Join the search with John Curtis

by
Laura
Rutherford

Level: All
Works with: All
Updated: 01/02/2002

The modes of searching in Notes and Domino have continued to evolve to fit the needs of users. John Curtis, senior technical staff member, tells us what the most significant changes to search have been and what specific features make Notes and Domino search stand out. He also talks about some of Lotus's search products—such as the Lotus Discovery Server and Domain Search—and what's new with them and the rest of search for Rnext and beyond.

Can you summarize the history of Notes and Domino search over the years? Specifically, tell us how it has evolved and what factors have driven its evolution.
I wasn't around then, but I believe that back in R3, Steve Beckhardt did the project to put the Verity search engine into Notes. Verity was the original engine we used. What was needed was another way to find documents besides the traditional view mechanism (the mechanism that sorts a set of documents in a repository by design). We needed a free-form way of asking a question, such as "Give me all the documents containing a given word or words." Verity had an engine that did that, and a decision had been made to let its syntax go right through to the user. So all the field-based searches were all part of the engine. The development of the design of Notes with the Verity engine was part of that original project.

From V4.1 through 4.6, more and more services inside of Notes started using full-text search. Because we published the interface through LotusScript and Java, people could write applications now that used it. Once programmers starting using search, we had a new phenomenon showing up, called machine-generated queries. What happens in that type of query is that certain search boxes are exposed to people, but the query is decorated with "and this," "and that," "or this," and various other terms and constructs on top of the query. What the engine sees is a big long query that filters down to the right set of documents. That happened in roughly the 4.x timeframe. Also in the same timeframe, there were independently running agents that were built around full-text search.

Once search services became useful, people started using them a lot—to route mail, to move things around in their folder, and so on. They used search mechanisms to find things, move things around, and send messages. Once you have a set of documents, the sky is the limit. You can do whatever you want with those documents.

There was a facility built in 4.5 called Site Search, which was the first multi-repository index. You could get multiple databases and create a single index for them. You could find data across several collections of documents. In R5, that evolved into something called Domain Search.

Also, in the R5 timeframe there were several mission-critical applications—including routing mail, which is about as central as it gets in Notes—that started to use full-text searching. The LDAP lookup looked at the names and addresses to find the true e-mail addresses of those who were getting e-mail. If exhaustive search was implemented, the router waited for the full-text search before mail got routed. Well, that put us right in the white-hot center of Notes. Many applications started using the search services. So, the migration with search services has been from an add-on, nice-to-have, useful feature to something that is central to the way people do business.

What, in your opinion, have been the most significant changes to search over the years?
First of all, full-text search is only 15 years old or so, relatively young as technologies go. But, I have to say the most significant things that have happened in the last five or ten years are Web search engines. We are at a point today where, for instance, your grandmother could turn on a computer and the first application she would use is the search engine. That's a major change. The development of the Web as the richest repository of information in the world and the indexing of that repository are now a critical way of doing business or a critical way of finding data.

In Domino, likewise, you have had the growth of functionality in applications. That is, people have embedded search services that have been evolving over time. Once they find out that they have the capability, they build the application around it. The machine-generated queries are an important development in that people don't see the whole search string sent to the server, but in fact, that mechanism is used to qualify a search. You have a context that is invisible to you but on the back-end is very well controlled.

Google.com raised another bar in search with a novel page ranking scheme. It eliminates the need to write a complicated query by finding the best hits the first time. The advance in page ranking is something we are doing—really a Discovery Server specialty. Before Google, the basic notion had been that the more hits you get for a given term, the higher ranked the document is—those things are getting much smarter now.

Can you explain what the Lotus Discovery Server is and how it works?
The Lotus Discovery Server is a product and a set of tools that help people get their arms around the ever-growing body of data that they have every day in their companies. Also, Discovery Server helps people understand what other people know. The notion is that people who write stuff know about that stuff—or the people who get written to about stuff know about that stuff. You can use the same categories that you use to classify documents, to classify what people know.

For instance, if I write a lot of e-mail or if I post a lot of entries in databases on full-text search and those are identified as keywords, then the system will automatically identify me as someone who knows about full-text search. Actually, because every enterprise has its own distinctive data, it is not completely automatic—you do need somebody to camp out and mind the corporate categories and keep them current. Once you do that, what gets produced is an index of all documents plus a user experience that lets you browse the categories that your company has and drill down into different disciplines to find people and documents about given subjects, performing searches within those disciplines.

The Lotus Discovery Server contains a pretty powerful interface where you can go back and forth—you can perform a raw search of the whole body of information or you can drill down into the different taxonomic nodes within your tree of categories and search within those. It's the great knowledge management mantra put to code.

How does the Discovery Server work with Domino and Rnext?
The Discovery Server is sold separately. It takes the name and address directory information directly into its people repository. It takes directly from Notes that way. Also it crawls Notes databases, as they are one of the initial types of repositories it takes information from.

At the same time, it has independence and a mind of its own both from a product standpoint and from the evolution of where it is going. We run DB2 as well as Notes on the same machine. We have the ability to migrate data stores and environments to match those our customers are migrating to.

Can you give me an overview of what kinds of search services Notes and Domino have to offer?
Well, there is the basic free-text search, which is done by typing a few words into the search bar and clicking the Search button. Today, by default, we use phrase searching when two or more terms are specified consecutively; but if the keywords AND, OR or ACCRUE are specified, we apply the proper Boolean operation. Over time, this will evolve into using more Web-style semantics.

Even within free-text search with Boolean processing there are things like stemming (such as *farm* finds *farmer* and *farming*), fuzzy search (for misspellings), and results sorted by dates or relevance or left in view-sort order. Through our APIs, we offer "decorated" results, which include URLs, authors, titles, summaries, and other relevant document data. We also allow for wildcards—so you could put an asterisk before a set of characters or after a set of characters and we could find portions of words like that.

Then we start talking about things like fielded search. Fielded search is a key capability we have offered since we began. This capability allows users to specify terms or numeric/date-time values to be found within a field in a document. There are UI assists for these as well. For multi-repository indexes, we marshal universal field values like creation time, author, and title into fields that span all repositories. The future of this feature is to eventually allow user-specified fields across multiple repositories.

Finally we have numeric search, which lets you find data based on dates, times, or other numeric criteria.

If you pull all the search options together you have a very robust syntax with which you can let people search. There are many ways to get data out of an index and many flavors of results.

What are some of the search features that make Notes and Domino stand out?
We have a requirement that we keep things fresh. If you go to the Web, some of the Web search sites brag about being a week old. That is their goal. We go for every 15 minutes, if not faster. Our requirement is that we keep the index as fresh and up-to-date as possible.

Heterogeneity is another thing. We read things from disk. We handle attachments in Microsoft Office and some of those type forms, and do it intelligently—as well as files on disk. So we are kind of agnostic as far as what the data is coming in as.

We also offer global, multi-lingual support. We index documents written in any language together in the same index and allow users to search that index in any language. This is unlike many search engines that make people go to a French-only index, say, for French content.

We also do fielded search across repositories. There are universal fields like author, title, summary—those kinds of things—which we can marshal from the incoming data. That's true across the repositories.

The rich programmability of the interfaces is unique in my experience. We offer more comprehensive flavors of search and results than anyone. Our cross-repository, attachment indexing, and file system indexing are unequaled.

What are some of the overall goals you keep in mind when developing new features?
The biggest one is building things that people want. That sounds like a truism—why would you build something that people don't want? But in fact, if you don't take the care to filter what people are saying in such a way when developing a feature, you build something that some people want and some people hate. And that is something we have to play with all the time. You have to think things out.

Another goal is to minimize wait time. You can't sacrifice a user's wait time. The one thing people value more than anything these days is time. It is the number one resource. We cannot, for the sake of our feature, make it anything that sacrifices time.

Also, you have to be good server citizens. By this I mean that we have to be almost invisible or increasingly invisible on a server so that we don't assume resources that someone else needs.

Finally, we avoid limits that benefit no one but our developers. If there are limits you need to get your job done—there are some administratively prudent ones that people want, such as limits on time taken for indexing a document. Other limits, which were there only because they made the software easier to write, are candidates for design revision and removal.

For Rnext, give us a taste of some of the new and enhanced features we can expect.
A lot of them are server side. They are features that would impact people's

lives, and a lot of them are performance based. People don't notice the performance enhancements necessarily, but they do notice bad performance. If they have noticed what they consider to be less-than-optimal performance in previous versions, that will be improved.

The search engine now uses the Unified Buffer Manager (UBM), which is a way to share data and memory. We are able—across threads—to do a single search and share data among many users. So common searches will be in memory already. Notes databases used the UBM extensively in R5. Now the search engine is using it as well.

We have a new field option called Field is Present. It tells you whether or not a field is in a document or not. This feature helps you understand your documents better—the absence of a field helps you filter documents.

We did a lot of work on LDAP searching—did a lot to make that faster. Routing mail through LDAP should be a much more economical process.

Also, we have a capacity improvement. We can go up to 16 terabytes per physical index. Of course, Domain Indexes will approach that size first. We've eliminated the 6 MB maximum document size, which kept large documents from being indexed before. This is an example of a limit that got in the way of indexing, where we had a hard, though settable limit of the maximum document we would index. In Rnext, we will index documents in roughly 500K chunks, greatly decreasing our memory footprint.

Tell us a little about the GTR engine and its history in Notes and Domino. Also what is new with the GTR engine for Rnext?
In the R5 timeframe, we were in the throes of replacing the search engine with Global Text Retrieval (GTR). GTR is called an N-gram engine. N-gram is a way of indexing text where you take a word and break it up into components. So you take uniform size components, and in this case, they would be letters. For example, my last name (CURTIS) would be broken up into CUR, URT, RTI, and TIS and stored each as separate chunks of words. Then at query time the N-gram engine pieces together words based upon the equal-size word fragments and gets the hits for the words.

You may say, "Why would they do that? Seems like a lot of front-end nonsense." But in fact the N-gram preprocessing is so fast you can't measure it. Secondly, it makes things easier, particularly in the area of DBCS [double-byte character set] data where you have no word breaks, like in Japanese or Chinese where you have no spaces to delineate the words. N-gram is a wonderful approach because it creates sequences of words. It creates artificial words. And at search time, you piece together the words—so it kind of forces word breaks.

N-gram is also very good at things like fuzzy searches. If you say I want a 60 percent hit or wildcards, it is extremely fast at that. For instance, if you do a wildcard, say you type in **nation**, you can find the letters *n-a-t-i-o-n* surrounded by anything else and you are done. N-gram is markedly faster than other indexing schemes at that kind of thing.

In Rnext, during indexing, GTR will also update its largest files in-place. In R5, the previous version of GTR would make a working copy of the largest files, update that, and then erase and rename the file to replace the old one. No more. They now do update in-place. Also, they're using the UBM to provide much better memory management and caching during search.

Also, the Boolean process is much enhanced. GTR is making very good use of partial search results. If you have a complicated query and part of that query finds five documents and the other part finds ten million, the five documents are used very intelligently to filter the ten million. It's a drastic improvement. The scalability has also increased to 16 terabytes.

What about Domain Search? Can you explain how it works and what, if any, changes we can expect in Rnext?
Domain Indexing and Search is a capability introduced in R5 that allows data from many repositories to be indexed together in one central index. These indexes are the largest ever created in Notes/Domino.

The feature evolved from Site Search (from the R4 timeframe). The decision was made to leverage the improved Notes catalog, which is a database that already contains information about databases in the domain. In R5, it is up to administrators to proclaim a server to be the domain server. This server can or cannot have an index, but it certainly has the catalog that contains the domain information. We use that catalog to crawl the databases on the domain to get all the information to a central place on that machine. There are eight physical indexes that get created from that. From that, we now have a logical central index to query against. We do tell people to dedicate a machine, because it is a large index and you really need a lot of memory and horsepower to get the job done.

What we index is the data of every document coming in. We also marshal data into title, author, category, and so on. You can also keep a file system on that and index that as well.

So Domain Search bundles the file system into the index as well?
Yes, it does. You can search Notes content and file system content in a single query.

Can you explain what LDAP search options Notes and Domino have, and what's new for Rnext?
LDAP search for R5 had been a machine-generated query where you had up to 12 word terms searching across different fields. We did admirably, but that is a big query. We made several changes inside the GTR engine including the way memory is managed, the UBM, those kinds of things. Plus, we took those 12 fields and we marshaled them into just a couple. Every name and address field, for example, are put together into a single field that is queried by itself. So the query being generated is a lot simpler. And we leverage all the things that were improved in GTR Boolean processing. What we did with the fields is going to be, we believe, the basis for cross-repository fielded search and Domain Search for future releases.

Where is Notes and Domino search headed post Rnext?
We have a set of big projects that we have to go after, such as a fielded search that crosses repositories. Now this is a trick because people have different designs out there and people have designed their databases to have different field names that mean the same thing. So, potentially, huge mapping has to happen.

Parallel search is another thing. Whether it happens on Discovery Server or Domain Search, true parallel search would have multiple independent machines that produce marshaled results and you get the results in a central place and merge them.

There are many administrative features, such as backup recovery, that we may add. It takes typically three to five days to build a domain index, sometimes more, depending on how big your domain is. You can't spend that much in machinery resources and not back things up. People can back up by hand, but they would like us to develop a method by machine.

People have also asked for various types of partial index capabilities. Maybe they don't want to index every single field; they want to index only certain things. In the future, we may be putting some partial index capabilities in.

ABOUT JOHN CURTIS
John has been working in database internals for over 20 years. He’s worked on everything from compilers to device drivers, but most of all likes building server software that does very useful processing and performs like greased lightning. He started at Iris in 1998 as team lead of full-text search and the team has never been the same. He is the husband of one and father of three. In his spare time, he produces sacred music and drama, and studies theology.