Article
Kyle Baxter · Sep 9, 2016 5m read

Free Text Search: The Way To Search Your Text Fields That SQL Developers Are Hiding From You!*

Have some free text fields in your application that you wish you could search efficiently? Tried using some methods before but found out that they just cannot match the performance needs of your customers? Do I have one weird trick that will solve all your problems? Don ' t you already know!? All I do is bring great solutions to your performance pitfalls!

As usual, if you want the TL;DR (too long; didn ' t read) version, skip to the end. Just know you are hurting my feelings.

If you open up your version of Sample.Company in the SAMPLES namespace of a recent (2015.1 or later) Caché/Ensemble/HealthShare version you will see a Mission field that is pseudo-randomly generated text. Suppose we want to search this text field. For the purpose of this exercise, I generated about 256,246 companies – feel free to populate some on your own and follow along. Well you might run the following query:

```
SELECT * FROM Sample.Company WHERE Mission LIKE '% agile %'
```

That is quite the reasonable query, but how does it perform? Well, with no index it clearly has to read each entry, so we get 256,277 global references for our 7854 rows returned. This is not very good! Let ' s add an index to Sample.Company and see if we can do better. We add the following:

```
Index MissionIndex on Mission;
```

And we build the index and run the same query. What do we see? 279,088 global references.

Ah, I can hear you over there: " But Kyle, that ' s MORE global references! Isn ' t that bad? I thought indexes were supposed to help!!! "

Well slow down there, 3 exclamation points is just too many. Besides, when confronted with some inconsistent behavior, it ' s best to take a moment and think. What is the cost of reading an index vs. reading the full map? Well an index is going to be smaller, so reading the full MissionIndex is going to be a cheaper operation than doing a full table scan. Then we only need to read pieces of the full table for display. So while it ' s more Global References, it ' s less work (assuming everything is on disk). There is, of course, a lot more to this behavior but I would have to write extensively on the Caché Block structure and that ' s just WAY outside the scope of what I ' m trying to accomplish here.

OK, so our first crack and cheapening our query might have worked, but certainly less than we ' d like. We want lightning fast! We want less global references, and we want it easy! What can we do? The answer is an **iFind Index**.

Alright, I know you don ' t know anything about iKnow and this iFind thing sound suspiciously like iKnow. You want an SQL solution and you really don ' t want to learn the ins and outs of a whole new technology to get the performance you want. Don ' t worry, although iFind leverages the iKnow engine, you need precisely **ZERO** knowledge of iKnow to be able to use this. Check it out, you define the index like this:

```
Index MissioniFind on (Mission) as %iFind.Index.Basic;
```

What does %iFind.Index.Basic mean? Doesn ' t matter! Isn ' t that beautiful!? Just put it in and build the index the way you always build indexes (%BuildIndices()). Now, with this index there ' s a bit of a change in our query. We need to tell the query to use this new fancy index like so:

```
SELECT * FROM Sample.Company WHERE %ID %FIND search_index(MissioniFind,'agile')
```

The only thing you are entering, is the name of the index and the word you are looking for. OK so that ' s a lot to do, add an index, build it up, and use some funkier SQL than you might be used to. Is it worth it? Well this query gets the same 7854 rows back, but does so in 7928 global references.

7928 GLOBAL REFERENCES! That ' s barely more Global References than rows! That ' s some good stuff. Now I can tell you all have some questions, so I ' m going to not only predict the questions but answer them!

Can we combine this with other indexes? Great question! The answer is YES! Index combinations work really well with this technology.

Are there any restrictions? Sadly, yes. You need to have a bitmap enabled ID. That is, the ID for your table needs to be a positive integer. This means no compound IDs, no string IDs, etc.

Does it really work that well? BETTER! If you have a free text field and haven ' t at least tried an iFind index, you are doing yourself a disservice!

So what about iKnow? The iFind index is powered by the iKnow engine, however you don ' t need to know anything about iKnow to be able to use iFind in your existing applications. Just define, build, and use!

Where can I learn more? Documentation of course! Here:
http://docs.intersystems.com/latest/csp/docbook/DocBook.UI.Page.cls?KEY=GIKNOWifind

And Here:

http://docs.intersystems.com/latest/csp/documatic/%25CSP.Documatic.cls?PAGE=CLASS&LIBRARY=%25SYS&CLASSNAME=%25iFind.Index.Basic

Does iFind do other cool stuff? Of course! It ' s powered by iKnow so it can do a lot! You can do Fuzzy Search, Stemming/Compounding, Ranking, and you can look for iKnow entities. If you ' re interested, take a look at the %iFind.Index.Semantic and %iFind.Index.Analytic index classes. They use more iKnow bells and whistles than the Basic one we used above. Or, check out this DC post on a demo interface that works on top of any iFind index: https://community.intersystems.com/post/iknow-demo-apps-part-5-ifind-search-portal .

Most of this is beyond the scope of this post, but if you want to see more details on any of these subjects, just ask! You all know I ' m here to inform and enlighten!

TL;DR: If you want to do search on a free text field add an iFind index by defining it like so:

Index <IndexName> on <FreeTextField> as %iFind.Index.Basic

Build the index using %BuildIndices (as normal)

Rewrite your query like so:

…WHERE ID %FIND searchindex(<IndexName>,<Search Value>) AND …

And reap the benefits of lightning fast free text search!

*No one is hiding this from you! Come on now, you know by now I just do it for the clicks!

#Best Practices #iFind #Indexing #Object Data Model #ObjectScript #SQL #Caché

Source URL:https://community.intersystems.com/post/free-text-search-way-search-your-text-fields-sql-developers-are-hiding-you