Missouri University of Science and Technology
Scholars' Mine

Computer Science Faculty Research & Creative Works

Computer Science

01 Jan 2002

Controlling Web Query Execution in a Web Warehouse

Sanjay Kumar Madria
Missouri University of Science and Technology, madrias@mst.edu

Sourav S. Bhowmick

Recommended Citation

S. K. Madria and S. S. Bhowmick, "Controlling Web Query Execution in a Web Warehouse," *Proceedings of the 13th International Workshop on Database and Expert Systems Applications, 2002*, Institute of Electrical and Electronics Engineers (IEEE), Jan 2002.
The definitive version is available at https://doi.org/10.1109/DEXA.2002.1045996

Controlling Web Query Execution in a Web Warehouse

Sourav S Bhowmick
Nanyang Technological University
School of Computer Engineering
Singapore 639798
assourav@ntu.edu.sg

Sanjay Madria
University of Missouri-Rolla
Department of Computer Science
Rolla 65409
madrias@umr.edu

Abstract

Most of the contemporary web query systems have limited capabilities in controlling web query execution. Such query facility is important as it gives us an opportunity to optimize the evaluation of a web query. In this paper, we address this issue in the context of our web warehousing system called WHOWEDA *(*W*are*h*ouse* O*f* We*b* D*at*a*). Specifically, we investigate different types of* constraints *(related to query execution) which may be imposed on a web query such as number of query results, time of execution, restrict the evaluation of a query to specified set of web sites, etc.. An important feature of our approach is that it attempts to address the query evaluation issues which may arise due to the existence of broken links and forms in the Web.*

1 Introduction

The exponential growth of the Web in the last few years had a significant impact on the techniques used for querying Web data. One of the significant issue with respect to web queries is the execution of the query on the Web data. In contrast to the conventional database systems, we believe that it is important to investigate mechanisms to control the execution of a web query. The primary motivation is to address *some* of the limitations of the Web. Because the Web is large and the content and structure of relevant information may not be completely determined ahead of time, a web query can take considerable amount of time. Typically, the execution time of a web query depends on many factors such as number of instances satisfying the web query, *complexity* of the query, network traffic, locality cost [5] etc. The combine effect of these factor can adversely affect the time taken to retrieve the query results from the Web. Secondly, in conventional database, a user may wish to determine the complete results for his needs. For example, a user may wish to find details of all the employees whose salary is more than 50K. However, this computation of complete results may not always be necessary in the context of the Web. A user may not be looking for complete results of a query but a portion of it which is sufficient to satisfy his/her needs. For instance, suppose we wish to retrieve information related to the treatments of AIDS from the Web. There are numerous sites which provides information related to treatments of AIDS. Consequently, information from different sites overlap with one another as there are finite set of treatment procedures for AIDS. Thus, retrieving all documents related to treatments of AIDS may not be meaningful to a user. A user may only download a small portion of these documents that is sufficient for information related to treatment for AIDS. Thus, a user may ask a query with a condition set on the time allowed to answer the query. This condition may also be expressed by a condition on the number of query results returned, number of documents and so on. Observe that if we do not control the query processing, then the query may retrieve large number of results many of which are similar to one another.

Although the importance of investigating mechanism for controlling web query execution is undeniable, most of the web query systems support very limited [3] form of such mechanism, if any. A limited form of control mechanism which involves the variability of the depth of traversal of a query can be expressed by these languages. Only NetQL [4] provides a set of mechanisms to control the complexity of query processing. In NetQL, it is controlled in two levels. First, users are given various choices to control run time. For example, they can specify a more exact path if they have partial knowledge of the structure of the searched site or simply limit the evaluation of queries to local data or a fixed number of returned results. Second, an optimizing technique, which is based on semantic similarity, is developed to guide the search to the most promising direction.

In this paper, we explore various mechanisms for controlling the execution of a web query. Compared to NetQL [4], we focus on developing various choices to control the query processing. We provide much richer variety of choices compared to NetQL. For instance, our con-

Proceedings of the 13th International Workshop on Database and Expert Systems Applications (DEXA'02)
1529-4188/02 $17.00 © 2002 IEEE

trol mechanism attempts to address issues which may arise when a web query fails to traverse due to existence of forms or broken links.

2 Framework

We use as a starting point the WHOWEDA system (*W*are*ho*use *O*f *We*b *Da*ta), a data warehousing system for managing and manipulating relevant data extracted from the Web [1]. In WHOWEDA, a web query is specified using a *coupling query* [2]. A coupling query is used to populate the web warehouse by querying a set of interlinked HTML or XML documents. Coupling query can be visualized as a directed connected acyclic graph. Some of the important features of our query mechanism are ability to query metadata, content, internal and external (hyperlink) structure of Web documents based on *partial* knowledge, ability to express constraints on tag attributes and tagless segment of data, ability to express conjunctive as well as disjunctive query conditions compactly, ability to control execution of a web query and preservation of the topological structure of hyperlinked documents in the query results. Informally, the results of a coupling query is a set of directed connected acyclic graph containing node and link objects. These graphs are called *web tuples*. Intuitively, the set of documents and hyperlinks in a web tuple satisfies the constraints defined in a coupling query. Each web tuple preserves the inter-document structure of the relevant web documents.Note that our purpose is not to discuss about another powerful web query system. Instead, we focus on the problem mentioned above. Coupling query attempts to overcome the problems unsolved by most of the contemporary query languages. Specifically , *coupling query predicates* in a coupling query enables us to control the execution of a query. These components are specified by a user while formulating a coupling query. *Coupling query predicates* can be used to enforce a query to traverse a specified set of Web sites. It can also be used to control the number of query results, time of execution etc.. Note that it does not impose constraint on the content and structure of Web documents and hyperlinks.

3 Query Attributes

We now focus our discussion on the components which enables us to control execution of a coupling query, i.e., coupling query predicates. *Query attributes* in a coupling query predicate are a set of attributes associated with a coupling query over which one may impose additional constraints. We now elaborate on the motivation and syntax of each query attributes.

3.1 Polling Frequency

The attribute `polling frequency` is used to enforce a coupling query to be executed periodically. An ordinary coupling query is evaluated over the current state of the WWW, the results are then stored in a web table. An example of an ordinary web query is "find a list of side-effects and uses of drugs for various diseases". In contrast, a *polling* coupling query is a coupling query that repeatedly scans the Web for new results based on some given criteria. An example of polling web query is "every 15 days, find a list of side-effects and uses of drugs for various diseases". The polling frequency indicates the time at which the Web sources are to be polled.

3.2 Host

Sometimes it is required to restrict the execution of a web query to a particular host, e.g., `rex.nci.nih.gov`. The execution of web query only follow the links inside a web server. We restrict the execution of a web query to a particular host by imposing predicates on the coupling query itself in lieu of specifying predicates on the node type identifiers in the coupling query. The coupling query predicate is imposed on the attribute `host` to achieve this.

3.3 Number of Tuples

Unlike conventional database, one may wish to restrict number of instances returned by a coupling query or the time taken to execute a query for the following reasons:

- The execution time of a coupling query depends on many factors such as number of instances satisfying the coupling query, existence of free node or link type identifiers in the coupling query, *complexity* of the query, network traffic, locality cost [5] etc. The combine effect of these factor can adversely affect the time taken to retrieve the query results from the Web.
- In conventional database, a user may wish to determine the complete results for his needs. For example, a user may wish to find details of all the employees whose salary is more than 50K. However, this computation of complete results may not always be necessary in the context of the Web. A user may not be looking for complete results of a query but a portion of it which is sufficient to satisfy his/her needs. For instance, suppose we wish to determine the treatments of AIDS from the Web. There are numerous sites which provides information related to treatments of AIDS. However, these information from different sites overlap with one another as there are finite set of treatment procedures for AIDS. Thus, retrieving all documents

related to treatments of AIDS may not be meaningful to a user. A user may only download a small portion of these documents that is sufficient for information related to treatment for AIDS.

The query attributes `number_of_tuples`, `total_nodes`, `host_tuples`, `host_nodes` and `time` discussed next are used to provide the flexibility to control the execution time of a query. In this section, we discuss `number_of_tuples`. In the subsequent sections we discuss the remaining attributes.

The attribute `number_of_tuples` restrict the search to a certain number of returned web tuples. If the specified number of tuples is exceeded, the search stops. For instance,

$$q(G) \equiv G.\texttt{number\_of\_tuples LT "}\textit{100}\texttt{"}$$

The above predicate specifies that the coupling query should stop executing as number of web tuples exceed 100.

3.4 Total Nodes

Note that the coupling query attribute `number_of_tuples` indirectly affect the number of instances of node type identifiers. However, the user does not know ahead of time the precise number of instances of each node type identifier . Sometimes it may be required to control these instances of node type identifiers. For example, suppose instances of node type identifier d in a coupling query are pages containing treatment related information for a particular disease, say AIDS. As mentioned earlier there may exist thousands of pages in the Web containing information related to treatments for AIDS that satisfies the predicate defined on d. A user may wish to limit the number of web tuples returned by a query by specifying maximum number of instances of d to be retrieved. If the specified number of instances exceed, the execution of the coupling query stops. The attribute `total_nodes` is used to control the total number of instances of a node type identifier . Let $G = \langle X_n, X_\ell, C, P, Q\rangle$ be a coupling query. Then, the syntax of the coupling query predicate containing `total_nodes` attribute is given below where $x \in X_n$ and:

$$q(G) \equiv G.\texttt{(total\_nodes::}x\texttt{) op "V"}$$

The node type identifier whose total number of instances is to be controlled is specified by the identifier x and is separated from the query attribute `total_nodes` by the symbol `"::"`. An example of such coupling query predicate is given below:

$$q(G) \equiv G.\texttt{(total\_nodes::}d\texttt{) LT "}\textit{100}\texttt{"}$$

The above predicate specifies that the coupling query G stops executing when the number of instances of d exceeds 100.

3.5 Host Nodes & Host Tuples

A coupling query may return relevant set of documents from one or more Web sites. Although the attributes `number_of_tuples` and `total_nodes` may be used to limit the number of results of a query but it has *some* limitations. When a coupling query retrieves information from a single Web site (that is, all nodes in the query result belongs to the same Web site), the attributes `number_of_tuples` and `total_nodes` may provide the user flexibility to control the instances of query result. However, when a coupling query retrieves result from multiple heterogeneous Web sites these query attributes are not always the most efficient way of controlling the number of instances retrieved from *each* Web site. These attributes are useful in controlling the total number of web tuples or node instances but they are not an accurate mechanism for controlling number of web tuples and node instances from each Web site. Note that this problem does not arise when the query results contain information from a single Web site only. We elaborate this with the following examples.

Suppose a coupling query retrieves web tuples from three Web sites, say h_1, h_2 and h_3. Suppose we wish to control number of instances retrieved from each Web site. For instance, we wish to retrieve maximum 50 tuples from each of these three Web sites. That is, the total number of tuples retrieved from h_1, h_2 and h_3 will be less than 150. However, the predicate

$$q(G) \equiv G.\texttt{number\_of\_tuples LT "}\textit{150}\texttt{"}$$

may retrieve 150 tuples from h_1 and no tuples from h_2 and h_3. It may also retrieve 50 tuples from h_1, 100 from h_2 and none from h_3. Similarly, if we wish to control the instances of a node type identifier , say d for each Web site, then the attribute `total_node` may not be the most efficient way of doing it. Suppose from each Web site h_1, h_2 and h_3 we wish to gather at the most 20 instances of d. Thus, the maximum number of instances of d in the query results from these three Web sites must not exceed 60. The coupling query predicate used to impose this constraint is as follows:

$$q(G) \equiv G.\texttt{(total\_nodes::}d\texttt{) LT "}\textit{60}\texttt{"}$$

However, the coupling query may retrieve 60 instances of d from h_1 and none from h_2 and h_3. It may also retrieve 30, 25 and 5 instances of d from h_1, h_2 and h_3 respectively.

In order to resolve the above limitation for query results containing tuples or nodes from multiple Web sites we introduce coupling query predicates containing the attributes `host_tuples` and `host_nodes`. These attributes are used to control the number of web tuples and specified node objects from each Web sites respectively. The syntax of the coupling query predicate involving `host_tuples` is as follows:

$$q(G) \equiv G.\texttt{host\_tuples SATISFIES "V"}$$

where $V = n$ where n is an integer i.e, 100, 200 etc. or it is of the form $V = \{(h_1 \texttt{ op } n_1), (h_2 \texttt{ op } n_2), \ldots, (h_k \texttt{ op } n_k)\}$ where h_i is a string and indicates host name, n_i is an integer and op is relational operators such as $=, \leq, \geq, >, <$. Note that $V = n$ specifies that each Web site should satisfy the constraint imposed by the coupling query predicate. Some examples of coupling query predicates containing host_tuples attribute are:

$$\begin{aligned} q(G) &\equiv G.\texttt{host\_tuples SATISFIES "}\textit{50}\texttt{"} \\ q(G) &\equiv G.\texttt{host\_tuples SATISFIES} \\ &\quad \texttt{"}\textit{(ntu.edu.sg} \leq \textit{30,nus.edu.sg} \leq \textit{20)}\texttt{"} \end{aligned}$$

The first predicate indicates that number of tuples from each Web site must not exceed 50 and the second predicate specifies that number of web tuples from the Web sites ntu.edu.sg and nus.edu.sg must not exceed 30 and 20 respectively.

Similarly, the syntax of coupling query predicate containing the attribute host_nodes is as follows:

$$q(G) \equiv G.\texttt{(host\_nodes::}x\texttt{) SATISFIES "V"}$$

where $x \in X_n$, $p(x) \in P$, $X_n \in G$ and syntax of V is identical to that in predicates containing host_tuples. Some examples of this type of coupling query predicate are:

$$\begin{aligned} q(G) &\equiv G.\texttt{(host\_nodes::}d\texttt{) SATISFIES "}\textit{20}\texttt{"} \\ q(G) &\equiv G.\texttt{(host\_nodes::}d\texttt{) SATISFIES} \\ &\quad \texttt{"}\textit{(ntu.edu.sg} \leq \textit{20,nus.edu.sg} \leq \textit{10)}\texttt{"} \end{aligned}$$

The first predicate indicates that number of instances of d from each Web site must not exceed 20 and the second predicate specifies that number of instances of d from the Web sites ntu.edu.sg and nus.edu.sg must not exceed 20 and 10 respectively.

3.6 Time

The attribute time enables us to restrict the processing of web query to a specified amount of time. When the time reaches its limit, the execution stops and returns the web tuples found so far. For instance, consider Example 1. The predicate q_1 specifies that the coupling query G_1 will be executed for 20 minutes only.

3.7 Broken Link Mode

While traversing the Web, the execution of a coupling query may often encounter "Document Not Found" error (Error 404) (also called as *broken links*). A broken link prevents a query to traverse further. Hence, it may seem that whenever a coupling query encounters a broken link it should ignore the corresponding nodes and links as it does not satisfy the coupling query. However, including broken links in the web tuples has the following advantages:

Existence of relevant Web sites: If the results of a query does not contain tuples containing broken links, the user will not be aware of the existence of the corresponding Web site or Web pages which may provide useful information. Note that the broken link does not necessarily indicate that the Web site does not exist anymore. It may indicate that the address of the Web site has changed or it is temporarily removed from the Web or it may also suggest that the value of the HREF attribute of the link may contain typographical errors. Thus, awareness of the user about the existence of the Web site may result in inspection of neighbouring documents or modification of the coupling query by the user to locate the valid address of this Web site. However, if the query result ignores web tuples containing broken links then it may not be possible for a user to modify his/her query to locate the valid addresses of the Web sites or Web documents simply because he is not aware of it.

Appropriateness of Web sites for relevant information: Identifying number of broken links in a query result also helps us to determine the currency of relevant Web site(s) with respect to the information the user is looking for. For instance, suppose we wish to retrieve information I from a set of Web sites. If the query results contains large number of web tuples that has encountered broken links then it may suggest that the Web sites considered for retrieving I may not be the "best" collection of sources for information related to I. Note that a user may only be able to determine these issues if he/she is aware of the existence of web tuples containing broken links in the query result.

Determining currently unavailable information using polling coupling query: Observe that a broken link may occur if a Web document or Web site is temporarily removed from the Web. That is, it may not be available at time t_1, but may be globally accessible at time t_2 where $t_2 > t_1$. If we allow web tuples containing broken links in the query results then a user may be able to modify the query and execute it periodically to retrieve the set of relevant documents which were not previously available. However, a user may not be aware of these information if web tuples containing broken links are disregarded in query results. Thus, he/she may not find the need to re-execute the query periodically to retrieve these relevant web tuples which previously contained broken links.

The attribute broken_link_mode is used in the coupling query predicate to provide the flexibility to materialize web tuples containing broken links. The value of such attribute is set to "on", if we wish to retrieve web tuples that encountered broken links. On the other hand, the "off" mode will not retrieve and materialize in the web table those web tuples that encountered 404 error. By default, the broken_link_mode is set to "off".

3.8 Form Mode

While traversing the Web, the execution of a coupling query may often encounter Web pages containing forms. A query may encounter the following two types of Web pages containing forms: (1) Hyperlink-free form: A Web page containing form with no local or global hyperlinks to other Web pages is called a *hyperlink-free form*. (2) Non-hyperlink-free form: A Web page containing form as well as hyperlinks to other Web documents is called a *non-hyperlink-free form*. A non-hyperlink-free form does not hinder a query to navigate further from the hyperlinks contained in the page to determine the satisfiability of the coupling query. However, this is not true for hyperlink-free forms. To elaborate further, consider a Web page containing hyperlink-free forms. A query fails to navigate further from this Web page. Consequently, a web tuple containing hyperlink-free form may not satisfy a coupling query due to navigational failure. Thus, it may seem that whenever a coupling query encounters a hyperlink-free form it should ignore the corresponding web tuple from the query results. However, not including web tuples that encountered hyperlink-free form in the query result has *some* disadvantages. If the query results ignore web tuples containing hyperlink-free forms then the user will not be aware of the existence corresponding nodes which may provide relevant information. These information are in Web pages that are generated by programs given user inputs (in the forms) and are therefore not accessible to crawling. In order to retrieve data from such documents one has to be aware of the existence of these hyperlink-free forms. Thus, query results that include web tuples containing hyperlink-free forms enables us to further manipulate these forms to harness relevant information.

We impose a predicate on the attribute form_mode of the coupling query for retrieving web tuples containing hyperlink-free forms. If the form_mode is "on" then the coupling query will retrieve web tuples containing hyperlink-free forms. Otherwise, the "off" mode will ignore web tuples that encounter these forms. By default, we set the value to "off".

4 Conclusions & Future Work

In this paper, rather than developing another powerful web query system, we focus on the problems ignored by most of the other query languages. We discussed a query mechanism called coupling query which supports various mechanisms to control the execution of a web query. Currently, we have implemented a preliminary version of the coupling query. As part of our future work, we intend to do the following: (1) We have seen that the query attributes like number_of_tuples, total_nodes enables us to harness a small portion of query results when information from different sites overlap with one another. However, providing such constraints on the coupling query is not sufficient to address this issue as it does not guarantee the retrieval of restricted set of results against information loss. We are exploring ways to minimize information loss while accessing only restricted set of results. (2) Investigate the performance of coupling queries containing non-empty set of coupling query predicates. (3) Perform a cost-benefit analysis of query processing with respect to the usage and non-usage of coupling query predicates. (4) Finally, we intend to optimize coupling query processing and adapt it in such a way that it can emulate browsing behaviour. One of the issues we wish to investigate is how to guide the search to the most promising direction so that the expected web tuples can be obtained as soon as possible. The queries restricted by time and the number of results will be benefited directly. We wish to use semantic information on hyperlink labels for heuristic search.

References

[1] S. S. BHOWMICK. WHOM: A Data Model and Algebra for a Web Warehouse. *PhD Dissertation*, Available from www.ntu.edu.sg/home/assourav/pub.htm, School of Computer Engineering, Nanyang Technological University, Singapore, 2001.

[2] S. S. BHOWMICK, W.-K. NG, S. MADRIA. Anatomy of a Coupling Query in a Web Warehouse. To appear in *International Journal of Software and Information Technology*, Elsevier Science, 2002.

[3] D. FLORESCU, A. LEVY, A. MENDELZON. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Records*, 27(3):59-74, 1998.

[4] M. LIU, T. GUAN, L. V. SAXTON. Structured-Based Queries Over the World Wide Web. *Proceedings of the 17th International Conference on Conceptual Modeling (ER'98)*, pp. 107-120, Singapore, 1998.

[5] A. O. MENDELZON, G. A. MIHAILA, T. MILO. Querying the World Wide Web. *Proceedings of the International Conference on Parallel and Distributed Information Systems (PDIS'96)*, Miami, Florida,