

Distributed (Indexed) Searching: Evolution to XML

In the beginning, there was unindexed searching. Randomly opening documents to find the correct one is a form of unindexed searching. Advanced unindexed searching might iterate over files looking for specific key words. The main drawback of this searching technique is slow speed; it is particularly inefficient for successive searching.

The alternative to unindexed searching is indexed searching. As document collections grow from megabytes to gigabytes to terabytes, searching these large document sets usually mandates some form of indexing.

Just as it is often faster to locate

This article uses dtSearch-based examples for illustrating its main concept — distributed searching using XML as an interchange medium. This is a different use of XML than as a database storage format for holding Web-based and other fielded data.

a particular topic using a book's index rather than thumbing through each page, it is faster to search for information using indexed computer files. Moreover, indexing with a modern full-text search program is easy — simply

click on directories or drives and the search program does the work.

For complex document formats such as Word, WordPerfect, Access, Excel, PowerPoint, PDF, ZIP, HTML, XML, etc., a code at the beginning of each document informs the program what formatting to expect as it parses the document. Good indexing programs can even fully index the occasional corrupt document (the one that a word processor suddenly refuses to open).

Indexed search engines can also operate over a network. If EnterpriseX has a 5,000-page policy manual, each employee could separately index it, saving and



A dog digging in its own backyard to find bones that the dog remembers burying is performing an indexed search.



A dog digging in a neighboring backyard in the hopes of finding some of the neighboring dog's buried bones is performing an unindexed search.

searching the resulting index. Or, for improved efficiency, the network administrator could build one policy manual index for shared access. Hundreds of people can simultaneously search the index, with the network software simultaneously updating the index.

While other search engines are available, this article uses dtSearch-based examples for illustrating its main concept — distributed searching using XML as an interchange medium. This is a different use of XML than as a database storage format for holding Web-based and other fielded data as described in a Web-based data management article in the November / December 2000 issue of *PC AI*.

Whereas that article describes intelligent hierarchical nested-field and full-text searching of XML databases (and other Web-based formats, such as HTML and PDF), this article explores the intelligent XML-based exchange of information. While the data exchange methods may entail XML as well as HTTP / HTML, the target databases can be in any supported format: XML, PDF, HTML, SQL, ZIP, Office, etc.

The distributed search returns to the dtSearch client multiple streams of XML data, which it then combines and presents to the user in a single, unified view... Files appear with highlighted hits, as well as (for HTML and PDF) images and links intact.

Distributed Searching Using HTTP / HTML

Indexed searching is an easy solution for documents residing on a PC or Local Area Network (LAN). However, an EnterpriseX user in New Jersey may need to search files on an EnterpriseX Web server in California.

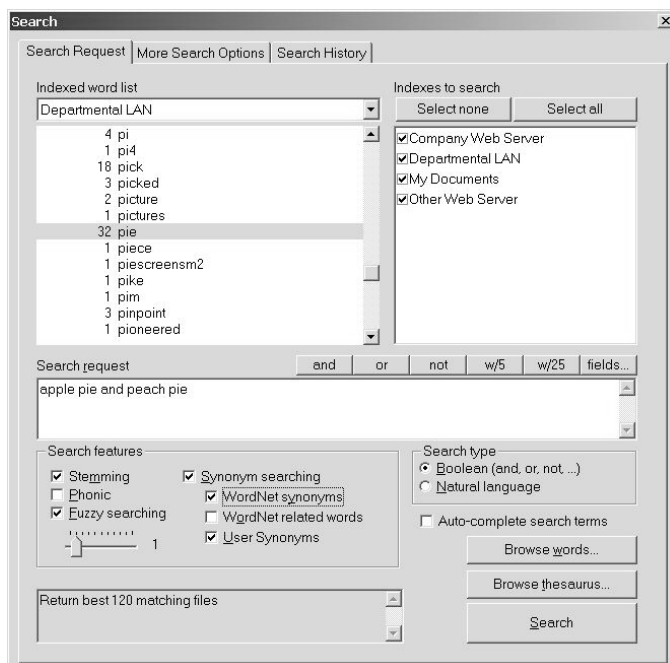
Assuming the site had a search engine, the user could access the EnterpriseX server directly through the user's browser — entering a query through the site's user interface just like any Internet user accessing a search-enabled site. Or the user could query the search engine directly, entering an HTTP request directly in the browser's address bar. A sample search request might be:
<http://www.enterprise-x-california.com/scripts/dtisapi6.dll?cmd=search&request=apple%20pie%20and%20peach%20pie&index=AllDocuments&fuzziness=2&sortby=hits>

This article assumes that the EnterpriseX server in California is running dtSearch Web. While other search programs accept different syntaxes, the overall query concept is generally the same. The first part of the HTTP request is the applicable Uniform Resource Locator (URL). The text query, *apple pie and peach pie*, follows.

The name of the index to search, "AllDocuments," is next. While this search only selects one index, it could also select multiple indexes. Next is a value specifying a fuzziness of 2 to pick up not only exact search terms, but also misspellings. The final value requests that the server sort search results, or all retrieved files matching the search request, according to the number of hits (or matches) in each retrieved file.

The above single-string URL is a GET request. A POST request is similar but allows for transmission of longer streams of data. In response to the GET request, the EnterpriseX Web server would return an HTML stream directly to the user's browser.

If the user were unfamiliar with HTTP syntax, a client application could automatically send the HTTP request through the user's Web browser. The server would return



With distributed searching, a single search request can span multiple local and remote locations. Here, the user selects indexes to search in the client. An HTTP request automatically goes out to remote servers, and a unified, manipulable set of XML search results comes back.

XML Resources

Some XML books to check out:

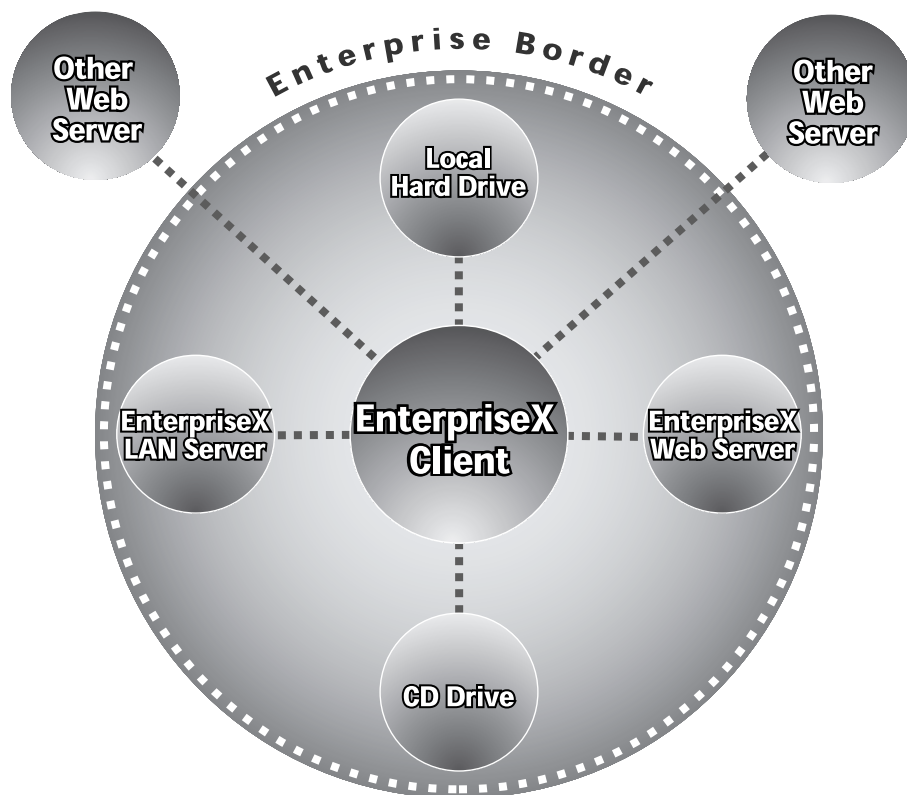
- XML Pocket Reference by Robert Eckstein (published by O'Reilly);
- XML Complete by Steve Holzner (published by McGraw-Hill);
- XML Bible by Elliotte Rusty Harold (published by IDG Books);
- Beginning XML by David Hunter (published by Wrox Press Ltd.).

Some Web-based XML resources:

- www.w3.org/XML (information on XML and XML-related standards).
- www.xml.com (directory of XML-related resources maintained by O'Reilly, a leading publisher of software-related books).
- www.xml.org/ (general XML-related information and news).

Finally, for generally searching XML Web-based programming resources, see www.codehound.com/xml/

Distributed Searching



The distributed search returns to the *dtSearch* client multiple streams of XML data, which it then combines and presents to the user in a single, unified view. Retrieved HTML, PDF, XML, ZIP, Office, etc. files appear with highlighted hits, as well as (for HTML and PDF) images and links intact.

these search results as an HTML stream, this time popping up inside the client application.

Independent of where the user enters the search request, it is the server, not the client, handling most of the workload. If, for example, the user wants the search results returned according to descending number of search terms, or hits, then it is the server, not the client, that does this. To resort search results by file date instead of number of hits requires a separate request to the server.

Since the search results are in HTML, the client software has little ability to understand or manipulate search returns. When the client receives the search results, the browser simply paints the HTML results for the user. The same is true if the user went to a public Web server and entered a similar query.

Distributed Searching Using XML

To obtain smarter search results, a variable residing within the HTTP request indicates whether the search results should be in XML instead of HTML. The use of XML in essence transfers responsibility from the server to the client.

HTML tells the client how to display the data, not what the data is. In contrast, XML tells the client what the data is, not how to display the data. While HTML simply paints a picture of what search results should look like, XML can include numeric values, such as number of hits and word offsets of hits, indicating where hits are in each document.

To illustrate, assume a user searches six different indexes on six

different servers — one on a local hard drive, two on separate network servers, and three on different Web servers. One approach to collectively organizing these search results is listing all retrieved files by hits. The file containing the most hits is first, the file with the next largest number of hits would be second, etc., independent of source.

With HTML, the client receives six different search results back, with no method for combining them. With XML, the client receives six streams of XML data. The client can then combine the information in the XML data, presenting a single, unified and flexible view to the user.

The user could begin viewing retrieved files by number of hits, then instantly resort by file name or file date, with hits still highlighted. Retrieved HTML files appear just as they would in a Web browser, i.e. including embedded links and images, with the addition of highlighted hits.

A retrieved PDF file would look similar to a retrieved HTML file display, including all existing embedded images with highlighted hits. In the case of the PDF file, however, the EnterpriseX server's transmission includes an XML component along with the underlying file.

The server first sends links to the original PDF file followed by links to an XML file describing where the hits are as page and character offsets. Adobe Reader, operating through the browser on the client machine, downloads the XML file with the hit offsets to highlight hits.

Distributed Searching Beyond the Enterprise Borders

The above discussion glosses over a key point. The original URL request included information about a particular Web server, such as exactly where the search engine was running. How does the search client obtain this

Anatomy of an Index

Once a search program has built an index, a full-text search encompassing multiple gigabytes or more usually takes less than a second. A book index typically provides only a list of pages where a term appears. A search program can use the extensive document information that it collects, combined with computer processing, to enable a wide range of sophisticated search techniques.

Sometimes, a single word takes you exactly to the right place in the right document. However, as the number of files increases to the tens of thousands, and more, sophisticated search options to expand the search request to look for multiple words or concepts, to intelligently rank retrieved files, or sift through misspellings, become

vastly more important.

A search index generates a list of every unique word in a document collection along with where in each document the word resides. The word list plus location information enables a variety of phrase, boolean (and/or/not), and proximity searching as well as various hierarchical search-sorting techniques.

For example, relevancy-ranked natural language searching uses the indexed information to rank files according to the density and rarity of search terms in the document collection. The rarer the search term, and the more frequently it appears in a file, the higher that file's overall score. Advanced sorting can also use index information to process user-defined variable term weights, whether positive or

negative.

In addition to exact word searching, the same search index can find related words. An online thesaurus, in combination with an optional user-defined thesaurus, retrieves concepts or synonyms of indexed search terms. The search index also supports word derivatives, such as words that sound alike (phonic searching), different endings for the same route (stemming), and wildcard placeholders.

Indexed fuzzy searching enables the adjustment of word matching at the moment of a search to instantly pick up spelling and typographical deviations. A low-level fuzzy search for *alphabet* also picks up *alphaqet*. Selecting a slightly higher fuzziness, the same search picks up both *alphaqet* and *alp2aquet*.

information, enabling the user to simply select the index name off a search-request dialog box to execute the search?

XML is again the answer. The EnterpriseX California server, for example, could allow a user to select a link that automatically downloads an XML file to the user's desktop. When the user opens the file, the operating system could automatically inform the client, inserting the index name in the search dialog box.

If a Web site does not include this particular option and the site is publicly accessible, then the user could still spider, index, search and display the Web site's contents. In the case of the client accessing the EnterpriseX California server, the index continues to reside on the server.

In contrast (at least with the dtSearch spider), this index would reside on the client machine. Adding in the new XML stream from the spider, the client continues to display unified search results. An XML-enabled single search request can

thus reach within as well as beyond the enterprise borders.

Policing the Enterprise Borders

So how does an enterprise protect its sensitive information in the age of indexed power searching? The answer, ironically, is the simple unindexed search. An unindexed search over a particular data set is much slower than an indexed search. However, a single unindexed search on a data set is faster than building an index and doing an indexed search.

A number of enterprises, for example, are using unindexed searching to develop in-house email filtering applications. A typical application performs a single unindexed search on individual emails and attachments for certain key words before releasing the correspondence across the enterprise's firewall. Ideally, the application informs employees that their correspondence may contain confidential material, as well as

possibly providing them with override capability.

While unindexed searching may be slower than indexed searching, it exhibits many of the same intelligent search features. For example, unindexed searching can allow such features as boolean, phrase, wildcard, proximity, concept/synonym, stemming, phonic, and even fuzzy searching.

Even so, filtering will always remain an uphill battle. And one that takes continued intelligence on the part of in-house administrators and programmers to input ever changing parameters into the filtering program to stop sensitive-topic emails, without disrupting too much the flow of legitimate communication.

XML-enabled indexed distributed searching effectively draws information into the enterprise. Its counterpart, unindexed searching, polices the borders to make sure that sensitive information does not escape.

For more information about dtSearch, please call (800) IT-FINDS or visit www.dtsearch.com