex n_k and the descriptive text associated with this link is the anchor $label(e_j)$.

Each page, $V \in V_{web}$, is modeled as a graph $G_{page} = (V_{page}, E_{page})$, where the vertices ($V \in V_{page}$) denote the content of the page and the directed edges ($E \in E_{page}$) denote the containment relationship. V_{page} includes a special vertex (root) v_0, which denotes the page itself; every other vertex is reachable from v_0. The vertices in this graph are labeled as *Table*, *Form*, *Image*, and *Link*. For *Table* and *Form*,

they may further contain *Table*, *Form*, *Image*, and *Link*. Therefore, only those vertices which are labeled as *Table*, *Form*, *Link*, or v_0 (i.e. page) can have outgoing edges.

The vertices of images have no outgoing edges to *Table*, *Form*, or other pages[2]. Each vertex, labeled as *Image*, is modeled as a tree $T_{image} = (V_{image}, E_{image})$. The vertices of this tree are image objects. The edges are the containment relationship. For more details about the image modeling, please refer to [1].

In the next subsection, we show how we map the above Web modeling to a data model for implementation on database systems.

3.2 Data Model

The design and implementation of WebDB is based on the object-relational concept; the Web is modeled using the object-oriented model and the query language is based on SQL3. The Web modeling in WebDB is illustrated in Figure 3 and is as follows:

- A Web document, *Doc*, is modeled as a compound object with a hierarchical structure. Document level information, such as title, URL, content length, file types, last modified date, collected date, and keywords, are the attributes of *Doc* object. These attributes are listed in Table 1 and examples of attribute values are given in Appendix C.

- Intra-document structures are modeled as sub-objects of *Doc*, including *Form*, *Image*, *Table*, and *Link*. The relationship between *Doc* and the sub-objects is *contains*. Sub-objects also have their own attributes and structures. The attributes for *Image* include *Metadata*, *Format*, *Caption*, and *Size*. The attribute for *Form* and *Table* are contents contained and captions.

  Since there may be forms, images, tables, and links within a form or a table, we further model its internal structure recursively as attributes: pointers to other subobjects *Form*, *Image*, *Table*, and *Link*. For example, if *Table A* contains *Table B* and *Image I*, *Table A*, *Table B*, and *Image I* are three separate subobjects and there are pointers from *Table A* to *Table B* and *Image I*. If there is no other subobject within *Table B*, the attributes *ContainedTable*, *ContainedImage*, *ContainedForm*, and *ContainedLink* have a *Null* value. This modeling scheme allows users to *explicitly* search internal structures of subobjects, such as "retrieve documents containing a table which has an image in the gif format within it". However, the users can also *implicitly* perform such search through a text-based search on *Table.Contents* as follows:

```
Select Doc D1
From WebDB
Where
and D1 contains Table T1
and T1.contents mentions "imgsrc*gif"
```

[2]The images mentioned here do not include images used as anchors.

Attribute Name	Data Type	Description
Doc.Title	List of Strings	Document title
Doc.URL	String	Document's universal resource locator
Doc.Keyword	Set of Strings	Keywords associated with the document
Doc.Length	Integer	Document size in bytes
Doc.Type	String	File MIME type
Doc.Last_modified_date	Date	Document last modified date
Doc.Collected_date	Date	Document collected date
Doc.Image	Sub-object of Image data type	Image contained in the document
Doc.Form	Sub-object of Form data type	Forms contained in the document
Doc.Table	Sub-object of Table data type	Tables contained in the documents
Doc.Link	Sub-object of Link data type	Hyperlink references in the document
Image.Metadata	Image Metadata	Visual characteristics for image matching
Image.Size	Integer	Image size in byte
Image.Format	String	Image format, such as gif, jpg, bmp, etc.
Image.Caption	List of Strings	Description of an image
Table.Contents	List of Strings	Contents within a table
Table.Caption	List of Strings	Description of a table
Table.ContainedImage	Pointer to Image data type	Images contained in a table
Table.ContainedTable	Pointer to Table data type	Tables within a table
Table.ContainedLink	Pointer to Link data type	Links within a table
Table.ContainedForm	Pointer to Form data type	Forms within a table
Form.Contents	List of Strings	Contents within a form
Form.Caption	List of Strings	Description of a form
Form.ContainedImage	Pointer to Image data type	Images within a form
Form.ContainedTable	Pointer to Table data type	Tables within a form
Form.ContainedForm	Pointer to Form data type	Forms within a form
Form.ContainedLink	Pointer to Link data type	Links within a form
Link.TexualAnchor	List of Strings	The label of links
Link.ImageAnchor	Pointer to Image data type	The image logo of links
Link.URL	String	URL of the document referenced

Table 1: Attribute Classification, Types, and Descriptions

We have formally defined a complete set of attributes of *Image*, *Form*, and *Table*. Since *Image.metadata*, *Form.Contents*, and *Table.Contents* are frequently used, these attribute names are default attributes for *Image*, *Form*, and *Table* respectively in our query statements for the simplicity reasons.

- Inter-document information is represented by *Link*, which is a sub-object of *Doc*. *Link* has three attributes: *URL* for the destination URL, *TextualAnchor*, and *ImageAnchor*[3]. They are used for modeling inter-document linkage information. Inter-document links are modeled through *Join* operations on *Link.URL* and *Doc.URL*. For *TextualAnchor*, WebDB supports text-based search, while image match operations are supported for *ImageAnchor*. For a given Web document which we view as the main page, there are three types of Web documents:
  - Link-in pages: The pages can reach the main pages.
  - Link-out pages: The pages can be reached from the main page.

[3]There are two types of anchors for users to click; namely, an image or a list of strings.

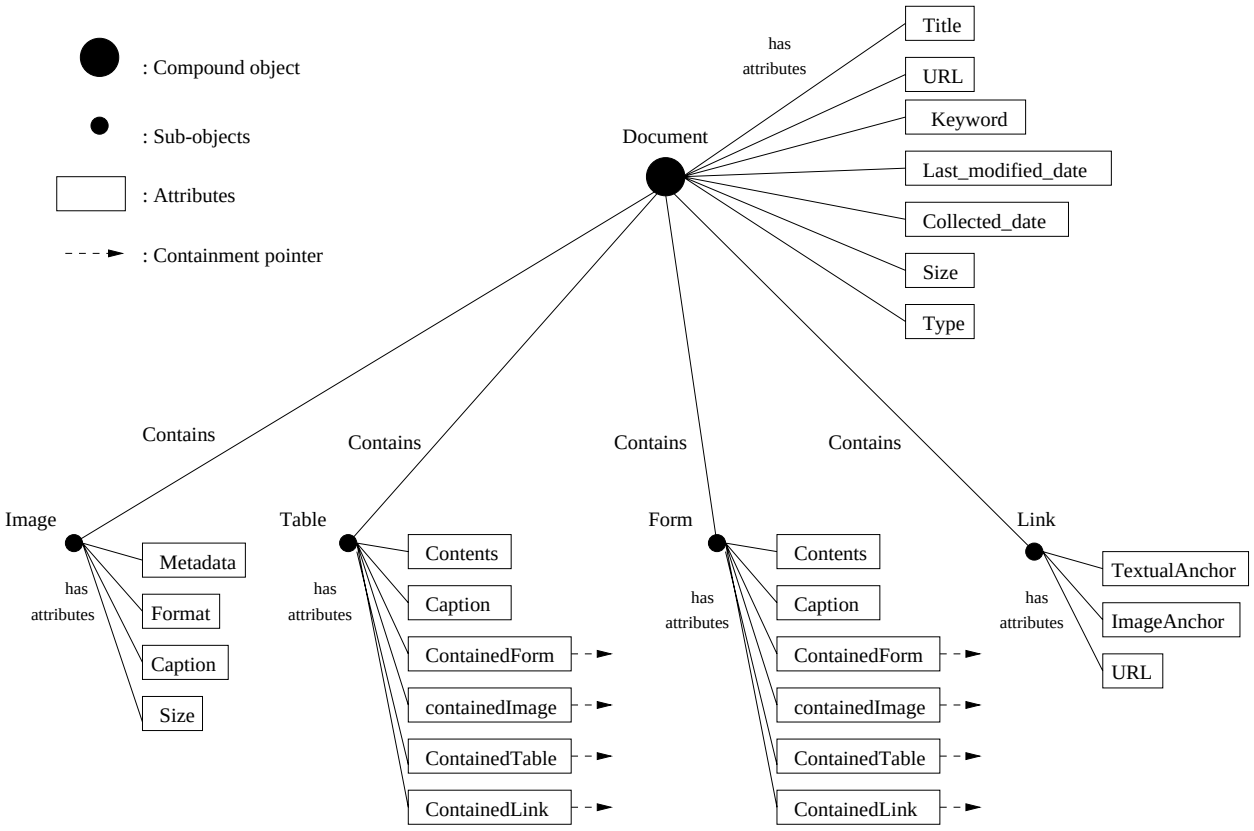

Figure 2: Data Model for Hierarchical Multimedia Web Page

– Unrelated pages: These pages can not be reached from the main pages and these pages can not reach the main page neither.

We illustrate the relationships between the main page and its link-in/link-out pages in Figure 3.

These attributes of *Image*, *Form*, *Table*, and *Link* are shown in Table 1 and examples of attribute values are also given in Appendix C. Note that `Table.Contents`, `Form.Contents`, `Link.Anchor`, and `Doc.Title` are of type *list of strings*, while `Doc.keyword` is of type *Set of strings*. In a list of strings, the sequence of string appearance is stored, but the sequence of string appearance in a set of strings is not stored. The reasons why the `Doc.keyword` is of type *Set of strings* are as follows:

- Keywords are associated with documents, not with each other;
- Many keywords appear multiple times in a document, but their sequence may vary;
- Many keywords are derived from the document contents. As a result, keywords used for indexing may not be the same as the terms used in the document.

An example of HTML documents and its parsing results are given in Appendix B and Appendix C. The statistics of Web document parsing is provided in Appendix D.

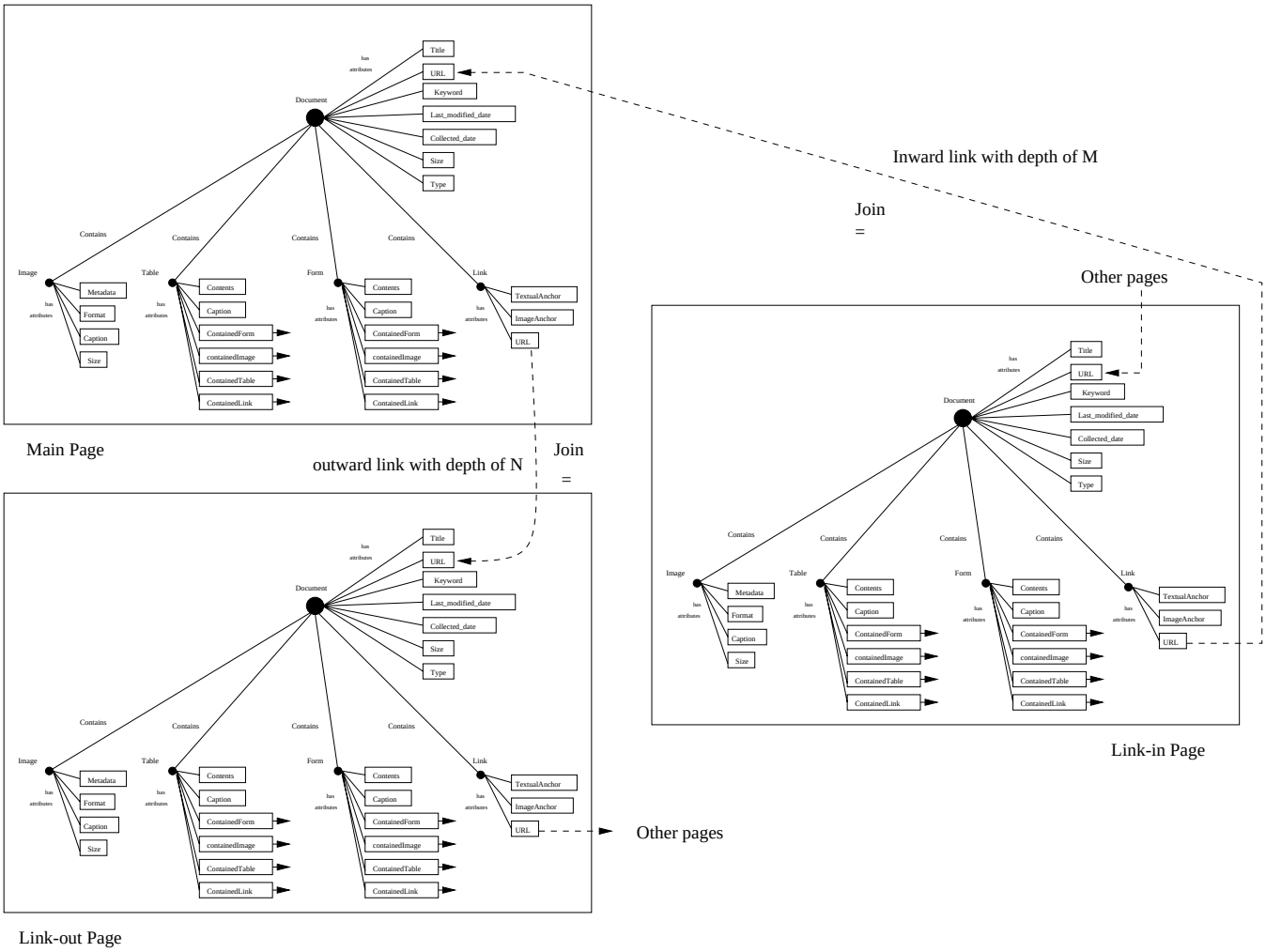

Figure 3: Web Modeling in WebDB

3.3 Operations on Web Document Hierarchical Structure and Links

Through this modeling scheme presented in the previous subsection, the Web can be represented as shown in Figure 3. In Section 3.1, we define the Web as a labeled directed graph $G_{web} = (V_{web}, E_{web})$, where the vertices (V) denote the pages and the edges (E) denotes the links. Using the model scheme in Figure 3, the vertex (V) is represented as a compound object with a hierarchical structure for modeling its internal structure.

The edge (E) is modeled *implicitly* through join operations on *Doc1.Link.URL* and *Doc2.URL*, where *depth* is a parameter for join operations, defining the number of join operations to be performed recursively. Given *Doc1* and *Doc2*, which correspond to two vertices on $G_{web} = (V_{web}, E_{web})$, the *depth* from node *Doc1* to *Doc2* is the shortest distance from *Doc1* to *Doc2* through links.

SQL is suitable to manipulate only traditional textual or numeric data stored as tables. SQL3 extends SQL to support user defined data types and user defined functions to manipulate user defined data types. By viewing objects as entities and links as relations, we can map the modeling representation in Figure 3 to the Entity-Relational (ER) model for designing a SQL-based query language. Since we

model Web documents as compound objects, we need to extend the capabilities of the traditional SQL with the following functionalities:

- **Traversal of the intra-document structure:** The intra-document structure traversal is by way of the predicate *contains*. For example, after the statement "Document *Doc* contains Tables T" is issued, the system can access sub-object, represented by the variable T, which is contained by the document object *Doc*. In our system, "Document *Doc*" and "Tables T" in the statement "Document *Doc* contains Tables T" are actually viewed as variable declarations. An example statement for traversal intra-subobject structures, such as a link within a table, is as follows:

  Select Doc D1
  From WebDB
  Where
  and D1 contains Table T1
  and T1 contains Link L1
  and L1.TexualAnchor mentions "NEC"

  Note that *contains* is an overloaded operator. The statement $D1$ *contains Table* $T1$, the system defines $T1$ is a table subobject of document $D1$, while the statement $T1 contains Link L1$ defines the link subobject $L1$ is pointed by the attribute *ContainedLink* of the table subobject $T1$.

- **Traversal of the Web (inter-document links):** The Web structure, on the other hand, is modeled through hyperlinks and depth restriction information. Traversal of the Web from page *Doc_x* to page *Doc_y* through a link with depth of 2 is by way of the following join operations:

  $Doc\_x.Link.URL = Doc\_y.URL$
  or
  ($Doc\_x.Link.URL = Doc\_z.URL$ and $Doc\_z.Link.URL = Doc\_y.URL$)

- **Query relaxation:** WebDB supports keyword-based search in conjunction with both syntactic similarities (e.g. term cooccurring frequency) and semantic similarities (i.e. semantic meaning).

- **Similarity-based image matching:** WebDB provides an *i-like* (image like) predicate to perform image matching. The image matching functionality is carried out by an image database SEMCOG and users can specify image related queries using the IFQ visual query interface. Detailed information of SEMCOG and IFQ is available in [1].

- **Searches on the contents within forms and tables:** WebDB provides a *mentions* predicate for searching information within forms and tables in a document. Given that the contents of forms and tables are of type list of strings, users can specify the sequence of appearance for multiple terms to explore row and column structures in tables although row and column structures are not *explicitly* defined here. For forms, tables, images, and links within subobjects of *Form* and *Table* data types, the users can *explicitly* (i.e. using pointers) or *implicitly* (i.e. using string matching) pose queries to explore.

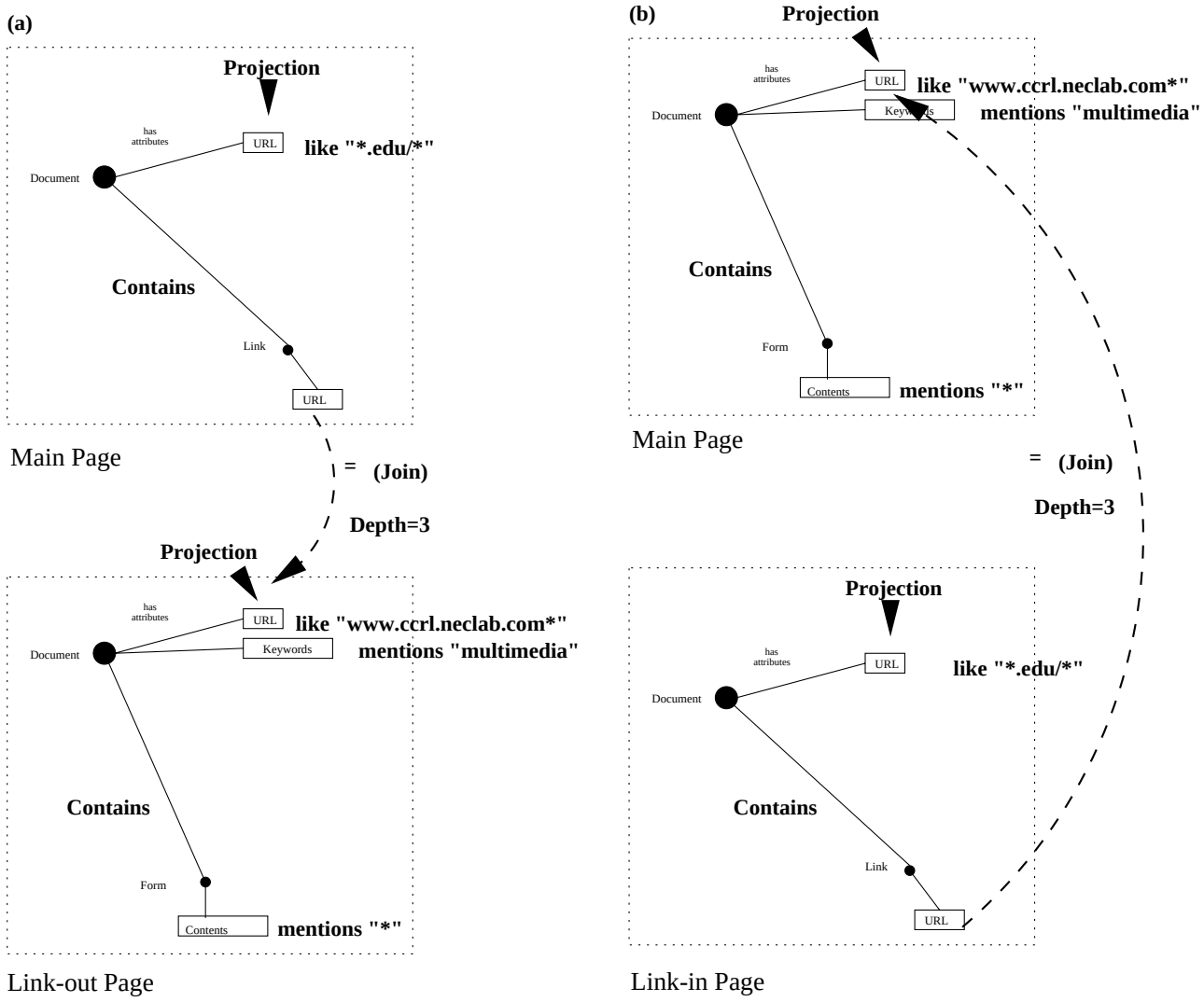

Figure 4: Query Modeling

3.4 Query Modeling

We model a Web query as a partially-labeled subgraph, $G_{query} = (V_{query}, E_{query})$. The problem of finding an answer to a user's query can then be regarded as finding a subgraph of G_{web}, $G_{sub} = (V_{sub}, E_{sub})$, such that G_{sub} is isomorphic to G_{query} and the specified labels in G_{query} matches those labels on G_{sub}.

For example, the query "list all Web pages in the domain of `.edu` which have links to pages in NEC Web site, `www.ccrl.sj.nec.com`, containing a keyword "multimedia" within depth of 3" can be posed as follows (formal definition will be given in next section):

Select Doc D1
From WebDB
Where D1 *contains* Link L1
  and L1.URL = Doc D2.URL Depth 3
  and D2.URL *like* "www.ccrl.sj.nec.com*"
  and D2.Keyword *mentions* "multimedia"

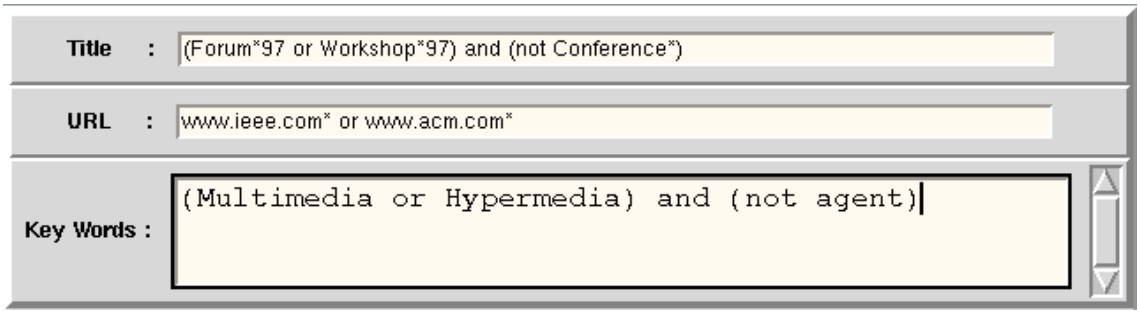

Figure 5: User Query Criteria as *Boolean Expression of Strings*

and D1.URL *like* "*.edu/*"

Based on the data model shown in Figure 2, we can model this query as shown in Figure 4(a). The query statement is a subgraph of the Web data model in Figure 3 with projection operators to project values of attributes `URL`. In Figure 4(b), we show another way to model the query, where the NEC Web site is the main page and the Web pages in the domain of `.edu` are link-in pages. Since the relationship between the main window, link-in window, and link-out window are relative, and not *absolute*, the results for these two types of query models are equivalent. We next give the formal definition of WQL.

4 Query Language Design - WQL

WQL (Web Query Language) is the language we developed for Web query specification. WQL is a descriptive language based on SQL3 to allow users to access and manipulate document structures and attribute values of objects and their sub-objects.

WQL matches the Web model described in Section 3. Being a descriptive language, WQL's syntax is model-dependent, but not schema-dependent. As a result, WQL simplifies the process of query generating; i.e. the translation of visual query specification into WQL query statements, as described in Section 5.

4.1 Query Definitions

In Table 1, we give the basic type definitions: *String*, *List of Strings*, *Set of Strings*, *Date*, and *Integer*. In addition, we introduce a new data type, *Boolean Expression of Strings*, which is a set of string combination using AND, OR, and NOT. Note that the data type *String* is equivalent to type *Boolean expression of Strings* with no AND, OR, and NOT.

Boolean Expression of Strings includes the criteria specification for *Doc.URL*, *Doc.Title*, *Doc.Keyword*, *Table.Contents*, and *form.Contents*. Examples are shown in Figure 5. An example of its usefulness is as follows: A user wants to retrieve documents containing keywords *Multimedia* or *Hypermedia*, but not *Agent*. This query may be posed as follows:

Doc.Keyword *mentions* "Multimedia"
or Doc.Keyword *mentions* "Hypermedia"
and Not(Doc.Keyword *mentions* "Agent")

In order to simplify the users' specification process for complex queries of Boolean expression, most query interfaces let users pose *only* Boolean expression of Strings as shown in Figure 5. For the above query criteria, WQL allows users to pose all expressions together using a simplified format as follows:

Doc.Keyword *mentions* "((*Multimedia* or *Hypermedia*) and (not *Agent*))"

The query criteria is then converted back into the correct Boolean expression format for internal query processing. However, a query optimizer would more likely process the query criteria based on its simplified form since an "intelligent" query processor would scan through the table only once to check all four conditions, rather than scan through the table four times.

Given two arguments, $\mathtt{S}_1$ and $\mathtt{S}_2$, WQL allows the following operations of comparisons:

- $\mathtt{S}_1$ *like* $\mathtt{S}_2$ where *like* is a popular vendor implemented SQL predicate for string match. $\mathtt{S}_1$ is of type *string*, such as *URL*, and $\mathtt{S}_2$ is *Boolean expression of strings*. In the case that $\mathtt{S}_2$ is *Boolean expression of strings* with AND, OR, and NOT, queries are translated into Boolean combinations of multiple statements and each statement is in the form of $\mathtt{S}_1$ *like* $\mathtt{S}_{2i}$, where $\mathtt{S}_1$ and $\mathtt{S}_{2i}$ are of type of *string*.

- $\mathtt{S}_1$ *mentions* $\mathtt{S}_2$: The functionality of the predicate *mentions* is to test if $\mathtt{S}_2$ is sub-set of $\mathtt{S}_1$ where $\mathtt{S}_1$ is of type *set of strings* or *list of strings*, such as *Doc.Title*, *Doc.Keyword*, *Table.Contents*, and *form.Contents*. $\mathtt{S}_2$ can be *string* or *Boolean expression of strings*. In the case of $\mathtt{S}_2$ is of *Boolean expression of strings*, queries are translated into Boolean combinations of multiple statements and each statement is in the form of $\mathtt{S}_1$ *mentions* $\mathtt{S}_{2i}$, where $\mathtt{S}_1$ and $\mathtt{S}_{2i}$ are of type of *string*.

With WQL, users can explore internal structures of forms and tables, such as rows, columns, or containment other media and data types. For example, if a user wants to retrieve pages containing images in the gif format within a table, the query can be specified as:

Select Doc D
From WebDB
Where D *contains Table* T
  and T *mentions* "img*src*gif"

WebDB also support the following two functions to allow users to specify a query relaxation scheme:

- *s_like(S, N)* (Semantic Like) where S is a string and N is the number of semantically similar terms requested: The output of this function is of type of *list of strings*. For example, a function call *s_like(car, 3)* may return sedan, coupe, and truck. This function is supported through an on-line lexicon dictionary, Wordnet[33].

- *Cooccurrence(S, N)* where S is a string and N is the number of terms requested which are related based on their co-occurrence frequency in the same document: The output of this function is of type of *list of strings*. For example, a function call *s_like("Michael Jordan", 3)* may return "NBA", "Chicago Bulls", and "Scotty Pippen". This function is supported through pre-built indices for co-occurring terms in documents.

4.2 Selection Criteria

Each WQL query is in the following format:

```
Select {projected attributes | [aggregation functions(attributes)] }
From   { database names }
Where  { selection criteria }
[Group by] { grouping criteria }
[Order by] { ordering criteria }
```

The first line of the this format describes the attributes to be projected or aggregated values. The second line describes the database to be used for search. Note that users do not need to specify table names, which are hidden from the users. In WebDB, users are not required to know schema design, but need to specify the database space for search. The third line provides the selection criteria, as described above. The last two (and optional) lines describe the grouping and ordering criteria for the aggregation operators. The complete language syntax is given in Appendix A. We categorize query selection criteria based on the types of information involved in as follows:

- *Page level metadata selection criteria*: This type of search criteria is based Page level metadata, including *Title, URL, Keyword, Last_modified_date, Collected_date, Size,* and *Type*, associated with a given document. An examples query of this type is as follows:

  Select Doc D1.URL, D1.Title, D1.Last_modified_date
  From WebDB
  Where D1.Title *mentions* "forum"
  and D1.URL *like* "(www.ieee.com* or www.acm.com*)"
  and D1.Last_modified_date >= "07/01/1997"
  Order by D1.Last_modified_date

  This query retrieves all documents in the IEEE Web site or the ACM Web site whose titles contain "forum" such that these documents are last modified after July 1, 1997. This query also requests the system to output (project or compute aggregated values) *D1.URL* (default projected attribute), *D1.Title*, and *Last_modified_date* and to sort the output by *Last_modified_date*.

- *Intra-document structure and metadata selection criteria*: This type of search criteria is based on the attributes of subobjects *Form*, *Link*, and *Table* data types. An example query of this type can be as follows:

  Select Doc D1
  From WebDB
  Where D1.Title *mentions* "membership"
  and D1.URL *like* "www.ieee.com*"
  and D1 *contains* Form F1

and F1 *mentions* "*"

This query retrieve all documents in the IEEE Web site containing forms and keyword "membership". Note that *URL* is a default projected attribute if no attribute of document objects is specified. Note that the last statement

F1 *mentions* "*"

is a simplified version of the statement

F1.Contents *mentions* "*"

- *Inter-document linkage selection criteria*: This type of selection criteria requires *Doc.URL* (information at the document level), *Link.URL*, and *Link.Anchor* since inter-document linkage is modeled through *join* operations and string comparsions on the attribute *Anchor*. An example query for finding out the most popular Web search engines among Yahoo!, InfoSeek, and AltaVista used at the NEC Web site can be posed as follow:

Select Doc D2.URL, count(*)
From WebDB
  Where Doc D1.URL *like* "www.ccrl.neclab.com*"
  and D1 *contains* Link L1
  and L1.URL = D2.URL
  and D2.URL *like* "(www.yahoo.com/index.html or www.altavista.com/index.html
      or www.infoseek.com/index.html)"
  Group by D2.URL
  Order by 2

Note that this query may alternatively be posed as follows:

Select Link L1.URL, count(*)
From WebDB
  Where Doc D1.URL *like* "www.ccrl.neclab.com*"
  and D1 *contains* L1
  and L1.URL *like* "(www.yahoo.com/index.html or www.altavista.com/index.html
      or www.infoseek.com/index.html)"
  Group by L1.URL
  Order by 2

One difference between these two queries is that the first query *implicitly* requests the system to validate the existence of www.yahoo.com/index.html, www.altavista.com/index.html, and www.infoseek.com/ index.html, while the second query does not; the system only examines if the URLs of these Web search engines are specified in the documents at NEC Web sites.

5 WebIFQ as Query and Browsing Interfaces

Most existing systems focus on query capability or query optimization for semistructured data. We feel that most of their query languages, based on either datalog, first logic, SQL, or SQL3, are too complicated for non-system designers to use directly. As a result, usability of these systems are low.

WebDB features a visual query interface, WebIFQ, to assist users in specifying queries, which can be complicated. With WebIFQ, icons and menus are used in a drag-and-drop fashion to specify document level selection criteria, linkage conditions, and document information associated with link-in or link-out pages. The corresponding query statements are generated by the system automatically. With WebIFQ, users of WebDB are not required to be aware of the complex query syntax and the underlying schema.

We now present some window dumps to demonstrate how users specify Web queries in our system.

5.1 Query Specification

In Figures 6 and 7, we show a query specification for retrieving documents in the NBA Web with the keyword *Bulls* or top 5 related terms by co-occurrence statistics, a table containing a word "statistics" and a link to other documents in the NBA Web site with a anchor like "NBA*50" with depth of 3.

```
Select Doc D1
From WebDB
Where
 D1.URL like "www.nba.com*"
and D1.Keyword mentions "Bulls or cooccurrence("bulls",5)"
and D1 contains Image I1
and D1 contains Table T1
and T1 mentions "statistics"
and D1 contains Link L1
and L1.URL = Doc D2.URL
and D2.URL like "www.nba.com*"
```

There are three types of windows, namely, main, link-in, and link-out windows. WebIFQ allows users to switch between these windows to specify query criteria associated with each window by clicking the `Main`, `Link-in`, and `Link-out` buttons at the top of `Search Specification Window`. When users specify query criteria in one window, the system shrinks other windows but display their summarized query criteria.

Figure 6 shows the query specifications from the main window view while the link-in window is shrunk: the user specifies the search criteria for `URL`, `Keywords`, and `Form`. After the user clicks the `Link-in` button, `Search Specification Window` switches from the main window view (Figure 7) to the link-in window view (Figure 7). Figure 7 shows the query specifications from the link-in window view while the main window is shrunk: the user specifies the search criteria `URL` of the link-in window.

In Figure 6, there are 2 buttons, `P` and `A`, next to the attributes. Users can click on the `P` buttons to specify *Projection* operations on particular attributes. The projection on the URL for the main page

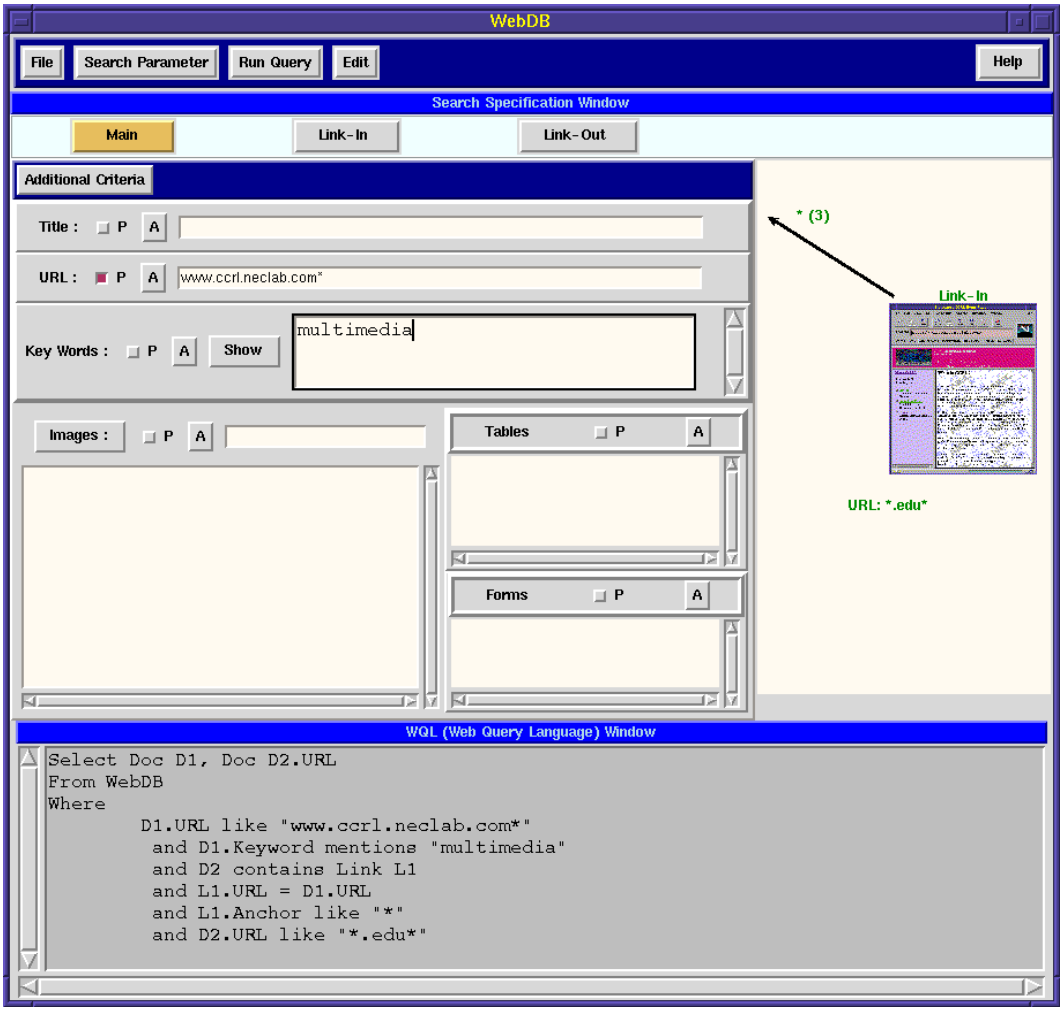

Figure 6: Query Specification Using WebIFQ (Main Window View)

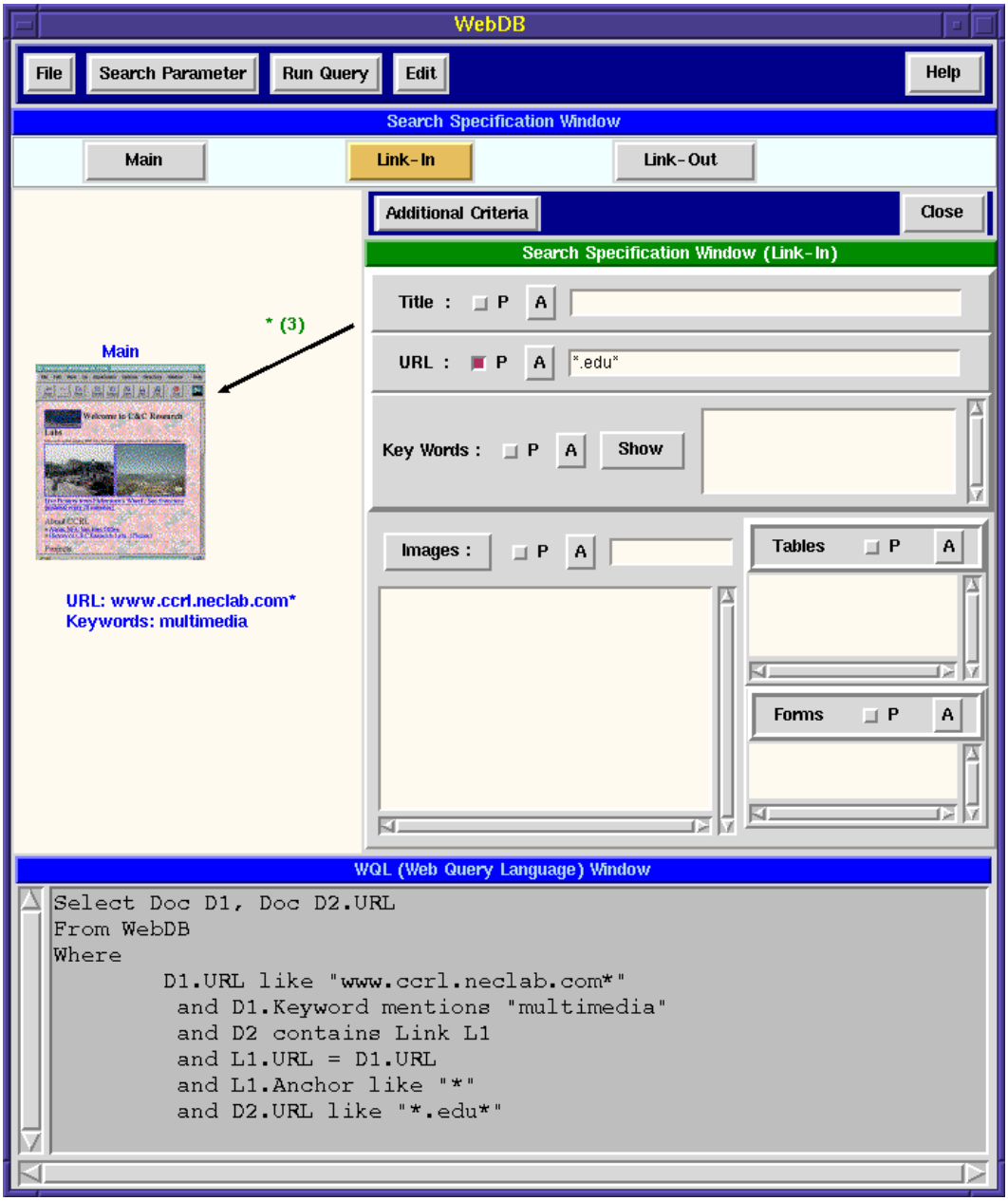

Figure 7: Query Specification Using WebIFQ (Link-in Window View)

is default as indicated by the highlighted `P` button.

Users can click on the `A` button to apply *Aggregation* functions to particular attributes. Currently, WebDB supports 7 aggregation functions: `Sum, Count, Avg, Stddev, Variance, Max,` and `Min`. Since not all aggregation functions can be applied to a given attribute, when users click on the `A` button, a pull-down menu appears to show the users the available aggregation functions which can be applied to the attribute. For example, all 7 aggregation functions can be applied to *Doc.Length*, *Doc.Last_modified_date*, and *Doc.Collected_date*, but only `Count` can be applied to attributes associated with strings, such as *Doc.URL*, *Doc.Title*, *Doc.Keyword*, *Table.Contents*, and *form.Contents*. If projection operators or aggregation functions are applied, the corresponding buttons are highlighted.

The interface visualize users' query specifications as follows:

- Linkage query criteria: In Figure 8, the system visualize the linkage relationship between the main window and the link-out window as well as the anchor and depth conditions.
- Document level and intra-document query criteria: There are three types of windows, namely, main, link-in, and link-out windows. Users can select any window to specify search criteria, while search criteria of other windows are summarized, as shown below the link-out window in Figure 8.

5.2 Browsing Query Results

After the query is processed, the result is displayed in an external result window. When users click on any candidate URL, the query interface interacts with existing Web browsers, such as Netscape Navigator, through remote procedure calls to display the selected home page. Users can also select the "automated display" mode for "slide-show-like" display of all or selected matching pages or simply store the result as a bookmark for further usage. In Figure 8, the user clicks on several documents (highlighted in this picture) that he/she is interested in browsing.

6 Conclusion

We views the Web as a database with semistructured Web pages. To support more comprehensive query functionalities, not only document level metadata but also intra-document structures and inter-document linkage information are collected in our system. This paper presents WebDB, a Web query system, and its modeling, language, and implementation. WebDB is based on object-relational concepts and features a query language based on SQL3. It provides access to all of these three types of information in a uniform manner. In addition, WebDB provides query facilities that traditional databases provide, including selection, projection, and organization (through aggregate functions) capabilities. We have demonstrated many useful applications of this system. WebDB provide services beyond existing Web search engines can. One significant contribution of WebDB is that it supports a visual query interface and a query generator for complex Web queries. WebIFQ makes WebDB more practical to use compared with other existing systems.

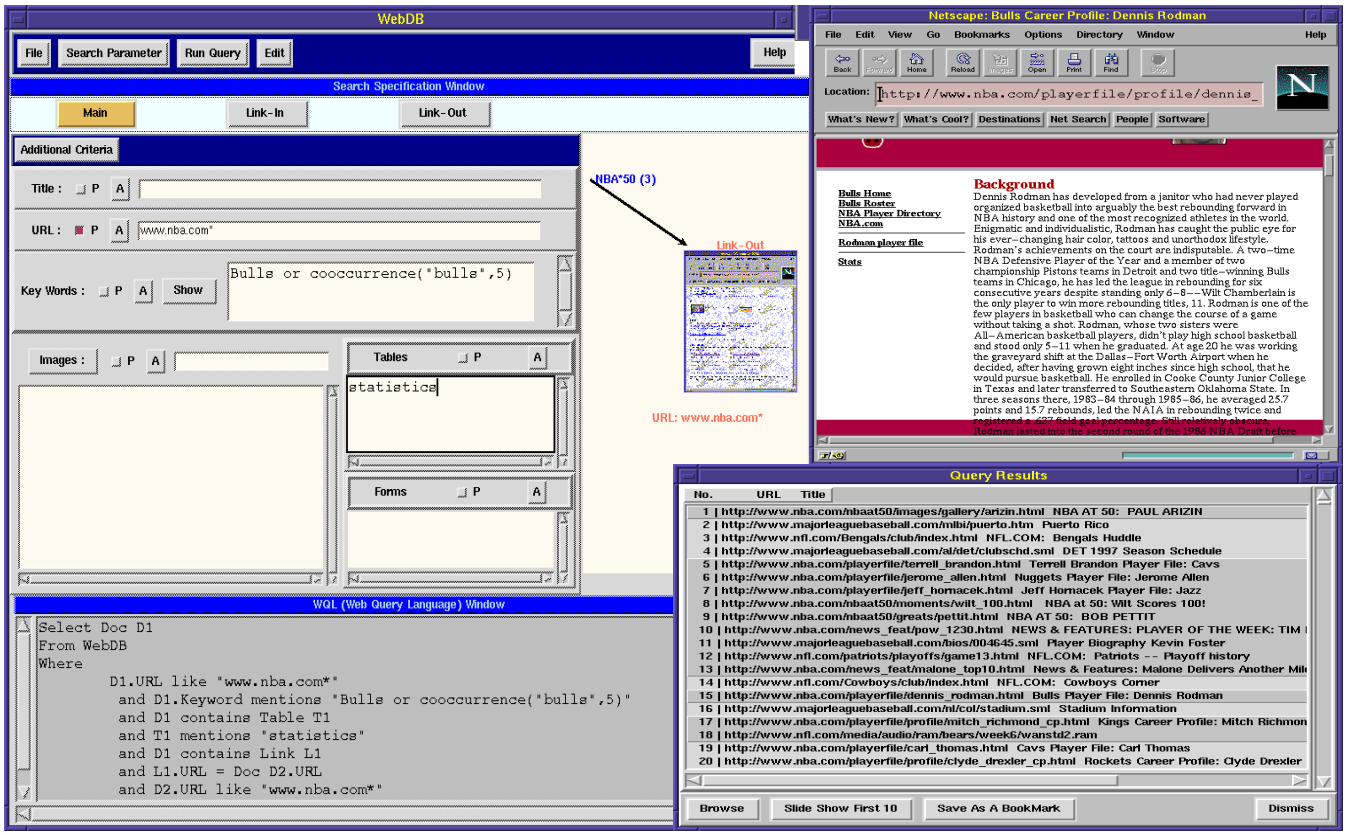

Figure 8: Example query Using WebDB Query and Browsing Interfaces

References

[1] Wen-Syan Li and K. Selçuk Candan. SEMCOG: A Hybrid Object-based Image Database System and Its Modeling, Language, and Query Processing. In *Proceedings of the 14th International Conference on Data Engineering*, Orlando, Florida, USA, February 1998.

[2] Wen-Syan Li, K. Selçuk Candan, K. Hirata, and Yoshi Hara. IFQ: A Visual Query Interface and Query Generator for Object-based Media Retrieval. In *Proceedings of the 1997 IEEE Multimedia Computing and Systems Conference*, pages 353–361, Ottawa, Ontario, Canada, June 1997.

[3] AltaVista Technology, Inc. of California. *Information available at http://www.altavista.com/.*

[4] Infoseek Corporation. *Information available at http://www.infoseek.com/.*

[5] Yahoo Communications Corporation. *Information available at http://www.yahoo.com/.*

[6] Excite Inc. *Information available at http://www.excite.com/.*

[7] Luis Gravano and Hector Garcia-Molina. Merging Ranks from Heterogeneous Internet Sources. In To appear in *Proceedings of the 23th International Conference on Very Large Data Bases*, Athens, Greece, August 12-15 1997. VLDB.

[8] S. Adali, C. Bufi, and Y. Temtanapat. Integrated Search Engine. In *Proceedings of the 1997 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX-97)*, Newport Beach, California, U.S.A, November 1997. IEEE.

[9] Yahoo Communications Corporation. *Information available at http://isurf.yahoo.com/.*

[10] C. Frankel and M. Swain and V. Athitsos. WebSeer: An Image Search Engine for the World Wide Web. Technical Report TR-96-14, University of Chicago, Chicago, IL, USA, July 1996.

[11] S. Abiteboul and D. Quass and J. McHugh and J. Widom and J. Wiener. The Lorel Query Language for Semistructured Data. *Journal on Digital Libraries*, 1(1):68–88, 1997.

[12] R. Goldman and J. Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In *Proceedings of the 23th International Conference on Very Large Data Bases*, Athens, Greece, August 1997. VLDB.

[13] S.S. Chawathe and S. Abiteboul and J. Widom. Representing and Querying Changes in Semistructured Data. In *Proceedings of the 14th International Conference on Data Engineering*, Orlando, Florida, USA, February 1998.

[14] Gustavo Arocena, Alberto Mendelzon, and George Mihaila. Applications of a Web Query Language. In *Proceedings of the 6th Int'l. WWW Conference*, Santa Clara, California, USA, April 1997.

[15] Alberto Mendelzon, George Mihaila, and Tova Milo. Querying the World Wide Web. *Journal on Digital Libraries*, 1(1):54–67, 1997.

[16] G.O. Arocena and A.O. Mendelzon. WebOQL: Restructuring Documents, Databases, and Webs. In *Proceedings of the 14th International Conference on Data Engineering*, Orlando, Florida, USA, February 1998.

[17] Mary Fernandez and Dana Florescu and Alon Levy and Dan Suciu. A Query Language and Processor for a Web-site Management System. In *Proceedings of the Workshop on Management of Semi-structured Data*, Tucson, Arizona, USA, May 1997.

[18] Mary Fernandez, Dana Florescu, Alon Levy, and Dan Suciu. A query language for a web-site management system. *SIGMOD Record*, 26, 1987.

[19] M. Fernndez and D. Suciu. STRUDEL - a Web-Site Management System. In *Proceedings of 1997 ACM SIGMOD Conference*, Tucson, Arizona, USA, May 1997.

[20] Peter Buneman and Susan Davidson and Gerd Hillebr and Dan Suciu. A Query Language and Optimization Techniques for Unstructured Data. In *Proceedings of the 1996 ACM SIGMOD Conference*, Montreal, Canada, June 1996.

[21] Peter Buneman and Susan Davidson and Mary Fernandez and Dan Suciu. Adding structure to unstructured data. In *Proceedings of the 1997 ICDT Conference*, 1997.

[22] Dan Suciu. Query decomposition for unstructured query languages. In *Proceedings of the 22nd VLDB Conference*, Bombay, India, September 1996. VLDB.

[23] Dan Suciu. Implementation and Analysis of a Parallel Collection Query Language. In *Proceedings of the 22nd VLDB Conference*, Bombay, India, September 1996. VLDB.

[24] M. Fernndez and D. Suciu. Optimizing Regular Path Expressions Using Graph Schemas. In *Proceedings of the 14th International Conference on Data Engineering*, Orlando, Florida, USA, February 1998.

[25] Laks V.S. Lakshmanan and Fereidoon Sadri and Iyer N. Subramanian. A Declarative Language for Querying and Restructuring the World-Wide-Web. In *Proceedings of 1996 IEEE RIDE- NDS*, New Orleans, USA, February 1996.

[26] J. Hammer, H. Garcia-Molina, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. Information translation, mediation, and mosaic-based browsing in the tsimmis system. In *Proceedings of the 1995 ACM SIGMOD Conference*, San Jose, California, May23-25 1995.

[27] David Konopnicki and Oded Shmueli. W3QS: A Query System for the World-Wide Web. In *Proceedings of the 21th International Conference on Very Large Data Bases*. VLDB, 1995.

[28] Steven D. Whitehead, Himanshu Sinha, and Michael Murph. GTE SuperPages: Using IR Techniques for Searching Complex Objects. In *Proceedings of the 23th International Conference on Very Large Data Bases*, Athens, Greece, August 1997. VLDB.

[29] Verity, Inc. *Information available at http://www.verity.com/.*

[30] Paolo Atzeni, Giansalvatore Mecca, and Paolo Merialdo. To Weave the Web. In *Proceedings of the 23th International Conference on Very Large Data Bases*, Athens, Greece, August 1997. VLDB.

[31] Databases & Logic Programming Bibliography Server. *Information available at http://sunsite.informatik.rwth-aachen.de/dblp/db/.*

[32] Chris Clifton, Hector Garcia-Molina, and David Bloom. Hyperfile: A data and query model for documents. *VLDB Journal*, 4(1), March 1995.

[33] G. A. Miller. WordNet: A Lexical Databases for English. *Communications of the ACM*, pages 39–41, November 1995.

Appendix A: Query Language Syntax in BNF

query ← *select* output_description *from* database_name *where* query_description |
select output_description *from* database_name *where* query_description group_by_description |
select output_description *from* database_name *where* query_description order_by_description |
select output_description *from* database_name *where* query_description group_by_description order_by_description

query_description ← atomic_query_description| composite_query_description

composite_query_description ← (query_description *and* query_description) |
(query_description *or* query_description) |
not(query_description) |
(query_description)

page_attributename ← *title*| *keyword_list*| *url*| *type*| *size*| *last_modified_date*| *collection_date*

numeric_attributename ← *size*| *last_modified_date*| *collection_date*

output_description ← output_term | output_term , output_term

output_term ← output_document_term | output_image_term|
output_table_term | output_form_term|
output_link_term

output_document_term ← document_variable.page_attributename |
sorted document_variable.page_attributename |
max document_variable.numeric_attributename |
min document_variable.numeric_attributename |
avg document_variable.numeric_attributename |
cv document_variable.numeric_attributename |
std document_variable.numeric_attributename

output_image_term ← image_variable |
semantic_variable |
dual_variable.*semantics* |
dual_variable.*image* |
sorted semantic_variable |
sorted dual_variable.*semantics*

output_table_term ← table_variable.*contents* |
table_variable.*caption* |
table_related_variable

output_form_term ← form_variable.*contents* |
form_variable.*caption* |
form_related_variable

output_link_term ← link_variable.*url* |
link_variable.*textualanchor*|
link_variable.*imageanchor*

```
atomic_query_description ← page_based_query_description|
                           web_based_query_description|
                           media_based_query_description

page_based_query_description ← title_query| keyword_query| type_query| size_query| date_query|
                               containment_query

title_query ← title_term mentions title_term
keyword_query ← keyword_term mentions keyword|
                keyword_term set_operator keyword_term
type_query ← type_term.type = type_term
size_query ← size_term num_operator size_term
date_query ← date_term date_operator date_term

title_term ← document_variable.title| title
keyword_term ← document_variable.keyword_list
type_term ← document_variable.type| type
size_term ← document_variable.size| size
date_term ← document_variable.last_modified_date|
            document_variable.collection_date | date

containment_query ← document_variable contains link link_variable|
                    document_variable contains table table_variable|
                    document_variable contains form form_variable|
                    document_variable contains image image_variable

media_based_query_description ← image_query| table_query| form_query

image_query ← nested_image_query|
              image_variable.caption mentions string

table_query ← table_variable.contents mentions string|
              table_variable.caption mentions string|
              nested_table_query
form_query ← form_variable.contents mentions string|
             form_variable.caption mentions string|
             nested_form_query

web_based_query_description ← url_descriptor = url_descriptor Depth number |
                              link_variable.textualanchor mentions string|
                              url_descriptor mentions string

url_descriptor ← document_variable.url| link_variable.url
set_operator ← ⊂ | ⊆ | ⊃ | ⊇ | =
num_operator ← < | > | ≤ | ≥ | =
date_operator ← before | after | = | within_operator
within_operator ← within numeric

group_by_description ← group_by attribute_list
attribute_list ← document_variable.page_attributename |
                 document_variable.page_attributename , attribute_list
title ← [ term_list ]
caption ← [ term_list]
contents ← [ term_list]
keyword_list ← [ term_list ]
database_name ← string
```

keyword ← string

url ← string

type ← string

size ← numeric

last_modified_date ← date

collection_date ← date

term_list ← string | string, term_list

nested_image_query ← image_query_description

image_query_description ← atomic_image_query_description|
(image_query_description *and* image_query_description)

atomic_image_query_description ← semantics_based_image_query_description|
scene_based_image_query_description |
cognition_based_image_query_description

semantics_based_image_query_description ← (entity_description *is* entity_description) |
(entity_description *is_a* entity_description) |
(entity_description *s_like* entity_description)

scene_based_image_query_description ← (sd_entity_description *to the right of* sd_entity_description
in image_entity_description) |
(sd_entity_description *to the left of* sd_entity_description
in image_entity_description) |
(sd_entity_description *above of* sd_entity_description
in image_entity_description) |
(sd_entity_description *below of* sd_entity_description
in image_entity_description)

cognition_based_query_description ← (id_entity_description *i_like* id_entity_description) |
(image_entity_description *contains* dual_entity_description)

entity_description ← semantic_entity_description | dual_entity_description |
image_entity_description

sd_entity_description ← semantic_entity_description | dual_entity_description

id_entity_description ← image_entity_description | dual_entity_description

semantic_entity_description ← semantic_constant |
semantic_variable

dual_entity_description ← dual_variable

image_entity_description ← link_variable.*imageanchor* | image_constant |
image_variable

semantic_constant ← $\underline{\text{term}_1}$ | $\underline{\text{term}_2}$

image_constant ← $\underline{\text{image}_1}$ | $\underline{\text{image}_2}$

semantic_variable ← $\underline{\text{A}}$ | $\underline{\text{B}}$ | ...

dual_variable ← $\underline{\text{A}}$ | $\underline{\text{B}}$ | ...

image_variable ← $\underline{\text{A}}$ | $\underline{\text{B}}$ | ...

```
nested_table_query                  ←  table_query_description
table_query_description             ←  atomic_table_query_description|
                                       (table_query_description and table_query_description)

atomic_table_query_description      ←  base_table_query_description|
                                       containment_table_query_description

containment_table_query_description ←  table_variable contains column column_variable|
                                       table_variable contains column[integer] column_variable|
                                       table_variable contains row row_variable|
                                       table_variable contains row[integer] row_variable|
                                       table_variable contains cell cell_variable|
                                       table_variable contains cell [integer,integer ] cell_variable

base_table_query_description        ←  table_related_variable mentions keyword |
                                       table_related_variable contains link link_variable|
                                       table_related_variable contains table table_variable|
                                       table_related_variable contains form form_variable|
                                       table_related_variable contains image image_variable|
                                       image_query | table_query | form_query|
                                       link_variable.textualanchor mentions string |
                                       link_variable.url = url_descriptor Depth number

table_related_variable              ←  column_variable[integer] | row_variable[integer] |
                                       column_variable| row_variable| cell_variable

table_variable                      ←  A | B | ...
row_variable                        ←  A | B | ...
column_variable                     ←  A | B | ...
cell_variable                       ←  A | B | ...

nested_form_query                   ←  form_query_description

form_query_description              ←  atomic_form_query_description|
                                       (form_query_description and form_query_description)

atomic_form_query_description       ←  form_variable mentions keyword |
                                       form_variable contains link link_variable|
                                       form_variable contains table table_variable|
                                       form_variable contains form form_variable|
                                       form_variable contains image image_variable|
                                       image_query | table_query | form_query|
                                       link_variable.textualanchor mentions string |
                                       link_variable.url = url_descriptor Depth number

form_variable_description           ←  A | B | ...
```

Appendix B: Sample HTML Document

```
<head>
<title>Bulls Player File: Dennis Rodman </title>
</head>
<body>

<TABLE border=0 width=620>
<tr>
<td valign=top width=90 height=100>
<img src="/Bulls/images/logo_sub.gif">
<a href="http://www.nba.com/Bulls/"> Chicago Bulls </a>
</td>

<td valign=top width=310>
<font face=arial size=2 color="#000000"><b>PLAYER   FILE</B></FONT><BR>
<font face=arial size=6 color="#fcfcfc"><b>Dennis Rodman<b>
</font>
</td>
<td valign=top width=75>
<img src="/playerfile/images/dennis_rodman.jpg" width=65 height=90 border=0 hspace=8 align=left>
 </td><td valign=top  align=left width=135>
<font size=2 face=arial color="#fcfcfc"><b>Position: Forward</b></font><br>
<font size=2 face=arial color="#fcfcfc">Born: 05/13/61</font><br>
<font size=2 face=arial color="#fcfcfc">Height: 6-8</font><br>
<font size=2 face=arial color="#fcfcfc">Weight: 220 lbs. </font><br>
<font size=2 face=arial color="#fcfcfc">College: Southeastern Oklahoma State '86</font><br>
</td>
</TABLE>

<p><br>
<a href="/playerindex.html">NBA Player Directory</a><br>
<a href="/index.html">NBA.com</a></font><br>

</body>
</html>
```

Appendix C: Parsing Result for Sample HTML Document

Attribute Name	Values
Doc.Title:	Bulls Player File: Dennis Rodman
Doc.URL:	http://www.nba.com/playerfile/dennis_rodman.html
Doc.Length:	34443
Doc.Type:	HTML
Doc.Last_modified_date:	866277163 (Sat, 14 Jun 1997 08:32:43 GMT)
Doc.Collected_date:	Thursday, 23-Oct-97 19:09:37 Local time
Doc.Image:	/Bulls/images/logo_sub.gif /playerfile/images/dennis_rodman.jpg
Doc.Form:	NULL
Doc.Table:	<img src="/Bulls/images/logo_sub.gif"> Chicago Bulls PLAYERFILE Dennis Rodman Position: Forward Born: 05/13/61 Height: 6-8 Weight: 220 lbs. College: Southeastern Oklahoma State 86'
Doc.Link (Anchor,URL):	Chicago Bulls, http://www.nba.com/Bulls/ NBA Player Directory, http://www.nba.com/playerindex.html NBA.com, http://www.nba.com/index.html
Image.Metadata:	Metadata in internal format
Image.Caption:	NULL
Image.Format:	gif
Image.Size:	865
Table.Contents:	<img src="/Bulls/images/logo_sub.gif"> Chicago Bulls PLAYERFILE Dennis Rodman Position: Forward Born: 05/13/61 Height: 6-8 Weight: 220 lbs. College: Southeastern Oklahoma State 86'
Table.Caption:	NULL
Table.ContainedImage:	Pointer to the image "*/Bulls/images/logo_sub.gif*"
Table.ContainedLink:	Pointer to the link of *href* = "*http* : *//www.nba.com/Bulls/*"
Table.ContainedForm:	NULL
Form.Contents:	NULL
Form.Caption:	NULL
Form.ContainedImage:	NULL
Form.ContainedForm:	NULL
Form.ContainedTable:	NULL
Link.TexualAnchor:	"Chicago Bulls", "NBA Player Directory", "NBA.com"
Link.ImageAnchor:	NULL
Link.URL:	http://www.nba.com/Bulls/ http://www.nba.com/playerindex.html http://www.nba.com/index.html

Appendix D: Statistic for Parsing HTML Documents

The statistic for our experiments on parsing HTML documents on three Web sites, www.nba.com, www.nfl.com, and www.majorleaguebaseball.com, are

- The number of documents parsed: 2914 in three Web sites.
- Forms are found in 13 documents.
- Tables are found in 2357 documents. Note that tables are frequently used for formatting purposes, rather than to express tabular information. This number includes tables used for both purposes. Using some type of filter to differentiate between "real" and "formatting only" tables is being investigated.
- There are 2324 distinct images found and every document has at least one image.
- There are 37030 links (including intra-document links) among documents.
- Keywords: 8402 distinct keywords are found and 3494 of them can be found in the dictionary.
- Size of the metadata for parsed documents:
  - 10.9M except tables and forms
  - Parsed metadata is 7.5M for tables and 7K for forms.
  - The original raw HTML file size is 23.4M for 2914 documents.

  Note that the metadata here is used to provided full text search.

Author Bibliography

Wen-Syan Li is a Senior Research Staff Member at Computers & Communications Research Laboratories (CCRL), NEC USA Inc. He received his Ph.D. in Computer Science from Northwestern University in December 1995. He also holds an MBA degree. His main research interests include content delivery network, multimedia/hypermedia/document databases, WWW, E-Commerce, and information retrieval. He is leading CachePortal project at NEC USA Venture Development Center and Content Awareness Network project at NEC CCRL in San Jose. Wen-Syan is the recipient of the first NEC USA Achievement Award for his contributions in technology innovation.

Junho Shim, Ph.D. He received his M.S. degree from Seoul National University at Seoul, Korea, in 1994 and the Ph.D. degree in Computer Science from Northwestern University at Evanston, Illinois, USA, in 1998. He is currently a Assistant Professor at Department of Computer Science in Sookmyung Womens University, Seoul, Korea. Previously he worked at Computer Associates International, New York, USA, and held assistant professor position with Drexel University, Pennsylvania, USA. His research interests include database systems, data warehousing, client-server architectures, information infrastructures, and distributed systems

Kasım Selçuk Candan is a tenure track assistant professor at the Department of Computer Science and Engineering at the Arizona State University. He joined the department in August 1997, after receiving his Ph.D. from the Computer Science Department at the University of Maryland at College Park. His dissertation research concentrated on multimedia document authoring, presentation, and

retrieval in distributed collaborative environments. He received the 1997 ACM DC Chapter award of Samuel N. Alexander Fellowship for his Ph.D. work. His research interests include development of formal models, indexing schemes, and retrieval algorithms for multimedia and Web information and development of novel query optimization and processing algorithms. He has published various articles in respected journals and conferences in related areas. He received his B.S. degree, first ranked in the department, in computer science from Bilkent University in Turkey in 1993.