

Web Based Data

Management:

HTML vs. PDF vs. XML

Suppose you have a collection of several thousand text files containing information on various states and localities. In addition to the full text of each document, you want to store meta information on each document, such as which state it relates to, and which locality. The traditional method of storing such information is in a database. Documents relating to New York might be referenced in one database row, while documents relating to New Jersey might be referenced in another database row. The New York documents might be further subdivided into Rural, Urban, and Suburban New York.

It is possible to locate every document in the database related to Rural New Jersey once the names or pointers are retrieved. The database, however, doesn't contain the actual document text; it only contains pointer references to the documents. To retrieve the full text of the documents requires entering the key or pointer reference in a separate program, such as a word processing program. This separation between documents and database results in

additional steps in referencing information.

The software world has tried to address the additional steps resulting from separation of the key or pointer

The dtSearch Text Retrieval Engine, for example, can perform a search for *Rural New Jersey* restricted to meta data, or a full text search for *red schoolhouse*, or a combined meta data, full text boolean or proximity search such as: *(metafield contains Rural New Jersey) and (red schoolhouse w/12 school lunch)*.

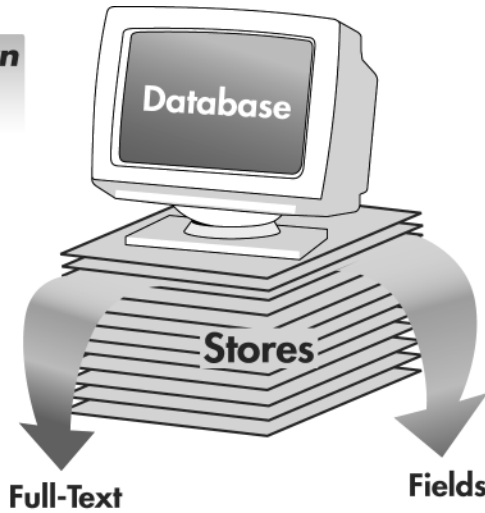
data and the documents. One approach is to enable the database to actually contain text documents along with fields. This article will refer to this solution as the "database down" approach. Databases such as SQL and Oracle store field or meta data for each document, along with

the full text of the document. In this fashion, they support searching for documents that relate to Rural New Jersey, and immediately retrieving them without having to submit pointers to a separate program.

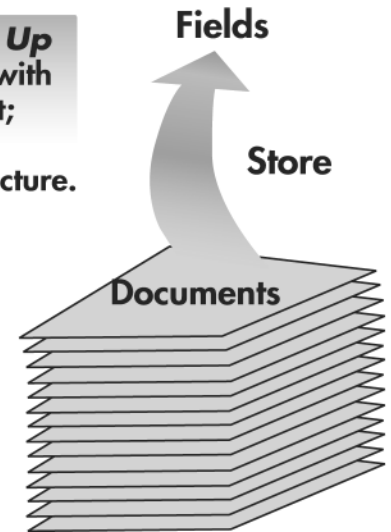
However, the database down approach is no magic bullet. While it does combine both fields and documents under one structure, it does not provide for full portability of documents, keeping fielded data intact, outside of a database structure. While it is possible to remove a document from the database and reuse it in another application, the removal separates the document from its database fields. Copying the meta information along with the document effectively requires transferring the meta information to a similar database type structure. There is thus no truly convenient way to transfer the meta data along with the document.

Another solution for bridging the separation between documents and database is a method that this article will term the "documents up" approach. Whereas the database down approach expands the database

Database Down
Text and field
information
stored separately
inside database.



Documents Up
Fields stored with
document text;
no separate
database structure.



to hold the documents, the documents up approach expands the documents to store the database information. The documents up approach eliminates the need for a separate database structure to hold the meta information — relying instead on the documents themselves to store the meta information.

This documents up approach has many advantages. Since the documents themselves contain the meta information, there is no additional database overhead and they become portable without any need to separately copy field information to preserve meta data. And since each individual document contains its own text as well as its own fields, each individual document becomes a fully self-explanatory and self-contained unit.

HTML – The Universal Format

Today, most text-based documents, including most word processing files, can hold field information, such as title, author, etc. The focus of this article is on Web-based formats that support meta data. The most common Web file type that holds meta as well as full-body text information is HTML.

HTML has the advantages that any Web browser can view it, and a wide range of tools are available for HTML creation and modification. HTML files can also contain embedded images as well as hypertext links to other HTML files for easy navigation.

Not only can a database consisting of HTML documents hold both meta and full text data, but it is also easy to search that data using a wide variety of search tools. The dtSearch Text Retrieval Engine, for example, can perform a search for *Rural New Jersey* restricted to meta data, or a full text search for *red schoolhouse*, or a combined meta data, full text boolean or proximity search such as: (*metafield contains Rural New Jersey*) and (*red schoolhouse w/12 school lunch*).

This search would look for documents whose meta information contains *Rural New Jersey*. Within that resulting document base, it would do a full text search for *red schoolhouse*, and retrieve that phrase anywhere it appeared within 12 words of *school lunch*. With stemming, the search can expand to *red schoolhouses* or

even *red schoolhousing*. With fuzzy searching, the search would find *red schoolhouse* even if it were misspelled as *red skoolhouse*. With variable-term ranked natural language searching instead of boolean searching ... well, you get the idea.

After a search, the user would have customizable, browser-based document sorting, document hit, and document navigation options. The retrieved documents would appear in the browser with the hits graphically marked, as well as all HTML links operational, and all embedded images intact.

PDF – The Other Text and Image Web Format

Another file format that both operates on the Web and supports mixed meta data, full text content, and images is PDF. Whereas HTML is a non-proprietary format, PDF is a proprietary Adobe format.

So why would anyone use PDF? For one thing, it provides tighter integration between text and images than HTML. This makes it a great format for OCR (Optical Character Recognition). The PDF format can fully display the image of a scanned-

Web-based file format	Proprietary file type	Requires browser add-on for viewing	Supports fields along with text	Supports nested-fields	Supports images	Full control over image & text display in browser
HTML	No	No	Yes	No	Yes	No
PDF	Yes Adobe	Yes	Yes	No	Yes	Yes
XML	No	No	Yes	Yes	No	No

in document, while simultaneously containing in hidden text format the OCR'ed version.

And, unlike HTML, which may not look the same in different browsers, the Adobe Reader always gives PDF the same precise look and feel. The level of control over the image that PDF provides makes it, in the right hands, an amazing tool for digital publishing. For example, *www.bookvirtual.com* took the text of this author's story, "Final Orbit," and e-published it in the image of a physical book. (Feel free to check it out.) And PDF is a lockable format, making it more versatile when file security issues are at stake.

So you realize that PDF is convenient for OCR and a great tool for digital publishing. But what about using it as a format for a Web-based text and field searchable repository? While PDF, unlike HTML, contains its own built-in text search functionality, the built-in search functionality does not operate over the Web. A number of third-party search tools do provide

Web-based PDF text searching comparable to HTML searching, including such features as full display over the Web of retrieved PDF files with highlighted hits, and all images intact.

But combining full text search with the ability to support multiple searchable meta fields is a different matter. Adobe includes four fields with PDF files: title, author, subject, and keywords. But for many advanced Web data warehousing needs, four fields are not enough. For example, an organization might want to add a department field, a project ID field, a date field, or even a Rural New Jersey field. DocuLex's OCR Product, *www.doculex.com*, for example, uses an overlay to the PDF document format to provide for extra user-defined fields. Using a proprietary overlay to the dtSearch Engine, it searches documents based on the characteristics of those additional fields, in combination with full text searching.

In connection with dtSearch, products like DocuLex can make full

use of the PDF file format by, for example, providing full support for both hidden and non-hidden document summaries. In recognition of the fact that many PDF documents are very long, such products can supply instant navigation to the location of a search hit. In this way, the user would not have to download 197 pages of a 200-page document before getting to the hit, but could instead immediately jump to page 197. Such products can also provide dynamically re-sortable search results and other bells and whistles, as well as support for HTML and XML.

XML, The Ultimate Text-Based Database

PDF and HTML both have advantages as formats for holding combined images with searchable meta and full text data. For text-only data, XML is an excellent alternative. Not only can it store both full text and field information, but it also supplies a uniquely versatile hierarchical data structure.

XML supports not only single-

Search Results
Request: /play //persona contains king henry
9 document(s) retrieved)

Hits	Document
6	rich_iii_01.xml Location:/docs/ShakespeareXML Date:4/1/00 Size:14104 Title: <play> <title> The Tragedy of Richard the Third <fm> <p> ASCII text plac View file (with hits highlighted) View file (with no highlighting)
4	hen_iv_2_01.xml Location:/docs/ShakespeareXML Date:4/1/00 Size:20002 Title: <play> <title> The Second Part of Henry the Fourth <fm> <p> ASCII text p View file (with hits highlighted) View file (with no highlighting)
4	hen_vi_1_01.xml Location:/docs/ShakespeareXML Date:4/1/00 Size:15153 Title: <play> <title> The First Part of Henry the Sixth <fm> <p> ASCII text pla View file (with hits highlighted) View file (with no highlighting)
4	hen_vi_2_01.xml

```

<persona> TRESSEL
<persona> BERKELEY
<grpdescr> Gentlemen attending on the Lady Anne.
<persona> Lord Mayor of London.
<persona> Sheriff of Wiltshire.
<persona> ELIZABETH, Queen to King Edward IV.
<persona> MARGARET, Widow of King Henry VI.
<persona> DUCHESS of YORK, Mother to King Edward IV.
<persona> LADY ANNE, Widow of Edward Prince of Wales, son to King Henry VI; afterwards
married to Richard.
<persona> A young Daughter of Clarence [MARGARET PLANTAGENET]
<persona> Ghosts of those murdered by Richard III., Lords and other Attendants; a Pursuivant
Scrivener, Citizens, Murderers, Messengers Soldiers, &c.
<scndescr> SCENE England.
<playsub> KING RICHARD III
<act>
<title> ACT I
<scene>
<title> SCENE I. London. A street.
<stagedir> Enter GLOUCESTER, solus
<speech>
<speaker> GLOUCESTER
<line> Now is the winter of our discontent
<line> Made glorious summer by this sun of York;
<line> And all the clouds that lour'd upon our house
<line> In the deep bosom of the ocean buried.
<line> Now are our brows bound with victorious wreaths;
<line> Our bruised arms hung up for monuments;
<line> Our stern alarums changed to merry meetings,
<line> Our dreadful marches to delightful measures.
<line> Grim-visaged war hath smooth'd his wrinkled front;
<line> And now, instead of mounting barded steeds
<line> To fright the souls of fearful adversaries,
<line> He capers nimbly in a lady's chamber
<line> To the lascivious pleasing of a lute.
<line> But I, that am not shaped for sportive tricks,

```

Sample Shakespeare XML data, as displayed by dtSearch Web

level fields such as PDF and HTML, but also nested fields. For example, an XML document might have a field for New Jersey, with a nested field for Rural New Jersey, with another nested field for Very Rural New Jersey, combined with full text passages relating to *red schoolhouses*.

A good example of XML's ability to combine text passages with nested fields is the Shakespeare XML document collection. This collection contains the works of William Shakespeare, converted to XML by Jon Bosak from the public domain sources created by the Moby Lexicon project. See above for a small sample, or visit www.dtsearch2.com for the full text.

Equally important to storing data in a hierarchical structure is searching that data, making full use of its hierarchical structure. This is done through nested field searching. Sample nested field

searches over the Shakespeare XML database might be:

- *persona contains Henry*
- *scenelstagedir contains exeunt citizens*
- *scenelspeech/line contains publius*
- */play/title contains Henry the Fifth*
- *scenellline contains publius*

The first example looks for any field entitled *persona* that contains *Henry*. The second search, containing the / as a field separator, looks for a field called *stagedir* containing *exeunt citizens*, with the *stagedir* field directly nested in a field called *scene*. The third example looks for a triple-nested hierarchical *scenelspeech/line* field sequence containing *publius*. The fourth example, starting with the /, looks for the *play* field at the top of the hierarchy, with a *title* field just beneath it containing *Henry the Fifth*.

The last example, with the //, looks for a field called *line*

containing *publius*. In contrast to the other examples, which specify precise hierarchical sequences, in this last example, the *line* field could be anywhere from directly beneath the *scene* field, to nested at multiple levels of depth. Finally, it is also possible to combine full-text searching with nested field searching such as (*henry the fifth*) and (*scenelspeech/line contains publius*).

Conclusion

HTML, PDF and XML all provide convenient documents up database support, and all have their advantages. For a text-only database, XML supplies maximum flexibility. HTML provides for combined text, fielded data and images in a browser-universal format. PDF provides for maximum control over text in relation to images.

For more information about dtSearch, please call 1-800-IT-FINDS or visit www.dtsearch.com.