

Architecture and Administration Basics

Workshop Day 1 - FTS



1

Introduction



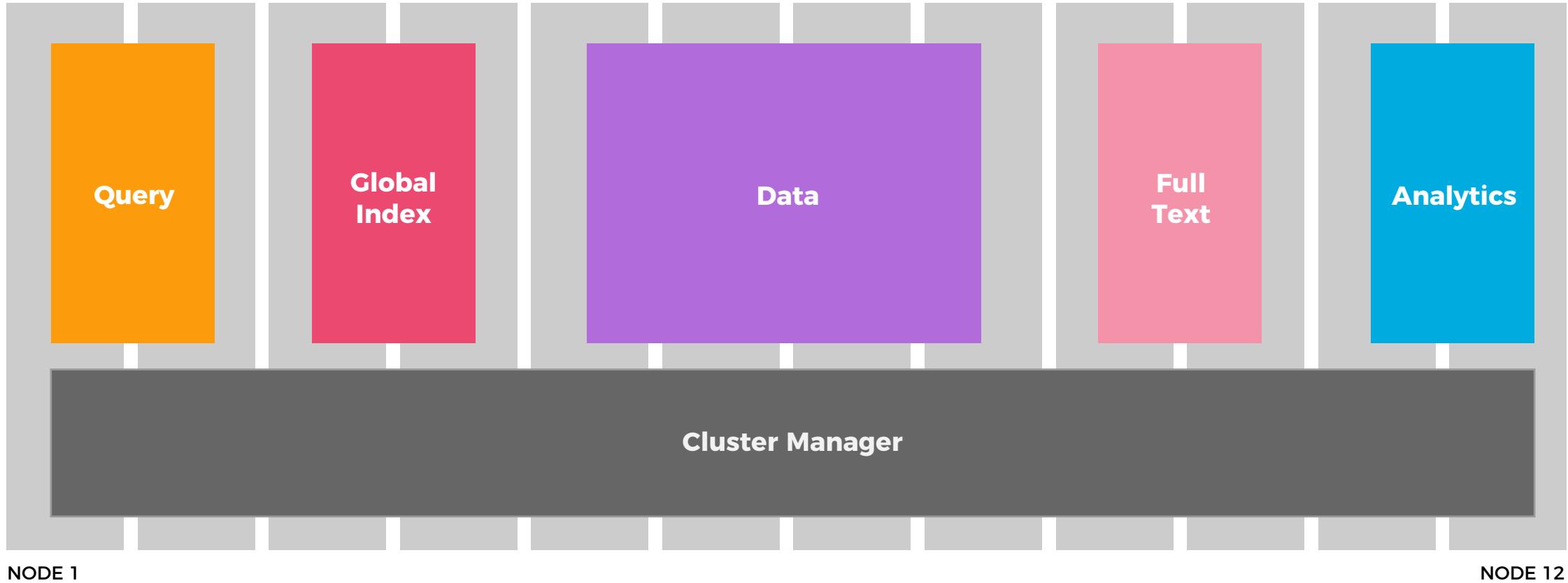
COUCHBASE

SCALABLE DATA PLATFORM





Multidimensional Scaling





2

Full Text Search Overview



Search is pervasive...



Born-digital businesses are relentlessly innovating to provide better experiences

amazon

ebay

PayPal

Google

LinkedIn

facebook

YouTube

"Search" is foundational to all born-digital businesses...

 Search... Search PayPal for... Search Facebook Search

Bleve - Engine behind FTS



Bleve is acronym of “Boiling liquid expanding vapor explosion”

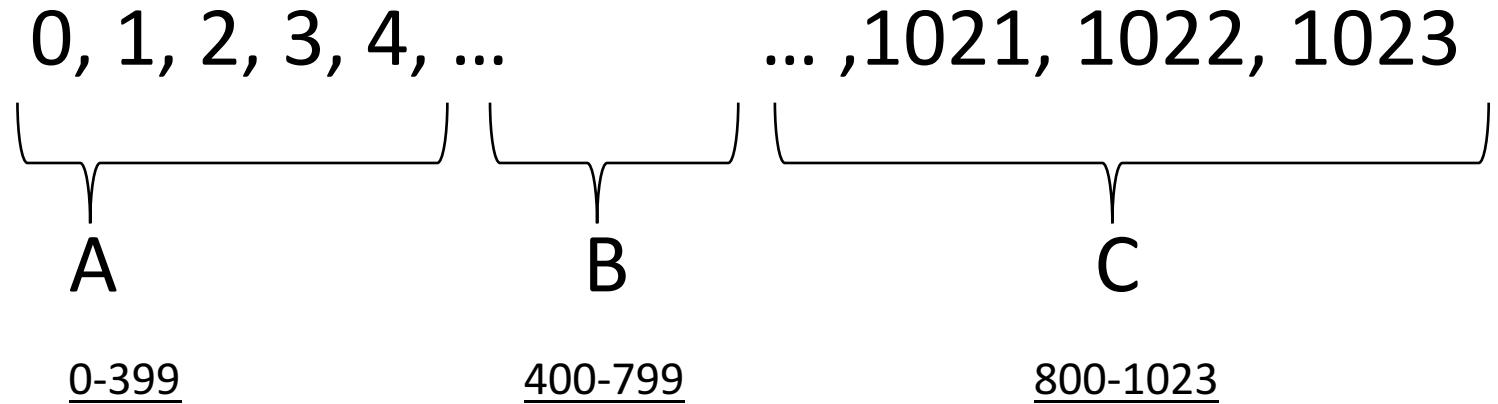
Bleve - open source full text search and indexing library written in Go - <http://blevesearch.com>

Simple, Powerful search engine with text analysis, facetting, scoring capabilities.

FTS design / index partitioning



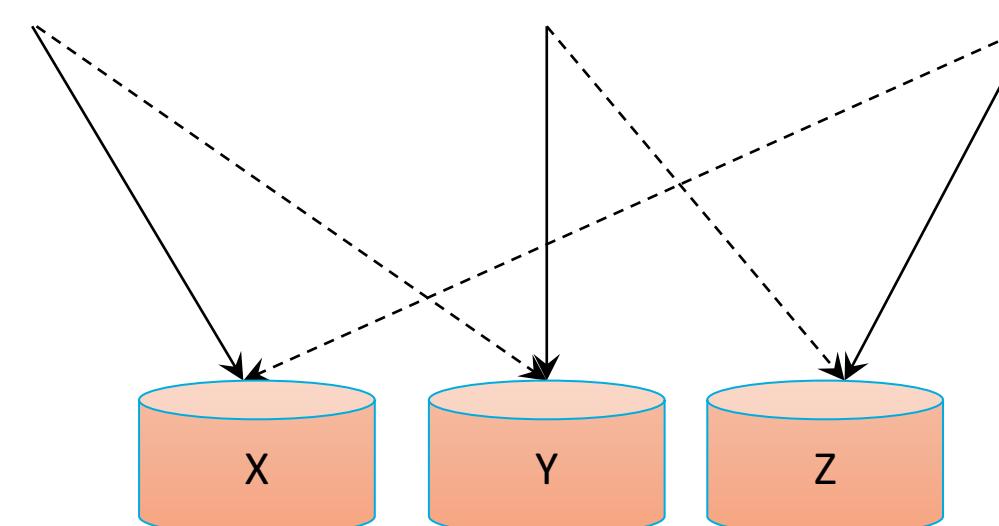
bucket partitions:
(1024 vbuckets)



index partitions:
(groups of vbuckets)

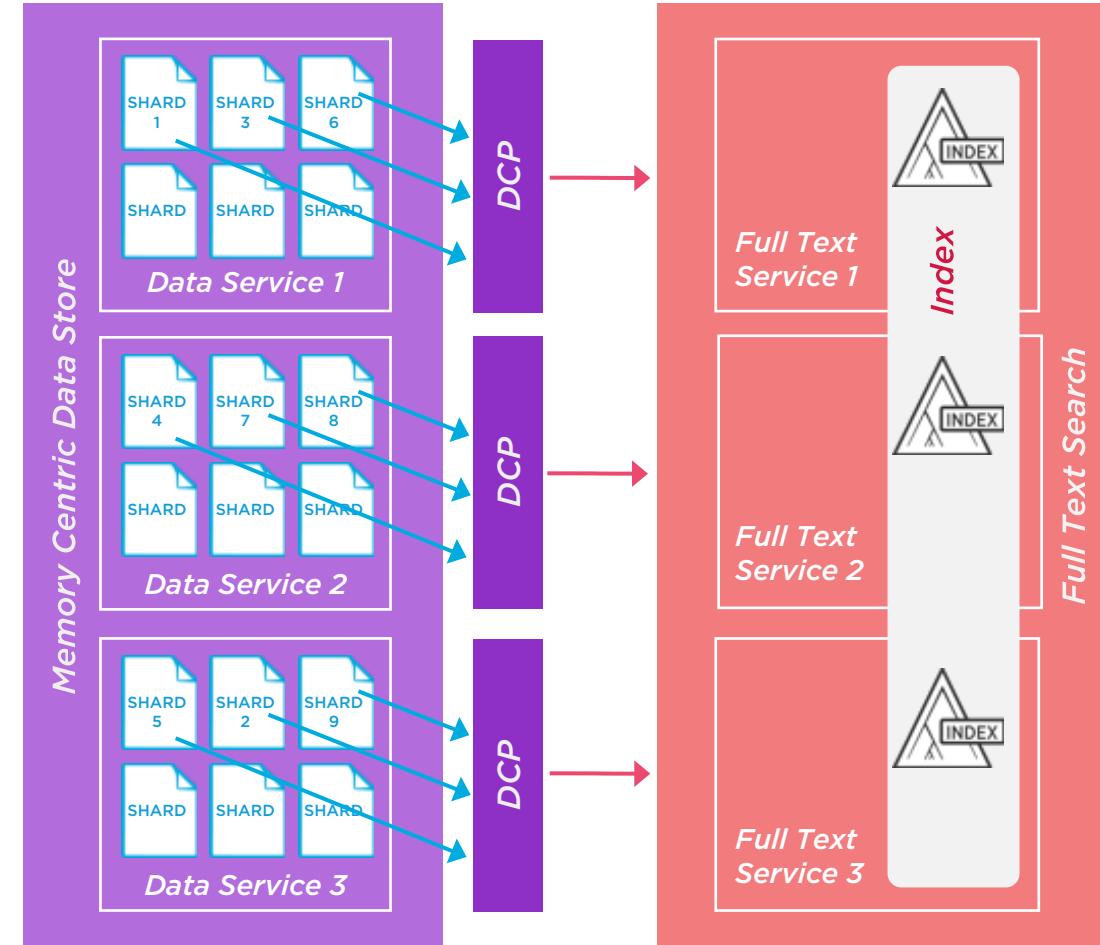
assign to FTS nodes:
replicas, too:

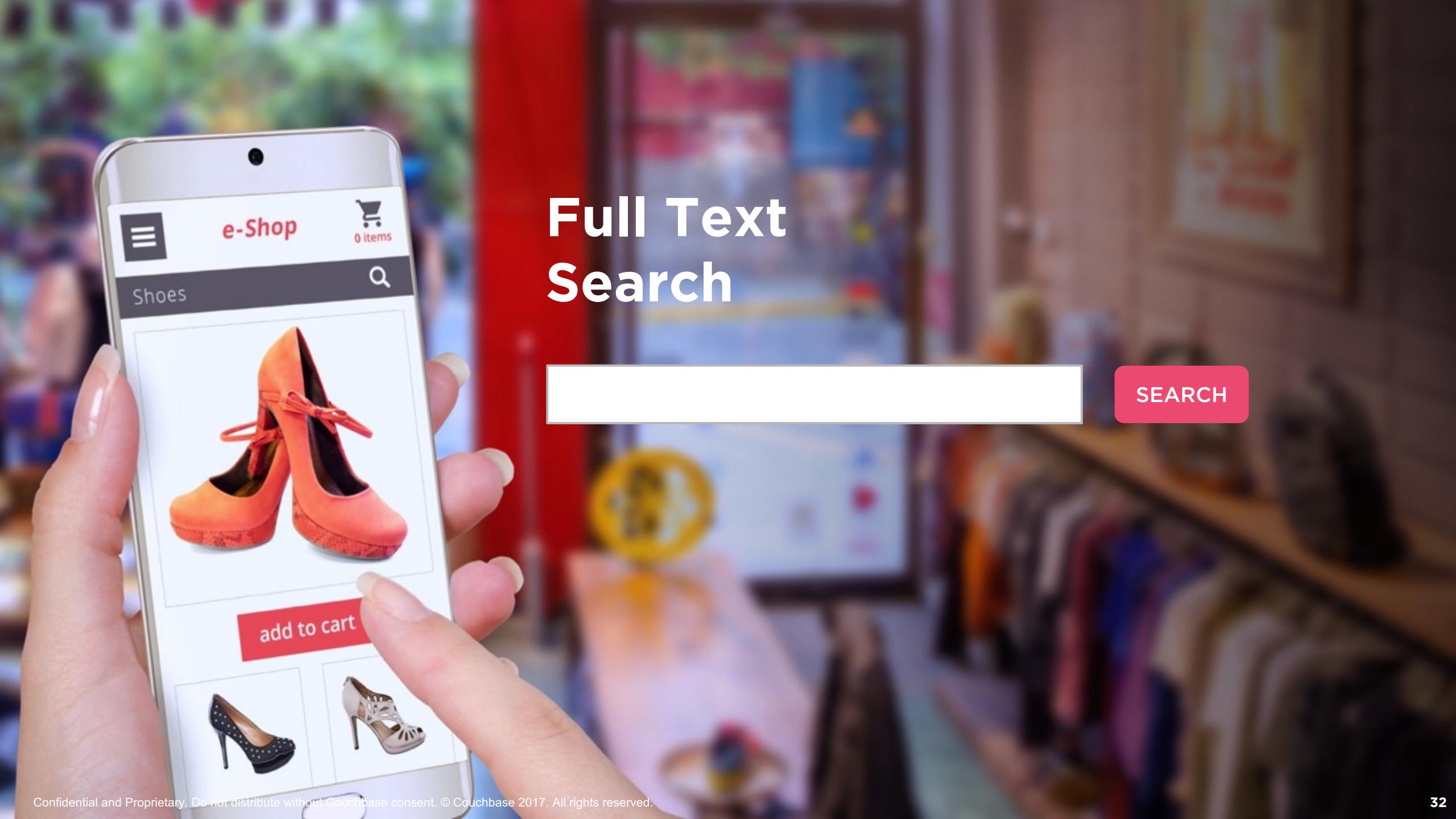
FTS nodes:



Multidimensional Scaling

- Each FTS node a DCP stream subscriber
- Indexing distributed across FTS nodes
- Any FTS service can receive queries
 - “scatters” to other nodes
 - “gathers” response
- Application sees single logical Index





Full Text Search

SEARCH



Full Text Search

Basic document search use case

- Query String

search couchbase docs...

SEARCH



Full Text Search

Basic document search use case

- Query String
- Term matching
- Scoring
- Context snippet

SEARCH

Search results

Scoring	Document ID	Description Matches
1.88	hotel_1234	best <u>location</u>
1.82	hotel_2345	loved <u>hotel</u> <u>location</u>
1.37	hotel_3456	<u>location</u> is awesome
1.25	hotel_4557	hard to <u>locate</u>



Index Analyze Search

1. Index fields of a document

"...located in the heart of the new City Quay development. The hotel has views of ..."

2. Analyze terms for index

located ... heart .. new City Quay development. .. hotel has views

3. Query the index

description: location

SEARCH

Return scored documents list

Scoring	Document ID	Description Matches
1.88	hotel_1234	best <u>location</u>
1.82	hotel_2345	loved <u>hotel</u> <u>location</u>
1.37	hotel_3456	<u>location</u> is awesome
1.25	hotel_4557	hard to <u>locate</u>



3

Full Text Search Concepts & Configuration

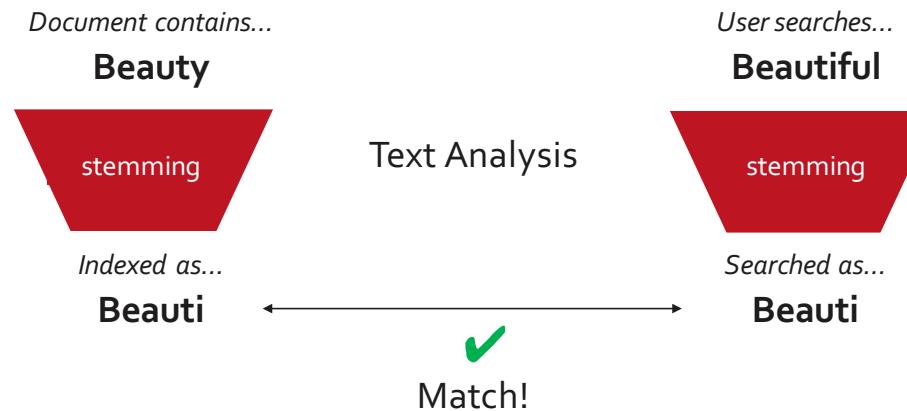
Underlying concepts



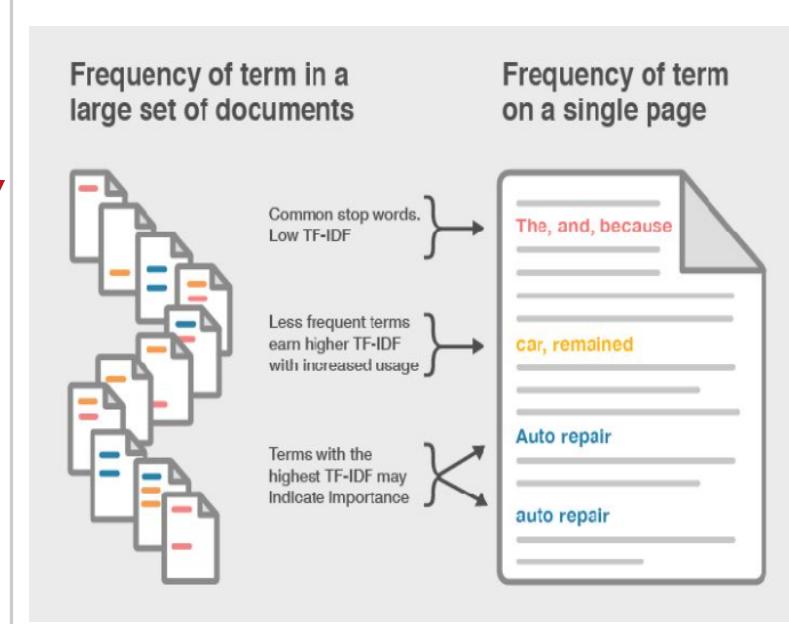
Inverted indexes

Terms	Where found
my:	Doc 1, Doc 2, Doc 3
dog:	Doc 1, Doc 2, Doc 81
has:	Doc 1, Doc 2, Doc 3
fleas:	Doc 1, Doc 81
...	

Language awareness



Scoring



1 - 2 - 3 step approach to FTS



Index



Identify document type

Exclude fields / sub-sections

Configure index behavior per field



Analyze



Remove undesirable characters in input using **character filters**

Split Input String to token streams using **Tokenizers**

Process token streams by chaining **Token filters**



Query



Query using SDK, REST API or web console

Optionally query using Field Scoping(:), Boosting(^)

Sort query results by any indexed field or score

Full Text Search – Index Identifier



- Default Index Mapping refers to the index mapping that Couchbase Server uses for JSON documents that don't match a more specific document mapping based on document type
 - Used to ensure FTS is working
 - Not very selective, very large index
 - Slow or result in high load if used in production
- Type Mapping – user-supplied "type" field determines where to find the type of each document
 - **JSON type field:** Specify the field in the JSON document whose value determines the type of the document. Defaults to "type".
 - **Doc ID up to separator:** The type identifier is the prefix of the document key, up to but not including the given character.
 - **Doc ID with regex:** Advanced users can specify a regular expression to match the type identifier.

Type Identifier

JSON type field:

Doc ID up to separator:

Doc ID with regex:

Full Text Search - Index Mapping



- Type mappings vs default mapping
 - Disable default mapping if you are only going to index specific types/fields
 - Disabling a type mapping can be used to ignore documents of a certain type
- For any type mapping, you can insert a child field to index the values in your JSON document with more control about what is indexed
 - "field" in index mapping refers to a name-value pair in JSON whose value is a simple type: string, number, true, false, or null
 - Child mapping allows you to index specific JSON sub-elements

▼ Type Mappings

+ Add Type Mapping

<input checked="" type="checkbox"/> # beer only index specified fields
description text index store include in _all field include term vectors
<input type="checkbox"/> # default disabled dynamic

Full Text Search - Index Mapping



- For a given “field” (name of name-value pair in the JSON document)
 - type:** Defaults to text, but other possible values are object, number, datetime, and disabled.
 - searchable as:** If a user limits their search to a specific field they would use this value instead of the actual name of the field in the JSON
 - analyzer:** The analyzer to use for this specific field

beer | only index specified fields

field	description	ok
type	text	cancel
searchable as	description	delete
analyzer	inherit	

index store include in _all field include term vectors



Full Text Search - Index Mapping

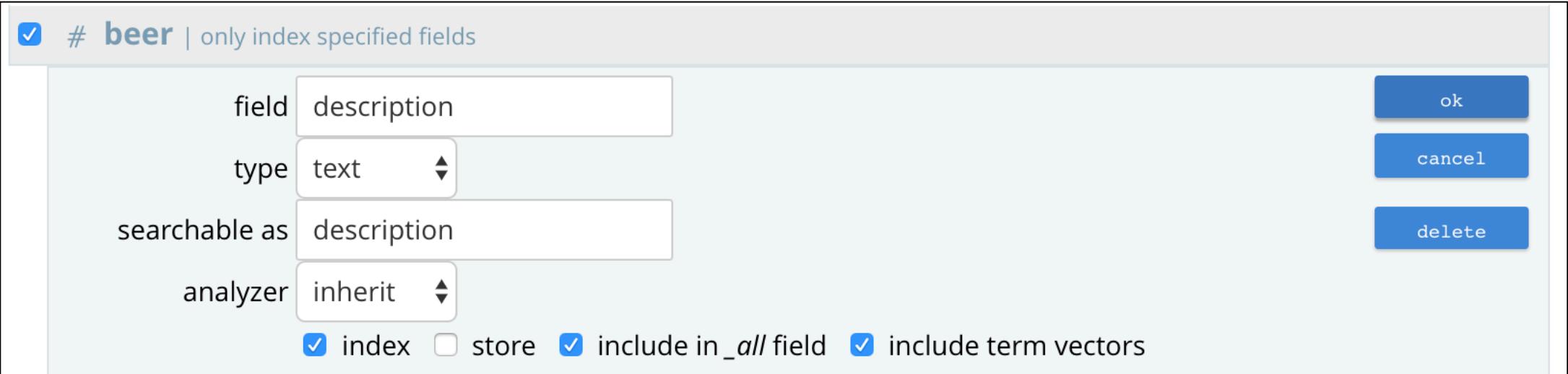


- **index:** If unchecked, fields that match this will not be indexed.
- **store:** By default only document IDs are written to a FTS Index; however, this allows the document contents to be written to the index.
 - Enables highlighting and result snippets but generally results in larger indexes that are slower to build.
 - Encourage use of multi-gets so users don't need to store the additional information in the index.
- **Include in _all:** The text in this field will be searchable in query strings without prefixing the field name.
If unchecked, the query must include this prefix (i.e. "description:modern").
- Not storing term vectors results in smaller indexes and faster index build times.

beer | only index specified fields

field	description	ok
type	text	cancel
searchable as	description	delete
analyzer	inherit	

index store include in _all field include term vectors





Web & REST Interface

Indexing

Name: FTSIndex1 Bucket: travel-sample

Type Identifier:

- JSON type field: type
- Doc ID up to separator: delimiter
- Doc ID with regex: regular expression

Type Mappings + Add Type Mapping

- # airline | disabled | dynamic
- # hotel | dynamic
- {} reviews | dynamic
 - content | text | index | store | include in_all field | include term vectors
- # default | disabled | dynamic

▶ Analyzers

▶ Custom Filters

▶ Advanced

Index Replicas ⓘ

0



Web & REST Interface

Searching

Best Hotel Location show advanced query settings
[full text query syntax help](#)

Results for FTSIndex1 917 results (20ms server-side)

Show Scoring

1. [hotel_27819](#)

reviews.content

◦ ...e **hotel** was very family friendly. A good breakfast was served daily; the staff was very friendly, professional and helpful. Staff responded to issues promptly. The **location** was close to main attractio...

2. [hotel_25160](#)

reviews.content

◦ ...Quarter, so when I tell you that the **hotel** is in a great **location** I know what I am writing about. You really cannot get a better **location** in the French Quarter that is convenient to all of the major a...

3. [hotel_1357](#)

reviews.content

◦ ...mes and this is by far the **best hotel** I have ever stayed in. As an American you are used to nicer hotels, but european hotels are....more like a bad Motel 6 experience to say the least. You put up wit...

4. [hotel_7769](#)

reviews.content

◦ ...die for.The **best** nights sleep that we had for the whole holiday. The **location** and proximity to Union Square is an added bonus as is the Cablecar stop right outside of the **hotel**. The **hotel** bar did clos...

5. [hotel_15956](#)

reviews.content

◦ ...s in San Juan for business. The **best** things about this **hotel** is its **location** in Old San Juan and the staff. You can walk everywhere in Old San Juan easily and the **hotel** is right on the bay so some roo...

Full Text Search – SDK



Python

```
termquery = fulltext.TermQuery('office')

results = bucket.search('travel-search', termquery, limit=25)
for result in results:
    ...
```

Java

```
TermQuery termquery = SearchQuery.term("office");

SearchQueryResult results = bucket.query(
    new SearchQuery("travel-search", termquery)
);

for (SearchQueryRow row : results) { ... }
```

Resources



1. 5.0 Beta - Full Text Search Intro Webinar video - "[FTS 5.0 Launch Overview](#)"
2. Presentation archive: <https://connect.couchbase.com/us/connect-new-york-2017>
3. Query Syntax: "[FTS Query Types](#)" - including JSON samples
4. Blog posts: https://blog.couchbase.com/search_gcse/?q=fts
5. Documentation: <http://bit.ly/2tHduXo> & <http://www.blevesearch.com/>
6. Forums - <https://forums.couchbase.com/c/couchbase-full-text-search>

Thank you



Couchbase