Streamline Remote Work With a Full-Text Search Engine

Article contributed by dtSearch®

As companies consider offering "work from anywhere" even after the effects of COVID-19 die down, accessing data in a simple and secure way becomes more important than ever before… dtSearch, shares why full-text search engines are instrumental in streamlining data access today.

When the COVID-19 pandemic hit businesses across industries, the shift to remote work changed from a suggestion to a necessity overnight. Even as states begin to open up, remote work will continue to be more prevalent than ever before. As most of the world continues to work from home and adjust to the "new normal" of remote working, companies are tapping into new technologies to streamline the process of accessing and analyzing data. One of the greatest assets for remote workers is the ability to search for data.

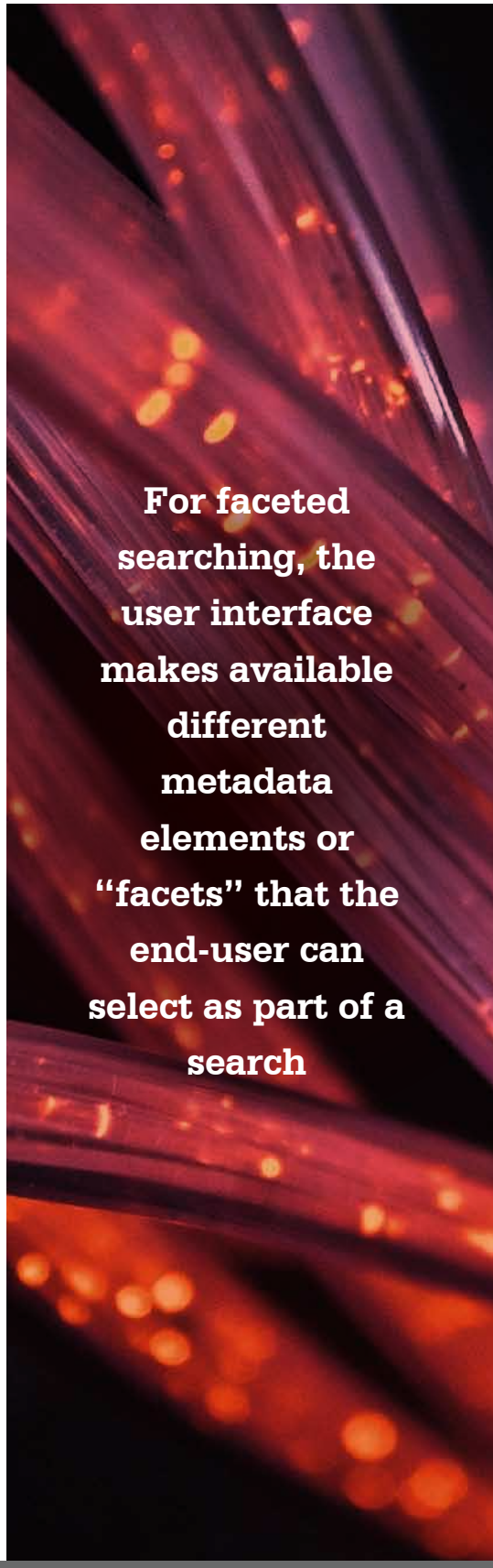
What Is a Full-Text Search Engine?

A full-text search engine runs on-premise or in a cloud-based environment like Azure or AWS. It allows remote workers to instantly search terabytes of "Office" files, PDFs, emails, along with nested attachments, databases, and Internet or Intranet data. Full-text search can operate in a completely stateless manner, making it very easy to scale to any number of concurrent search threads coming from remote workers.

However, even with over 25 different full-text search options, full-text search is only part of efficient remote access. The other part of the equation requires adding a metadata search element. For example, suppose a recruiter is remotely searching millions of resumes or CVs. A full-text search for Nebraska will find Nebraska in the work history field, the education field, etc. But what if the recruiter wants to narrow the search results to only the candidates willing to currently accept a position in Nebraska?

For faceted searching, the user interface makes available different metadata elements or "facets" that the end-user can select as part of a search. For example, the recruiter could check the U.S. as a target job location, then refine it to Midwest, then further refine it to Nebraska.

Simultaneously, the recruiter could drill down through separate metadata, representing job categories, like a programmer, database programmer, SQL, etc. After selecting these and maybe other facets, the recruiter can immediately hone in on the relevant candidates.



Reprinted with permission of Toolbox, toolbox.com | Original article link (9/9/2020)

Where To Store the Metadata?

The key architectural question for maintaining combined full-text and metadata searching is where to store the metadata. There are several different architectural options, each of which has its own pros and cons.

1. The documents can directly hold the metadata.

For example, structured resumes or CVs could contain specific metadata fields that the faceted search can leverage. The benefit of this approach is that it is simple. The data repository can consist of only the documents themselves, with no other overhead.

2. The search engine can add the metadata.

Alternatively, when the search engine indexes the underlying documents, here the resumes or CVs, the search engine can also add metadata or facets "on the fly" as part of the index. That way, if the metadata elements are not already in the underlying documents, they can still be a part of the search equation without requiring the editing of the underlying documents. And once again, the metadata can fall into multiple categories to correspond to a multilayer faceted search.

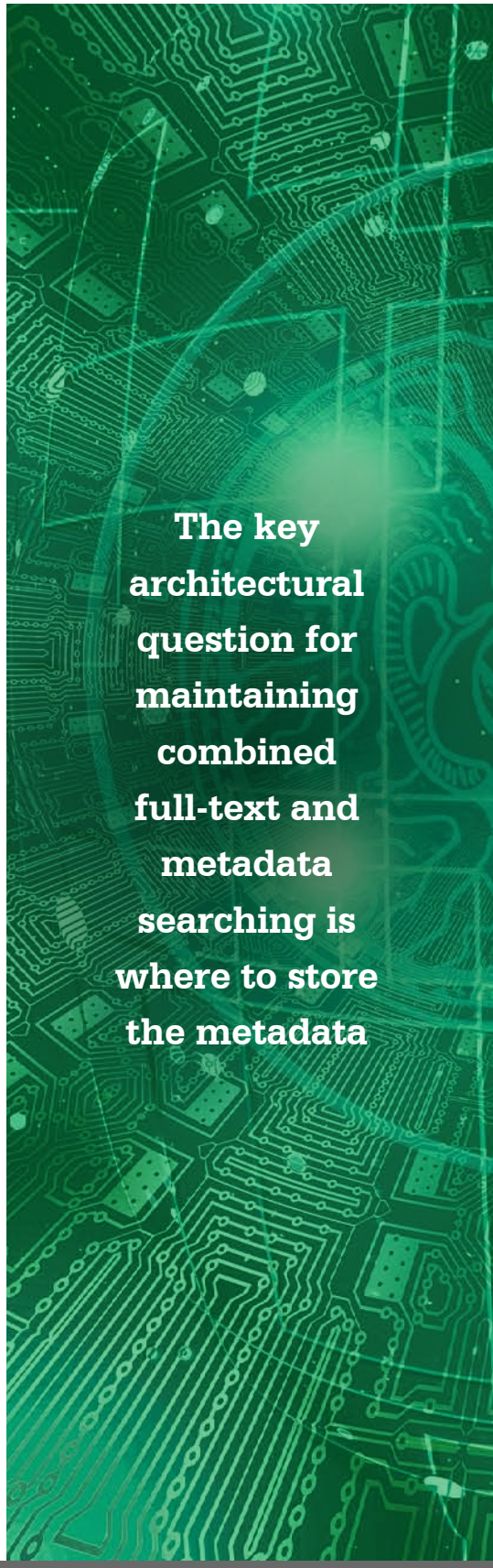
3. Maintain a separate database to store metadata.

Another option is to maintain a separate database like SQL, NoSQL, XML, SharePoint, etc., to store the metadata or facets where the database itself includes links to the referenced Documents when working remotely. A backend database is a particularly good option for more complex metadata structures. With this backend database in place, a search index can seamlessly cover both the back-end and referenced documents, allowing for integrative searching. But note that this enhanced integration only works if the same search index covers both the backend database and the separate documents; a "database only" search along with a "document only" search does not achieve the same results.

4. Store the documents in a database.

A final option is to store the documents themselves inside a database like SQL as BLOB data. The documents are stored right there in the database along with the metadata, enabling seamless integration of both for purposes of indexing and searching.

Article contributed by dtSearch®



Article contributed by dtSearch®

Some combination of the above options can also come into play. For example, a portion of the metadata may be stored in the documents themselves, another portion can be added "on the fly" while indexing, and yet another portion can be stored in a backend database.

Beyond end-user convenience and features like faceted searching, metadata can also be essential for a completely separate search-related reason: an enterprise requirement to filter retrieved data consistent with what the enterprise will allow the end-user to see. This consideration may not be relevant in the recruiter situation, but many datasets contain classified or sensitive information. The goal is to enable remote users to concurrently search the same dataset for "whatever," with the search mechanism granularly filtering search results consistent with each user's authorized purview.

The options for storing metadata for this type of search results classification are the same as the options for storing metadata for purposes of search efficiency. But filtering can also include full-text elements. Suppose ProjectABC is just one of many projects today but tomorrow becomes newly sensitive on account of updated disclosure requirements. The search engine can screen documents with ProjectABC as part of full-text data, even if ProjectABC is not explicitly referenced in associated metadata, for users without the authority to see ProjectABC information.