EARLY EXPERIENCE WITH W3QS - A WWW INFORMATION GATHERING SYSTEM

D. Konopnicki
Computer Science Department
Technion
Haifa, Israel 32000

O. Shmueli*
Computer Science Department
Technion
Haifa, Israel 32000

Abstract
The World-Wide Web (WWW) is a fast growing global information resource. It contains an enormous amount of heterogeneous information and provides access to a variety of services. Searching for information and services in the WWW is a widely recognized problem.

We have designed a high level SQL-like language called W3QL, to support effective and flexible WWW query processing. W3QL addresses both the structure and the content of WWW nodes. W3QL allows the use of indices, navigation through forms and view updating specification. We have implemented a system called W3QS which executes W3QL queries.

The system and its query language set a framework for the development of database-like tools for the WWW. The current architecture of W3QS provides a server which enables users to pose queries as well as integrate their own data analysis tools.

1 INTRODUCTION
The World Wide Web (WWW) is a fast growing global information resource. It contains an enormous amount of heterogeneous information and provides access to a variety of services. Since there is no central control and very few standards of information organization and service offering, searching for information and services is a widely recognized problem. To some degree this problem is solved by "search services", such as Lycos [1], WWW [4], Yahoo [5] and others. These sites employ search engines, known as "robots", that scan the network periodically and form text based indices. These indices are then utilized in performing queries by users. Some new services allow querying more than one 'search service' and consolidation of results [3].

These services are limited in certain important areas. First, the structural information, namely the organization of the document into parts pointing to each other, is lost. In addition, one is limited by the kind of textual analysis provided by the 'search service'. Second, many Web sites provide information based on forms filled by users. Robots do not fill forms as the number of possibilities is enormous, hence robots miss interesting avenues that humans might follow. Finally, even the hardest working robots cannot keep information truly up-to-date.

We have designed a high level SQL-like language called W3QL, to support effective and flexible WWW query processing. W3QL addresses the structure and content of WWW nodes and their varied sorts of data. We have implemented a system called W3QS which executes W3QL queries. W3QL allows the use of indices, navigation through forms and view updating specification.

Related Works
Halasz's pioneering paper [9] identified generic properties and limitations of hypertext based query processing. Analysis of files that have a strict inner-structure is treated in [6]. Such analysis is essential for WWW query processing and is performed, to a limited extent, by W3QS. The Graphlog language [8] can be used for the graphical specification of search patterns. In [7], Beeri and Kornatzky define a logic-based language to state structure-specifying queries on a hypertext structure. Their formulae uniformly treat structural aspects, content aspects and boolean operations. Their language is capable of altering the hypertext structure. A query language on a dynamically changing hypertext structure is proposed in [12]. This language requires an understanding of the internal organization of each hypertext node. W3QL uses a pattern definition sub-language as done in [8] and extends the hypertext model to nodes that handle user input, namely forms. In [13], the authors propose to use programs to periodically scan the whole WWW and build searchable indices from the gathered information. The network scanning programs are called "robots" or "knowbots". This approach leads to the various indices available in the WWW. An interface to the most important Web indices can be found in [2]. W3QS takes advantage of the information provided by those indices and essentially uses index services as traditional relational database indices. WebSQL [11] is a recent SQL-like query language for the WWW similar
in spirit to W3QL [10]. WebSQL addresses the issues of content and structure queries but does not allow views and form handling [11].

Paper structure: In section 2, the main characteristics of W3QL and W3QS are presented. Section 3 presents examples of W3QL queries. Section 4 briefly describes the W3QS system. A critical review and an outline of future directions are provided in Section 5.

2 W3QL AND W3QS

We have designed and implemented W3QL, a flexible, declarative, database-like, Web-wide language [10]. Some of its main features are:

- W3QL enables both content queries (queries about the content of the WWW accessible information, e.g. the author of an article) and structural queries (queries about the hypertext organization of WWW accessible data).
- W3QL takes advantage of existing WWW indices and search services.
- W3QL allows easy interfacing to user written programs as well as standard UNIX utilities.
- W3QL provides a view maintenance facility.

W3QS is a prototype system that executes W3QL queries. Some of its main features are:

- W3QS is accessible via any WWW-browser.
- Applications can utilize W3QS using an API defined by a function library.
- W3QS provides utilities for file content analysis and learning online form filling.
- W3QS allows users to integrate their own data analysis tools.
- W3QS simplifies the building of WWW search programs by providing a library of PERL [14] objects that can be used as "building blocks".

3 W3QL QUERY EXAMPLES

W3QS is very useful for extracting specific information from a site. While indexers, e.g. Lycos, usually return links to a page, forcing the user to perform information extraction on his own, W3QS automates this process. Consider the following example.

Writing a query, step-by-step, for finding all the images within the CS department site. We are interested in all the images accessible through pages that are mentioned in the CS department home page. We need:

- a starting point for the search (the home page of the CS department),
- to scan the pages accessible from this starting point that are located in the CS department site, and
- to extract the image files accessible from these pages.

Therefore,

1. We search for a path beginning at the CS department home page and ending at an image by following only hypertext links that stay within the CS department site. Such a path is expressed in W3QL by the expression:
   \[ n_1, n_{11}, (n_2, 12), 13, n_3 \]
2. Here, \( n_1 \) is the first node of the path and is the CS department home page. This is written as the following domain condition:
   \[ n_1 \text{ in } \{\text{http://www.cs.technion.ac.il}\}; (n_2, 12) \text{ an unbounded length path (length } \geq 1) \text{ of pages accessible from } n_1. \]
3. The links between these pages must stay in the CS department site. So, we require that the address of the target of a link contains the string 'technion.ac.il', as follows:
   \[ 12 \text{ in } \{\text{/technion\.ac\.il/}\}. \text{ (The expression in the curly brackets is a PERL regular expression.)} \]
4. The last node, \( n_3 \), must be an image located in the CS site. We use the PERLCOND program, a file analyzer developed within the project, to analyze the content of the node. This is expressed by the following script condition:
   \[ n_3 \text{ PERLCOND 'n_3.format == /image/'}; \]
5. In order not to overload the department's HTTP server, we limit the search length to 1000 HTTP requests and the explored path maximum length to 5. This is done by passing arguments to the algorithm used for this search, namely ISEARCHd.

The query is then:

Select
From n1,11,(n2,12),13,n3
Where
n1 in {http://www.cs.technion.ac.il};
12 in {/technion\.ac\.il/};
13 in {/technion\.ac\.il/};
n3: PERLCOND 'n_3.format == /image/'
Using ISEARCHd -d 5 -l 1000

The results of the query are returned in a table. Column \( n_3 \), in the table returned by W3QS, contains links to all the found images. If the first line of the query is changed to Select \( n_3 \); the effect is that those images are actually copied by W3QS into a subdirectory for this query. The user can obtain this directory, in Unix tar format and compressed. If the directive Evaluated every day is added to the query, the query will be executed every day in order to refresh its results.

Using an Index: To access complex services, search programs must be able to handle HTML forms. W3QS is able to learn how to fill out forms and then fills out forms automatically. The form filling learning mechanism can be used to access information supplied by indexers. Once the Lycos form is “learned” by W3QS, it can later be used to find starting points for searches. Essentially, we can turn existing WWW indexers into traditional database indices. We use the following query to search for information about ‘Pet Therapy’:

1. Select \( n_3 \);
2. From n1,11,(n2,12),13,n3
3. Where n1 in {
   http://eleven.srv.lycos.com/lycos-form.html};
4. Run learnform n1 cs76:0 If n1 Unknown in LycosDof;
5. fill 11 As In LycosDOF with
query = "therapy";

7    11: PERLCOND '11.content "/FORM/i';
8    n3: PERLCOND '(n3.content "/ltherapy/i) ||
9    (n3.content "/cat/i) ||
10    (n3.content "/bird/i)'
11    Using ISEARCHd -l 5000 -d 3

In line 6, the Lycos form is filled with the keyword 'therapy'. The Lycos supplied answer is the first node of the path (n2,12). We search for a node reachable from the Lycos answer page that contains the word 'therapy' or 'cat' or 'bird'. Observe that in line 4 we could have supplied the HTTP addresses of more than one indexers. Then, the query will need to be changed to reflect the keywords used by these other indexers.

4 THE W3QS SYSTEM

4.1 System Overview - User's view

W3QS is accessible through any WWW browser. The opening screen requests a login name and a password. A login name is required because each user has his/her own account which contains his/her private information. The information consists of: the user's queries and their results, the form files (that contain information about the filling of forms that were previously encountered by the user) and information concerning which views (queries) need be periodically updated and at what time intervals. When requested, a directory tree containing the actual files extracted from the WWW, while answering a particular query, is built. This mechanism allows the user to store a local copy of the gathered information.

At any point, the user can choose one of the following interactions:
1. Main: Return to the opening screen.
2. Add: Add a new query to the account.
3. Edit: Either edit an existing query or edit a template query which is a parameterized query with blanks to be filled by the user. A query that is submitted for execution is first parsed. If there are syntax errors, these are reported and execution stops. Otherwise, the system starts executing the query. This execution is done in background mode.
4. Delete: Delete a previously stored query, its results and messages.
5. List: A listing of all currently stored queries in tabular form.
6. Messages: Here, one can choose a query name and view messages that were generated during its execution. This viewing may be concurrent with the actual execution and so it provides a means for tracking executions and incrementally viewing potential results.
7. Results: Here, one can choose a query name and view the query results that were previously generated during its execution.

4.2 System Overview - System Components

We briefly explain the functionality of the major components described in the figure above:
1. Interfaces. One may access W3QS through a WWW browser (e.g. Netscape). A CGI script presents the user interface. Alternatively, a library of functions presents an API for applications that use W3QS services.
2. W3QS Server. The server manages the interactions with the different users. The server notifies the scheduler of changes which may affect executions (query added or deleted). This client-server architecture allows several users to work concurrently.
3. The Scheduler manages the list of active queries currently in the system. The scheduler executes new queries and refreshes views when needed. Query execution is initiated by calling a Remote Search Program (RSP).
4. The RSPs are the WWW search engines. Several RSP types implementing different search algorithms may be used. Writing new RSPs is made easy by using the RSP function library, a set of PERL classes implementing the primitive actions of a search, e.g. loading a page from the WWW and verifying that a page satisfies some conditions. In order to verify conditions, RSP uses standard Unix utilities such as grep or special condition checking programs from the W3QS Condition Programs Library.
5. Condition Programs Library. The main condition checking program is called PERLCOND. It uses the Unix file utility to determine the page format, and then it performs a rough analysis of the page (file) content. PERLCOND can verify conditions on the page content (See Table 1).
Table 1: PERLCOND conditions

<table>
<thead>
<tr>
<th>format recognized</th>
<th>example of condition</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>images</td>
<td>n1.format = /gif/</td>
<td>The page is a gif image</td>
</tr>
<tr>
<td>HTML</td>
<td>n1.title.content eq 'How to cure zebras'</td>
<td>The title of the HTML page is 'How to cure zebras'</td>
</tr>
<tr>
<td></td>
<td>n1.anchor[5].content = /zoo/</td>
<td>The fifth hypertext anchor contains the word 'zoo'</td>
</tr>
<tr>
<td>Latex</td>
<td>n1.section[1].content = /Zebra/</td>
<td>The title of the first section contains the word 'Zebra'</td>
</tr>
<tr>
<td>Postscript</td>
<td>n1.Author.content eq 'H. G. Wells'</td>
<td>The author of the file is 'H. G. Wells'</td>
</tr>
</tbody>
</table>

5 CRITICAL REVIEW AND FUTURE DIRECTIONS

W3QS offers a useful, albeit not yet polished, service. Its concept, that of a general database language, seems to be a promising direction in constructing useful services and functions for naive as well as more sophisticated users. Necessary improvements to W3QL and W3QS which have been identified include:
1. Syntax readability.
2. Language power.
   - A more powerful analysis of file formats and parsing.
   - The ability to project the result table on specific columns.
   - The ability to use negation, e.g. "there is no path from n1 to n2".
   - The ability to use some fragment of aggregation so that one can write queries such as "the lowest cost CD such that ...", ...
   - In some problems that we have tackled a need arose for having more than one graph in the query's From-Clause.
   - The ability to union results of queries (at the query language level).
3. W3QS capabilities.
   - With the emergence of languages such as Java and Python, which enable the execution of remotely supplied code (applets), we need to reconsider the current architectural decision to execute all user queries at the W3QS site.
   - A better presentation of results, employing graphic tools is needed.
   - Currently, the system is capable of handling only one path in its From-Clause graph specification. Clearly, more complex graphs should be specifiable.
   - The form handling ideas need be refined so that many forms can share their keywords. W3QS should also be able to handle non-textual forms.
   - Several interfaces, graded according to user sophistication, should be offered.

References