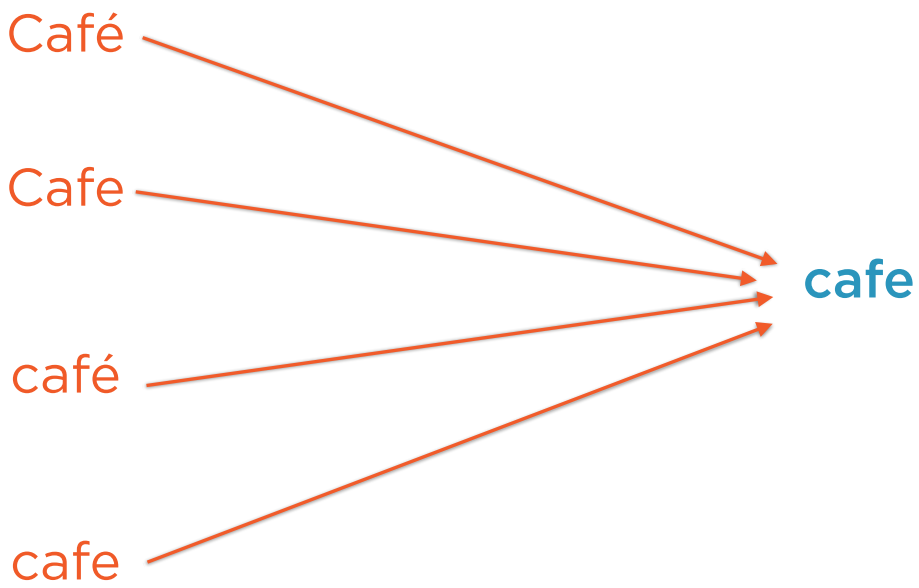
Creating a Non-default Text Index

Axel Sirota
MACHINE LEARNING ENGINEER
@AxelSirota

Handling Corner Cases

Text Indexes V3



The Diacritic Sensitiveness Case

{_id:1, beverage: Cafe con leche}
{_id:2, beverage: Café expresso}

We want to create a text index based on **field beverage**

The Diacritic Sensitiveness Case

Stemmed Word	ID
Café	2
Cafe	1
expresso	2
leche	1
con	1

The Diacritic Sensitiveness Case

Word	ID
cafe	[1,2]
expresso	2
leche	1
con	1

Mapping é, e to e and setting lowercase

Getting Some Coffee

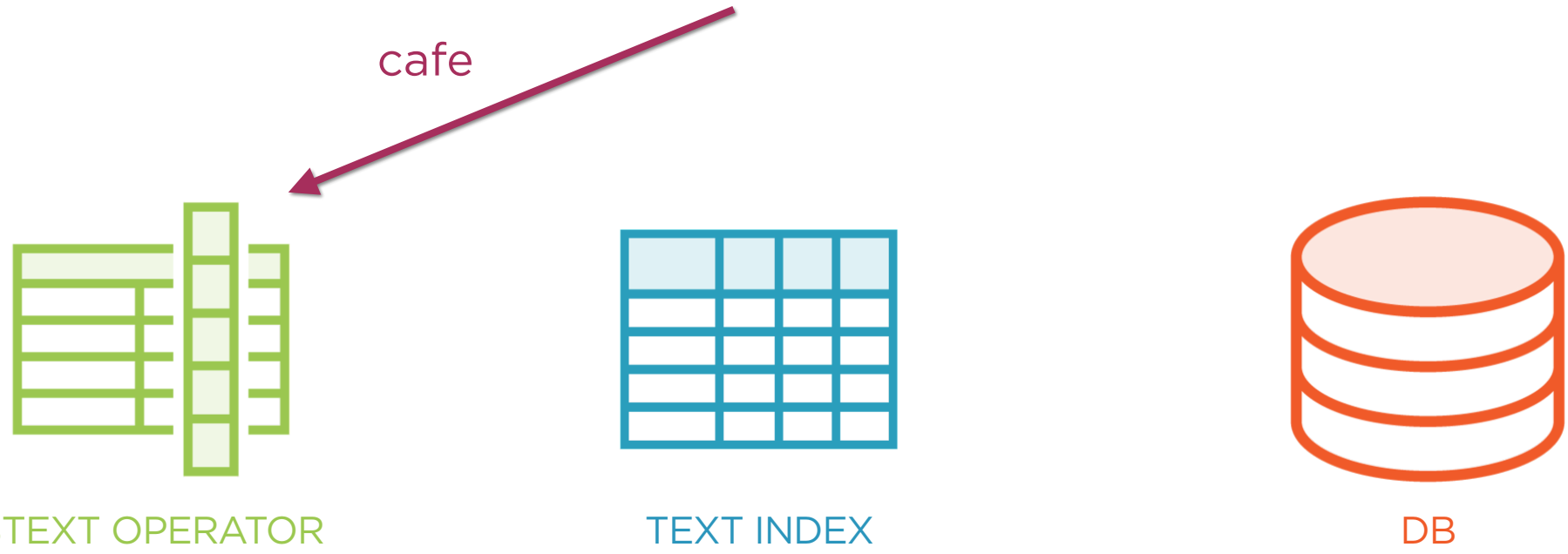
```
>db.drinks.find({$text: {$search: "cafe"}})
{_id:1, beverage: Cafe con leche}
{_id:2, beverage: Café expresso}

>db.drinks.find({$text: {$search: "café"},
$diacriticSensitive: true})
{_id:1, beverage: Café con leche}
```

- # On v3 indexes “cafe” gets both results
- # But with diacriticSensitive:true, it only gets the correct accent

This is central for multi-language!

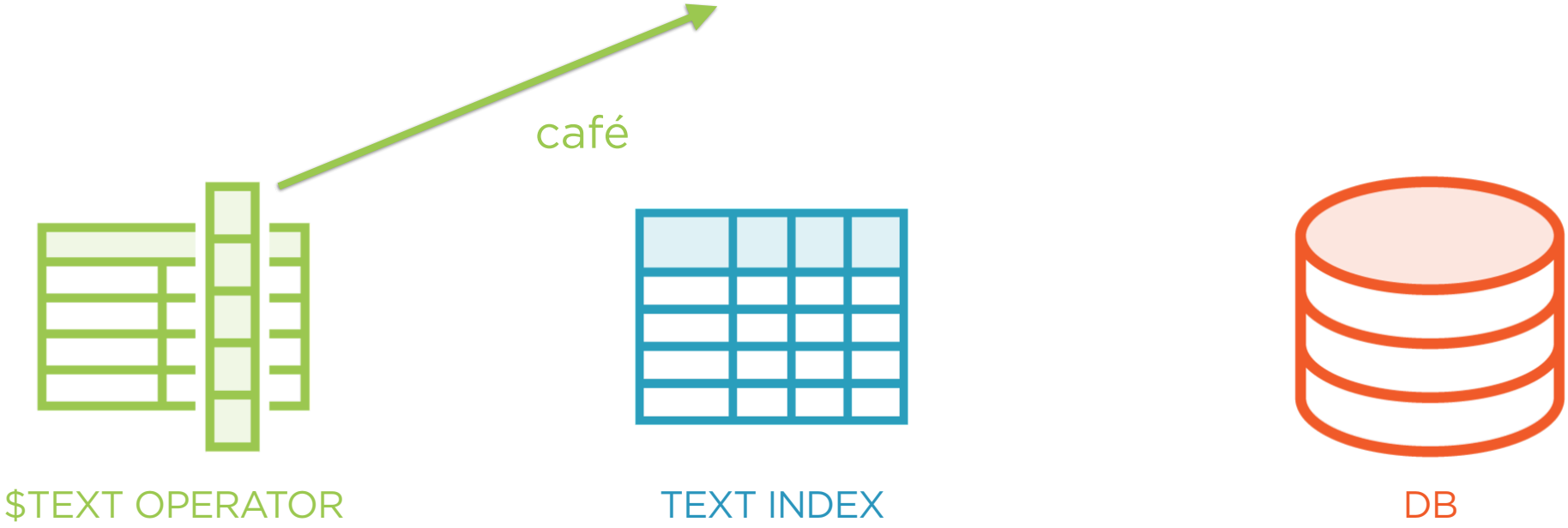
Getting Some Coffee

```
find({$text: {$search: "café"},
$diacriticSensitive: true})
```



Getting Some Coffee

```
find({$text: {$search: "café"},
$diacriticSensitive: true})
```



Getting Some Coffee

```
find({$text: {$search: "café"},
$diacriticSensitive: true})
```

Getting Some Coffee

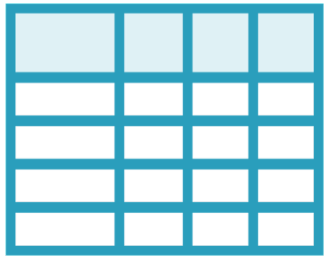
```
find({$text: {$search: "café"},
$diacriticSensitive: true})
```

Getting Some Coffee

```
find({$text: {$search: "café"},
$diacriticSensitive: true})
```

Fetch IDs [1,4]

$TEXT OPERATOR

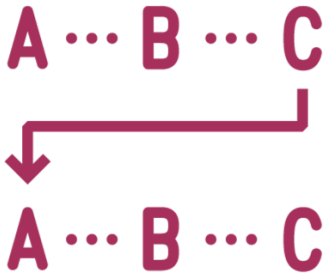


TEXT INDEX

DB

Getting Some Coffee

FILTER CAFE



FINAL RESULTS

This makes this search very inefficient

Getting Some Coffee

As v1/v2 indexes are diacritic sensitive

It would be a **single query**

But it would be **less performant** in size

The Case Sensitiveness Case

V3 Text indexes are **case insensitive by default**

Running a caseSensitive query is actually 2 queries

In older indexes this is 1 query, but overall **less performant**

Specifying Multiple Languages for a Text Index

Remembering Text Indexes

Word	ID
cafe	[1,2]
expresso	2
leche	1
con	1

con is a stopword in spanish!

Non-default Language Text Indexes

```
> db.drinks.createIndex({beverage: "text"}, {default_language: "es"})
```

This brings different **stems and stop words**!

For all supported languages check here

Spanish Text Indexes

Word	ID
cafe	[1,2]
expresso	2
leche	1

We could reduce the index size

Handling Multiple Languages

```
> db.quotes.findOne({}, {_id: 0}).pretty()
{
   "language" : "portuguese",
   "original" : "A sorte protege os audazes.",
   "translation" : [
      {
         "language" : "english",
         "quote" : "Fortune favors the bold."
      },
      {
         "language" : "spanish",
         "quote" : "La suerte protege a los
audaces."
      }
   ]
}
```

- **# Top level language**
- **# Set a different language for this subdocument**

This is central for multi-language!

Rules for Language

Language field of document

Inherit enclosing document's language field

Language from text index

Default language

Multi-Language Indexes

MongoDB offers a language **“none”**

No stemming is done nor stop words are removed

None indexes are huge, but needed in multi-language collections

Handling Multiple Languages

{_id:1, beverage: “Cafe con leche”, **language: “spanish”**, subtype: {type: "Blend de moca y negro"}}
{_id:2, beverage: “Espresso coffee”, **language: “english”**}
{_id:3, beverage: "Blueberry chai tea"}

db.drinks.createIndex({"$**": "text"},
{ default_language: "none" })

- **# Set spanish lanaguage for this document**
- **# But this one gets “none”**
- **Create the index with default_language: “none” since this collection is multilingual**

Assigning a Language

{_id:1, beverage: “Cafe con leche”, **language: “spanish”**, subtype: {type: "Blend de moca y negro"}}
{_id:2, beverage: “Espresso coffee”, **language: “english”**}
{_id:3, beverage: "Blueberry chai tea"}

{_id:1, beverage: "Cafe con leche", **language: "spanish"**, subtype: {type: "Blend de moca y negro", **language:”spanish”**}}. # Inherited language
{_id:2, beverage: "Espresso coffee", **language: "english"**}
{_id:3, beverage: "Blueberry chai tea", **language: “none"**} # Language from index

Tokenization with Language

Tokenization	Language	ID
[cafe, con, leche]	Spanish	1
[Blend, de, mocca, y, negro]	Spanish	1
[Expresso, coffee]	English	2
[Blueberry, chai, tea]	None	3

Stemming with Language

Stemmed	Language	ID
[cafe, leche]	Spanish	1
[Blend, mocca, negro]	Spanish	1
[Expresso, coffee]	English	2
[Blueberry, chai, tea]	None	3

Note that **Blueberry** did not map to berry!

Final Index with Language

Word	ID
cafe	1
leche	1
blend	1
moca	1
negro	1
expresso	2
cofee	2
blueberry	3
chai	3
tea	3

Demo

Analyze multi-language collection queries in depth

Check the difference in query results when the base language is none

Summary

Learned about different text index versions

In V3 indexes case and diacritic searches are actually 2 queries

How to handle properly multi-language collections