Architecture and Administration Basics

Workshop Day 2 – FTS (Full Text Search)

Full Text Search

Latency ← → **Flexibility**

- Very Flexible, but more concepts to learn initially

- Inverted Index

- Similar to Elasticsearch

Couchbase users need to search their documents

Full Text Search

search couchbase docs...

SEARCH

Full Text Search

| best hotel location | | SEARCH |

Search results

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

Index, Analyze, Query

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

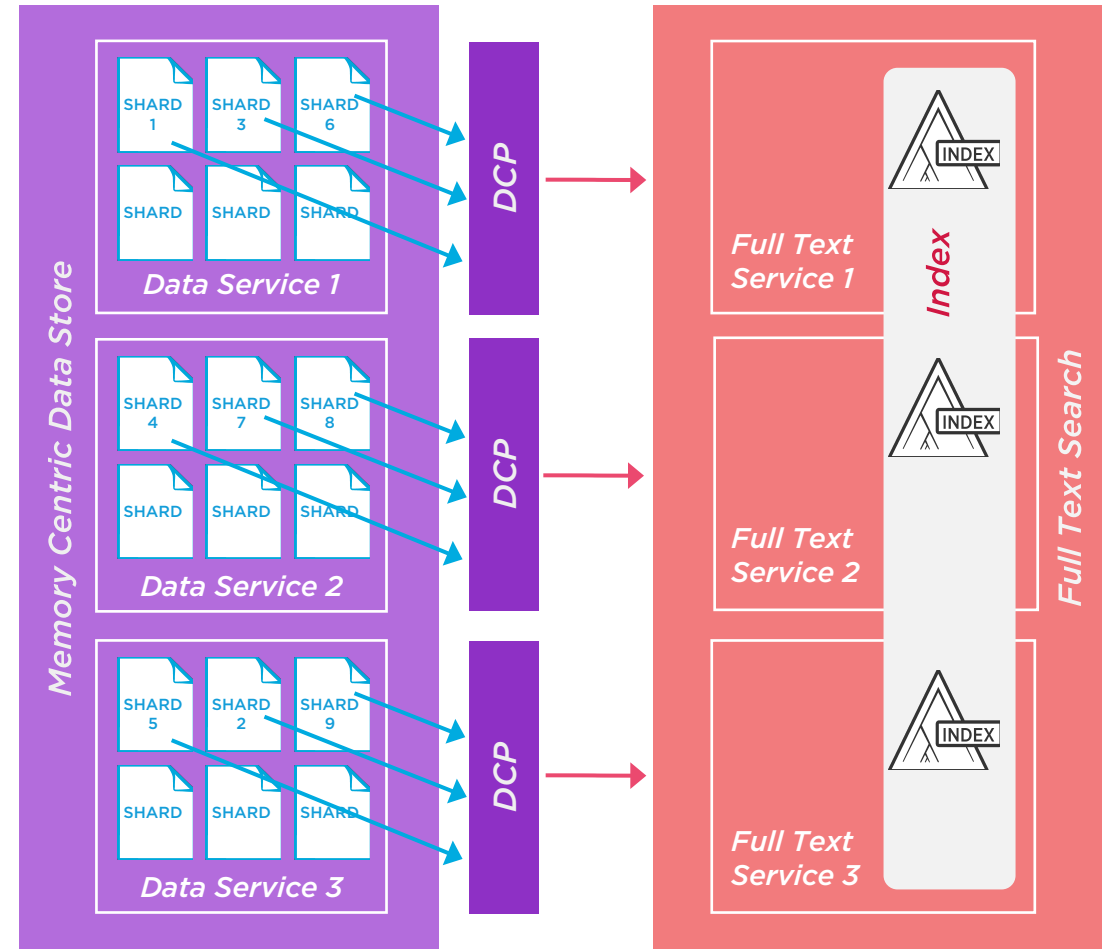3. **Query the index**
   description: *location*

Return scored documents list

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

Full-Text Data Projection

- Each FTS node is a DCP stream subscriber

- Indexing work is distributed across FTS nodes

- Any FTS service can receive FTS query, "scatters" to other nodes and "gathers" response

- Application sees a single logical Index

Memory Centric Data Store

Data Service 1
- SHARD 1
- SHARD 3
- SHARD 6
- SHARD
- SHARD
- SHARD

Data Service 2
- SHARD 4
- SHARD 7
- SHARD 8
- SHARD
- SHARD
- SHARD

Data Service 3
- SHARD 5
- SHARD 2
- SHARD 9
- SHARD
- SHARD
- SHARD

DCP

DCP

DCP

Full Text Search

Full Text Service 1

Full Text Service 2

Full Text Service 3

INDEX

Index

Full-Text Search - setup

Full-Text Search - search results

How does it work?

- Inverted indexes

- Language awareness

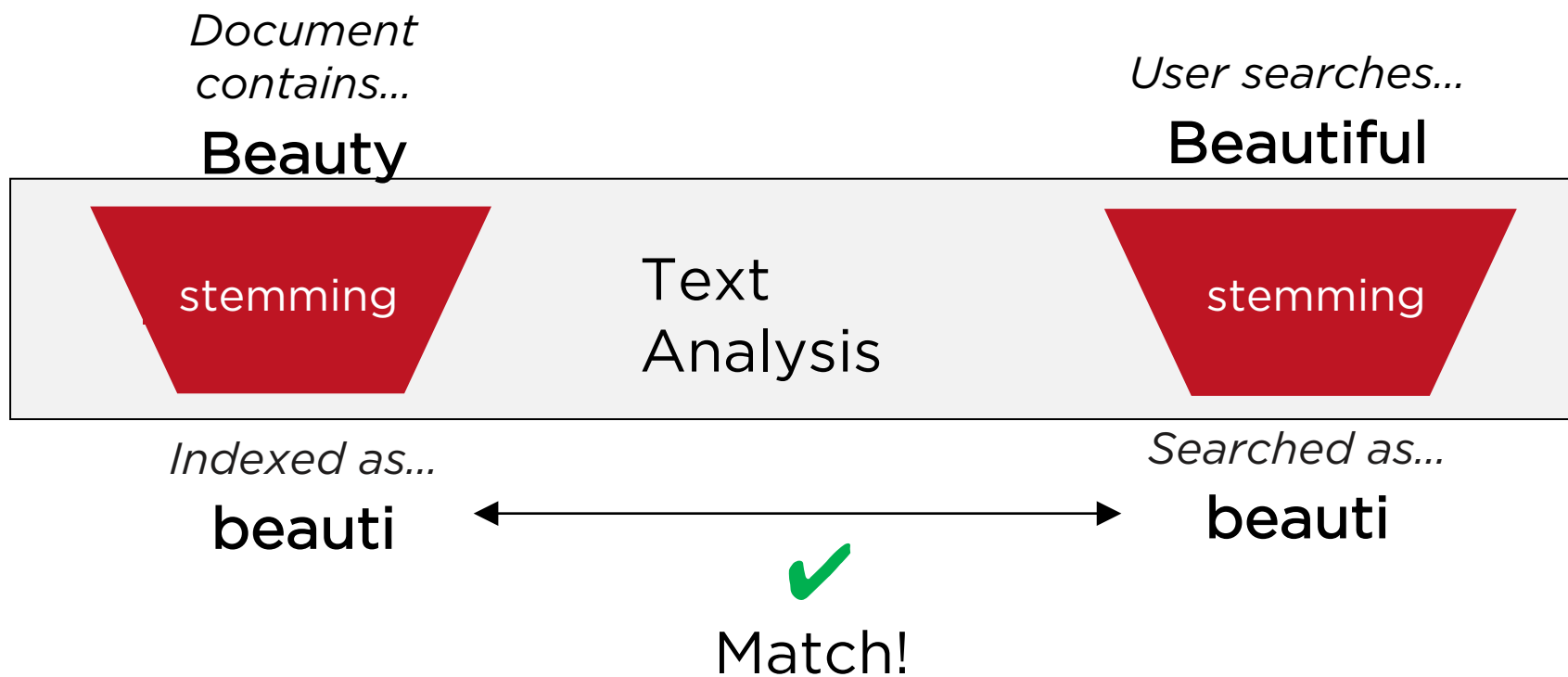- Scoring

Inverted index

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 aware

Document contains…

Beauty

Text Analysis

stemming

User searches…

Beautiful

stemming

Indexed as…

beauti

Searched as…

beauti

✔

Match!

Scoring

TF/IDF scoring

- **TF = Term Frequency**
  - How often does a term occur in a document?
  - More often yields a higher score

- **IDF = Inverse Document Frequency**
  - How many documents have this term?
  - More documents yields lower score
    - (because it means the term is more common)

Index mapping

▼ Type Mappings

default

repositories | text

status | text | index | store | include in _*all*_ field | include term vectors

createdAt | text | index | store | include in _*all*_ field | include term vectors

title | text | index | store | include in _*all*_ field | include term vectors

assignedId | text | index | store | include in _*all*_ field | include term vectors

id | text | index | store | include in _*all*_ field | include term vectors

type | text | index | store | include in _*all*_ field | include term vectors

{} developerInfo | only index specified fields

    email | text | index | store | include in _*all*_ field | include term vectors

    lastName | text | index | store | include in _*all*_ field | include term vectors

    firstName | text | index | store | include in _*all*_ field | include term vectors

Index mapping

- Exclude fields/sub-sections

- Configure indexing behavior by type of document (beer vs brewery)

- Configure indexing behavior per-field

  - Index Fields

  - Nested structures

  - Arrays

Index mapping

Before you search – you have to tell Couchbase how to index your docs...

```
{

   "json":"You know it!",       ✔

   "some_junk":"ignore me"      ✗

}
```

Type mapping

```
{

  "name": "21st Amendment Brewery Cafe",

  "type": "brewery",


   …
              Default type = no type or no type mapping for type
}
```

Simple types

Index as **"fields"**

```
{

  "type" : "brewery",

  "name":"21st Ammendment",

  "random_number" : 4,

  "edible" : false

}
```

Nested values

```
{

    "type" : "brewery",

    "geo": {

        "accuracy": "ROOFTOP",

        "lat": 37.7825,

        "lon": -122.393

    }

}
```

Use a **child mapping**, then index fields

Arrays

- Array of simple types

  Add a **field** for "pachyderms"
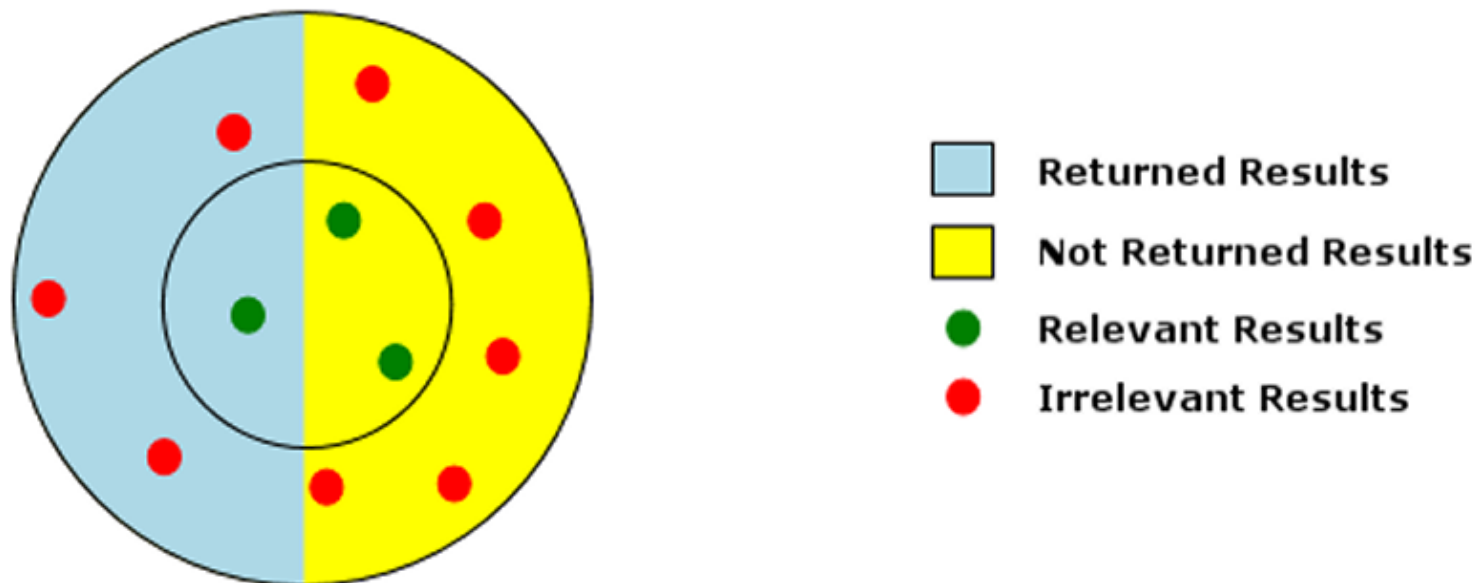
  ```
  {
    "pachyderms": [
      "elephant", "rhinoceros", "hippopotamus"
    ]
  }
  ```

- Array of Objects

  Use a **child mapping** for "alphabet"

  ```
  {
    "alphabet": [
      {"letter":"b"},{"letter":"c"},{"letter":"d"}
    ]
  }
  ```

precision vs. recall

Returned Results

Not Returned Results

● **Relevant Results**

● **Irrelevant Results**

- Precision – ratio of document matches that are actually relevant
- Recall – ratio of relevant documents that are actually matched
- High quality results depend on performing the right analysis for your text
- Beware: increasing precision may reduce recall (and vice versa)

How does it scale?

✔ auto index partitioning   (hash partitioning)

✔ to multiple FTS nodes   (auto-placement)

✔ rebalance   (add/swap/remove)

✔ scatter/gather queries   (partial results ok)

✔ replicas   (only primaries queried)

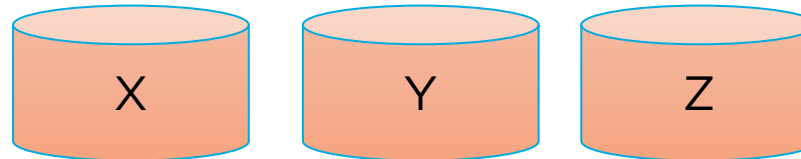✔ failover   (replicas promoted)

FTS design / index partitioning

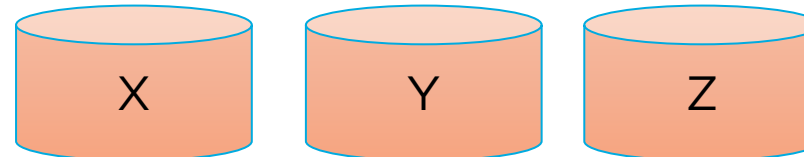bucket partitions: 0, 1, 2, 3, 4, ... ,1021, 1022, 1023
     (1024 vbuckets)

FTS nodes:

FTS design / index partitioning

bucket partitions: 0, 1, 2, 3, 4, ... ,1021, 1022, 1023
    (1024 vbuckets)

index partitions:

    (groups of vbuckets)

FTS nodes:

FTS design / index partitioning

bucket partitions: 0, 1, 2, 3, 4, …   … ,1021, 1022, 1023
      (1024 vbuckets)

index partitions: A   B   C
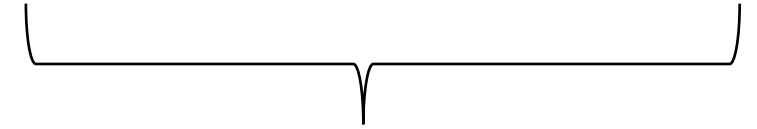
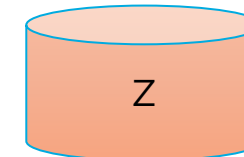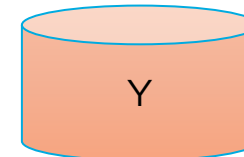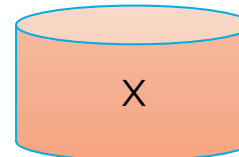   (groups of vbuckets)   0-399   400-799   800-1023
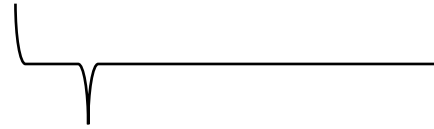
FTS nodes:

FTS design / index partitioning

bucket partitions: 0, 1, 2, 3, 4, ... ,1021, 1022, 1023
       (1024 vbuckets)

index partitions: A   B   C
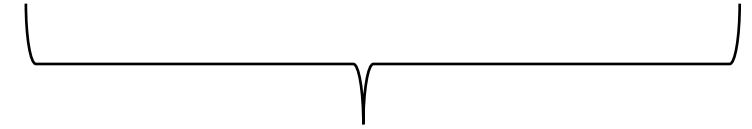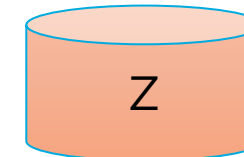   (groups of vbuckets)   0-399   400-799   800-1023
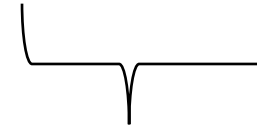
assign to FTS nodes:

FTS nodes:

FTS design / index partitioning

bucket partitions: 0, 1, 2, 3, 4, …   … ,1021, 1022, 1023
   (1024 vbuckets)

index partitions: A   B   C
   (groups of vbuckets)   0-399   400-799   800-1023
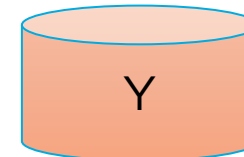
assign to FTS nodes:

FTS nodes:

X   Y   Z

FTS design / index partitioning

bucket partitions: 0, 1, 2, 3, 4, ... ,1021, 1022, 1023
   (1024 vbuckets)

index partitions: A   B   C
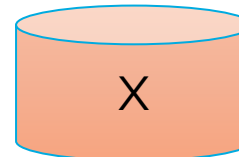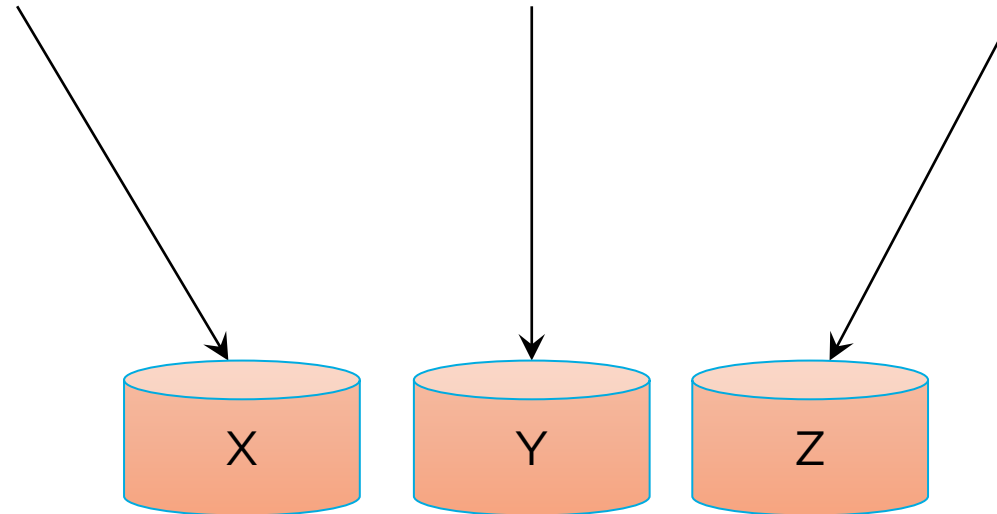
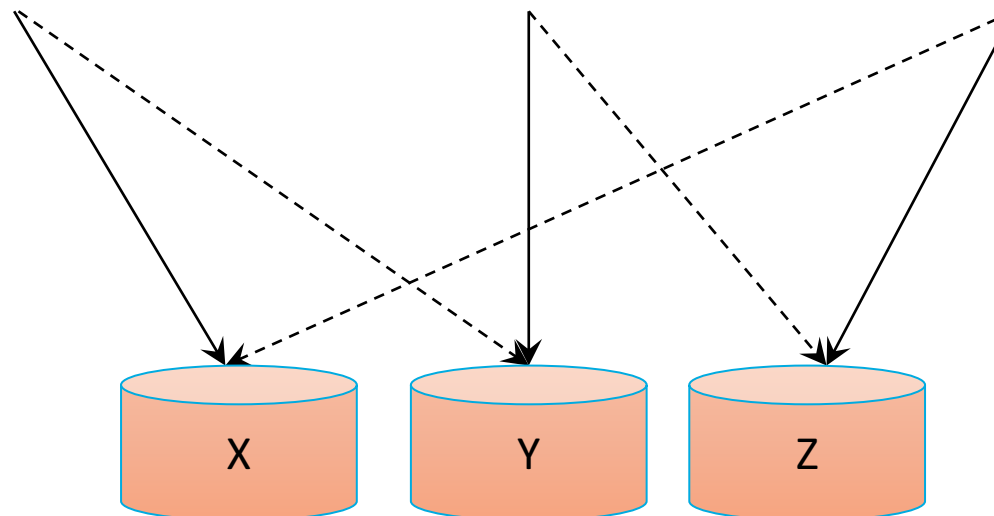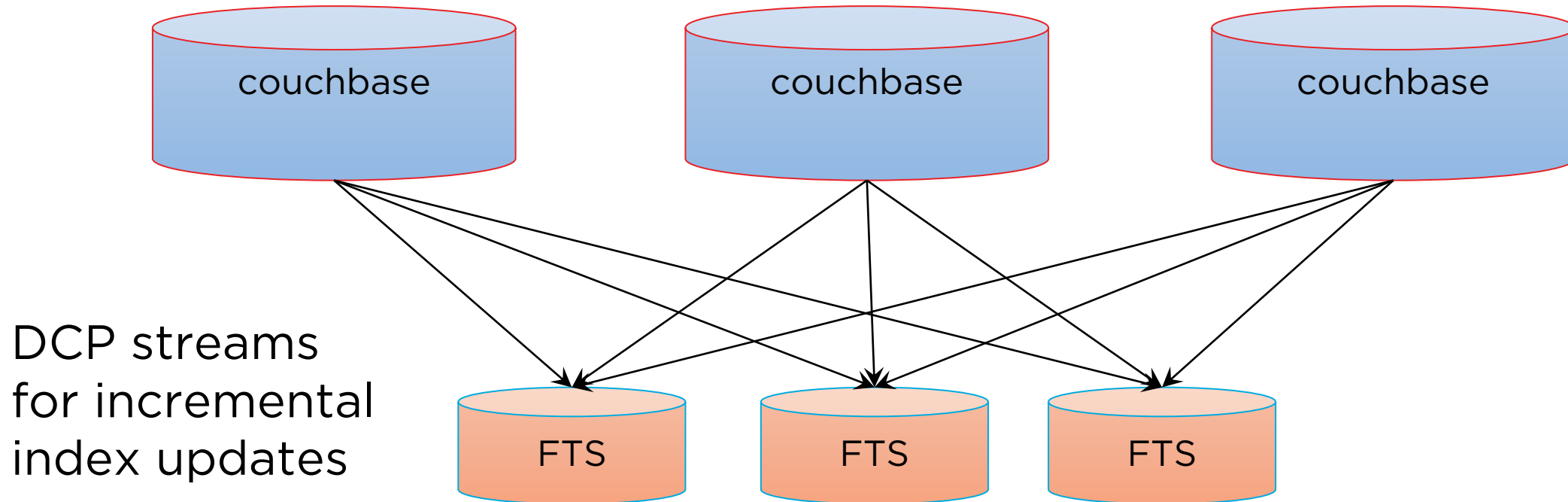   (groups of vbuckets)   0-399   400-799   800-1023
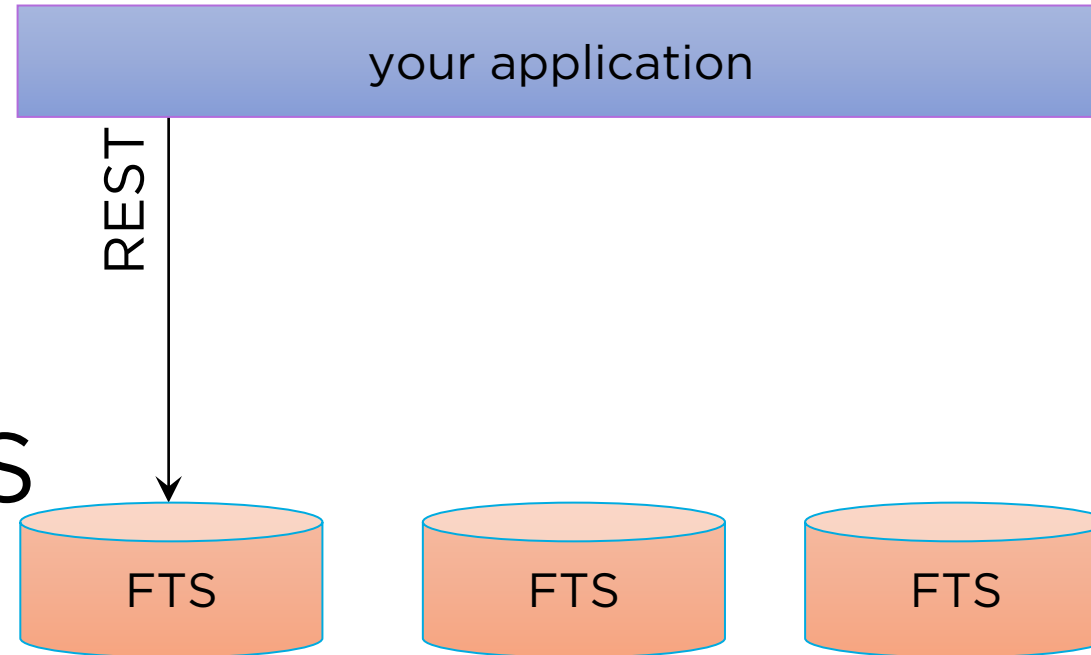
assign to FTS nodes:
   replicas, too:

FTS nodes:

FTS design / indexing

couchbase   couchbase   couchbase

DCP streams
for incremental
index updates

FTS   FTS   FTS

FTS design / queries

your application

REST

a query
sent
to any FTS
node...

FTS FTS FTS

FTS design / queries

your application

REST
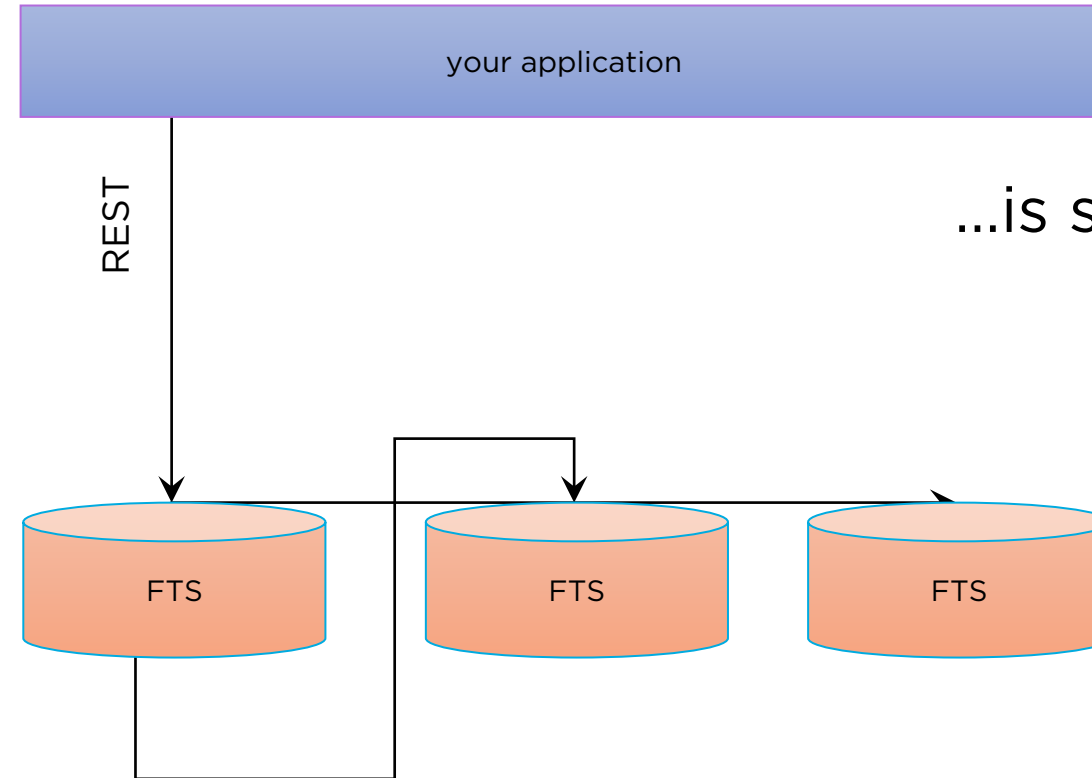
a query sent
to any FTS
node...

...is scatter / gathered
to the other
FTS nodes

FTS

FTS

FTS

Thank you

Couchbase