Introduction to MongoDB

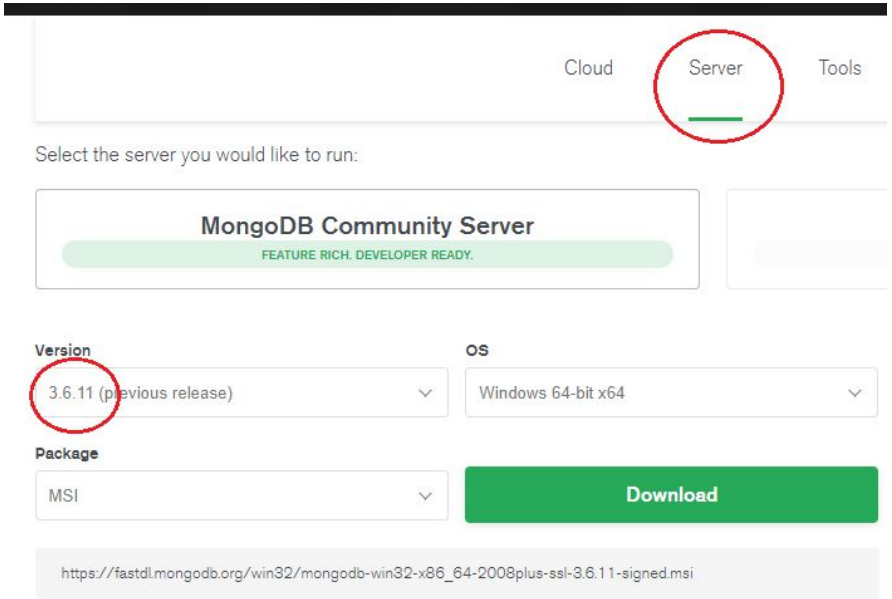
15/04/2019

What is MongoDB

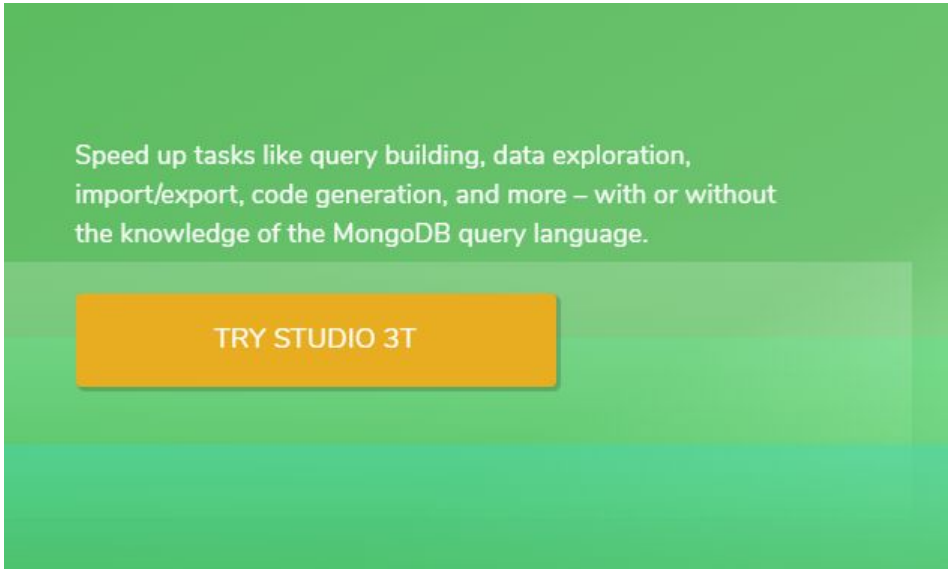
- It’s a free NoSQL document oriented database.
  - Not only SQL, since SQL queries are supported.
- It uses JSON-like documents to save data.
  - Flexible schemas - Allows missing fields.
- High performance, high availability, and automatic scaling.
  - Used by large companies like Amazon.

Installation

MongoDB community server

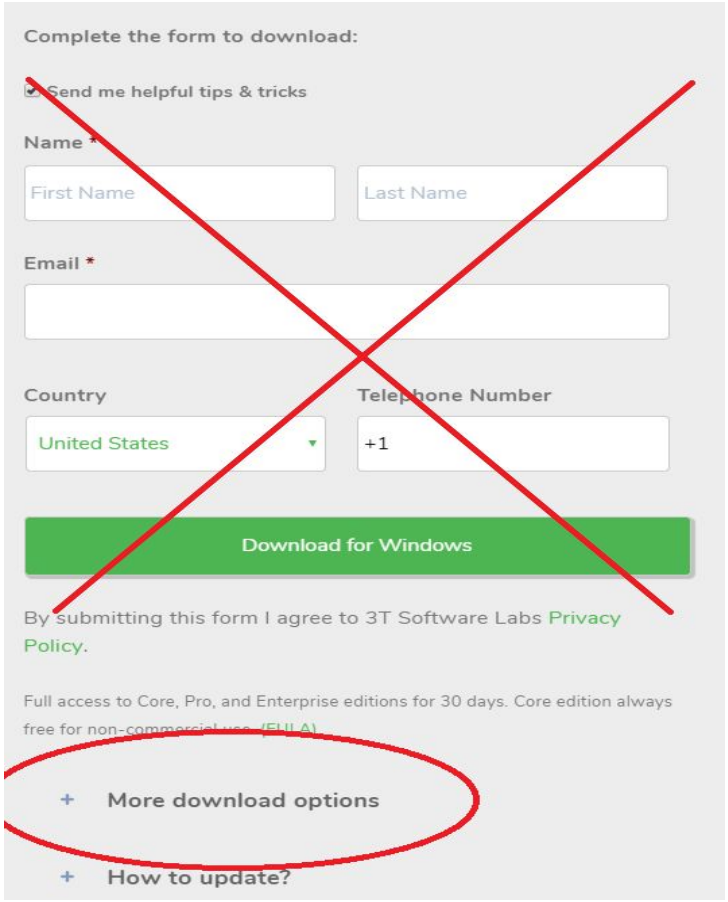


Studio 3T



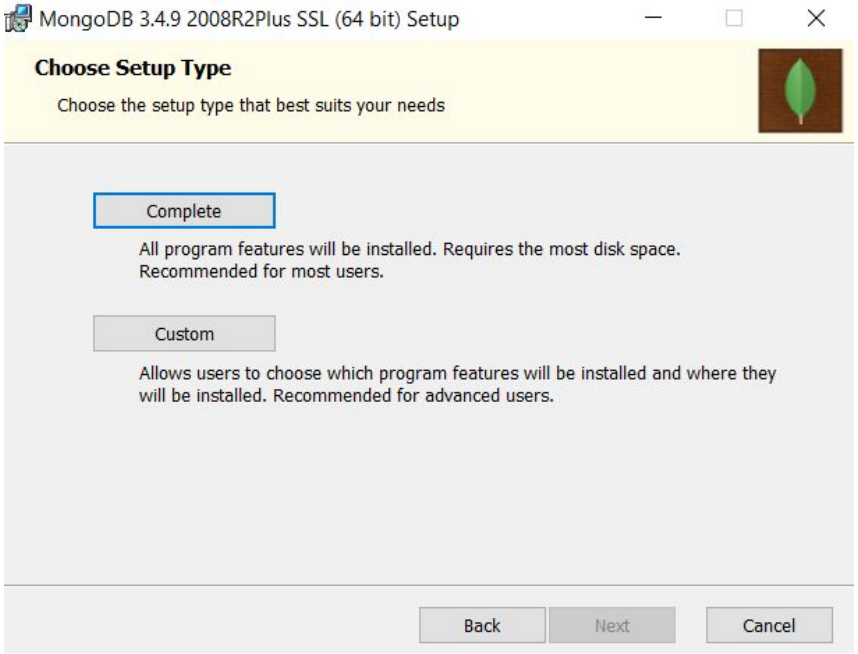
Installation

Studio 3T 2/2

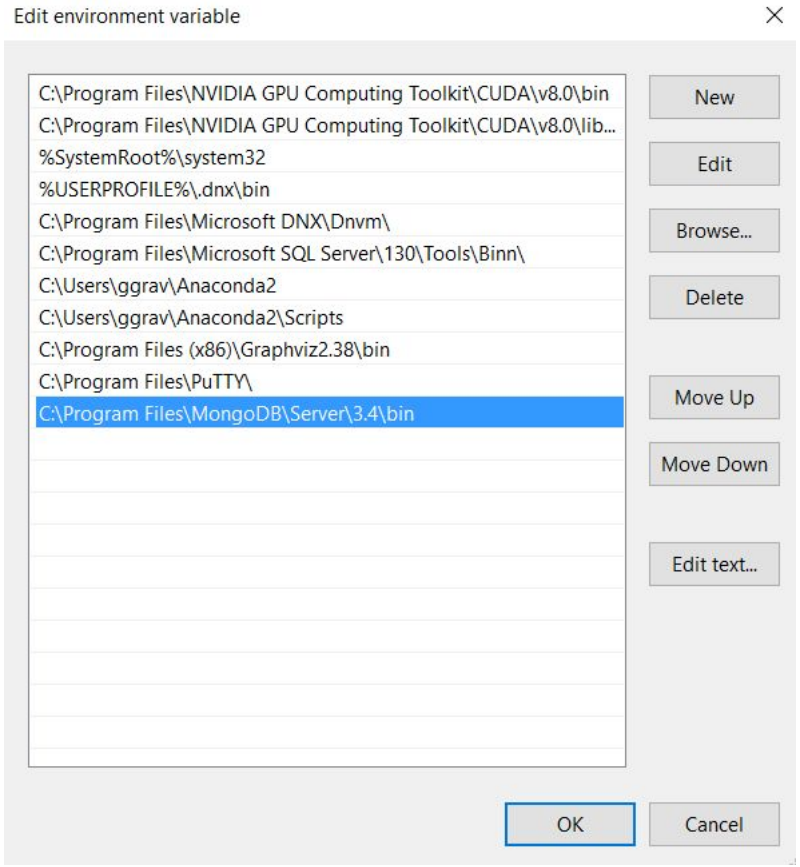


Installation

Step 1

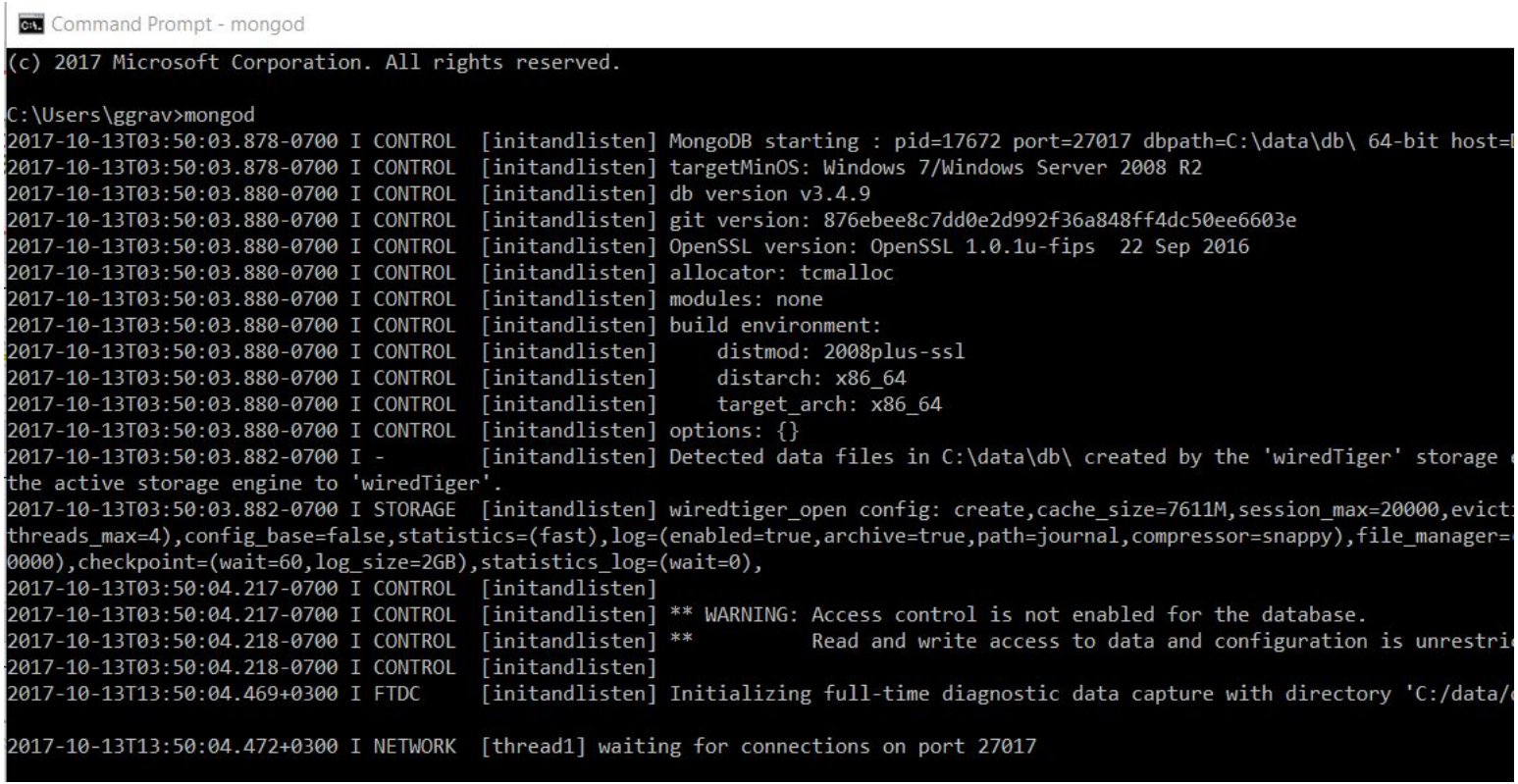


Step 2



Execution

Step 3 -> Create folder “C:\data\db\” -> Open cmd -> run “mongod”



What is a document

- A sum of key / value pairs

What is a collection

- A grouping of MongoDB **documents**.
- A **collection** is the equivalent of an RDBMS table.
- A **collection** exists within a single database.
- **Collections** do not enforce a schema.
- **Documents** within a collection can have different fields.
- **Collections** are grouped into **databases**.

Example: **db** = tweets, **collections** = greek_tweets, english_tweets, etc.

Operations

- Run / Connect to mongo
- Import documents
- Insert documents
- Create queries
- Quick data lookups
- Indexing

https://www.mongodb.com/what-is-mongodb

Connection

Studio 3T:

- Studio 3T -> Connect -> New connection -> (enter name) -> Save -> Connect
- Right click -> Add database -> (enter name) -> ok

Python:

```
1       import pymongo
2
3       client = pymongo.MongoClient('localhost', 27017)
4       db = client['yelp']
```

Import dataset

Studio 3T:

- Connect -> Select DB -> Import collections -> JSON -> Add sources -> Rename target collections -> Rename collections* -> Start import

* Rename collections according to the assignment.

Querying

Studio 3T:

- Right click collection -> Open Intellishell
- Examples:
  - db.restaurants.find({})
  - db.restaurants.find({"neighborhood": "Downtown"})
  - db.restaurants.find({ **$and**: [{"neighborhood": "Southeast"}, {"city": "Las Vegas"}]})

Python:

```
24          db['restaurants'].find({'neighborhood': neighborhood})
25          db['restaurants'].find_one({'business_id': restaurant_id})
26          db['reviews'].find({"$and": [{"business_id": restaurant_id}, {"stars": 5}]})
```

Data lookups - For quick data checking

Studio 3T:

- Right click collection -> Open Intellishell
  - Ex: db.collection_name.find({ "Search_Field": "value" }, { "Field_to_display": 1 })
  - db.restaurants.find({ "neighborhood": "Downtown" }, { "name": 1 })
  - db.restaurants.find({ "name": /.*pollos.*/ }, { "text": 1 })
    - /.* is the regex equivalent for: “any single character, 0 or more times”

Python:

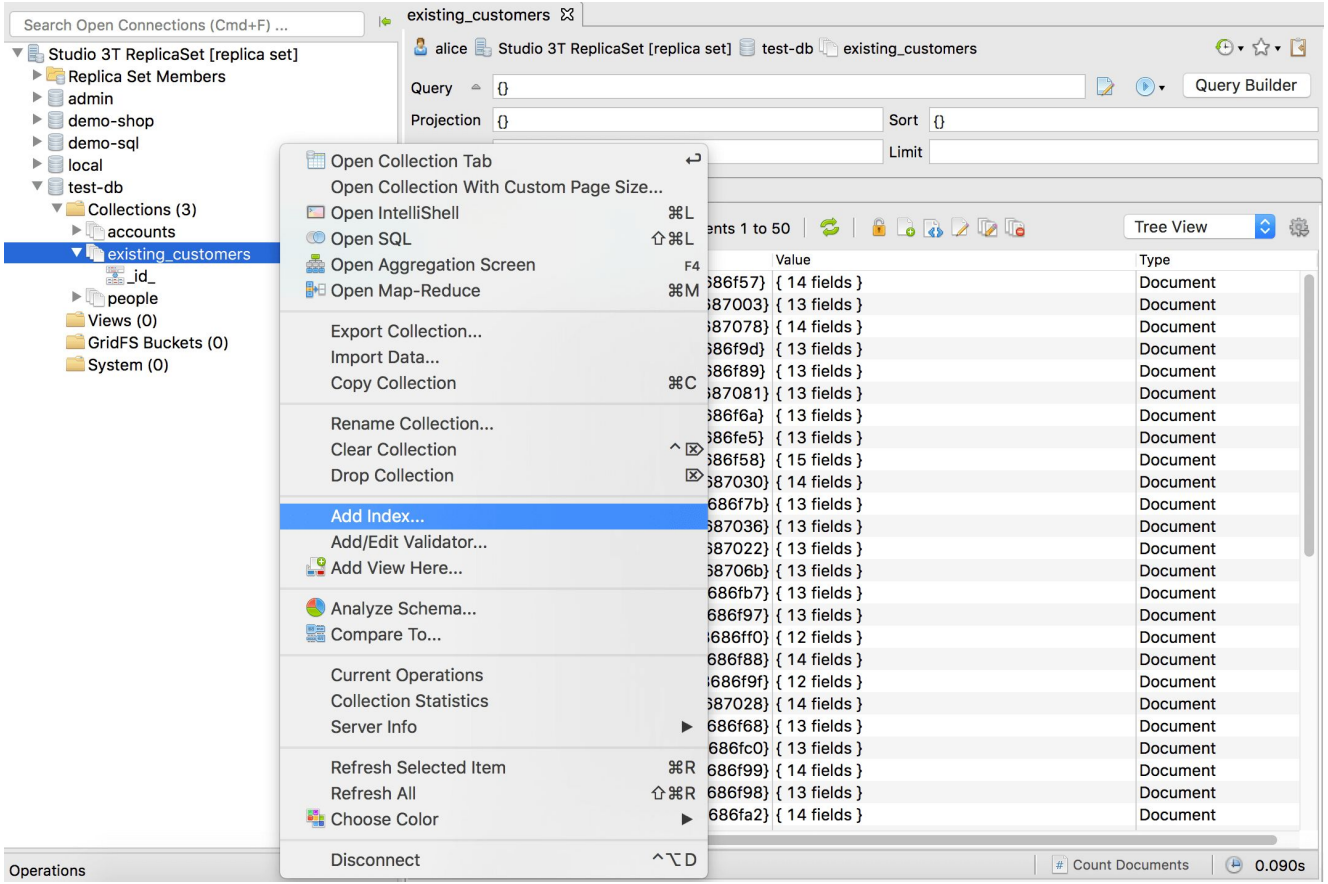
```
27      def find_reviews_that_contain_a_word(word):
28          return db['reviews'].find({'text': {'$regex': '.*' + word + '.*'}})
29
```

Indexing

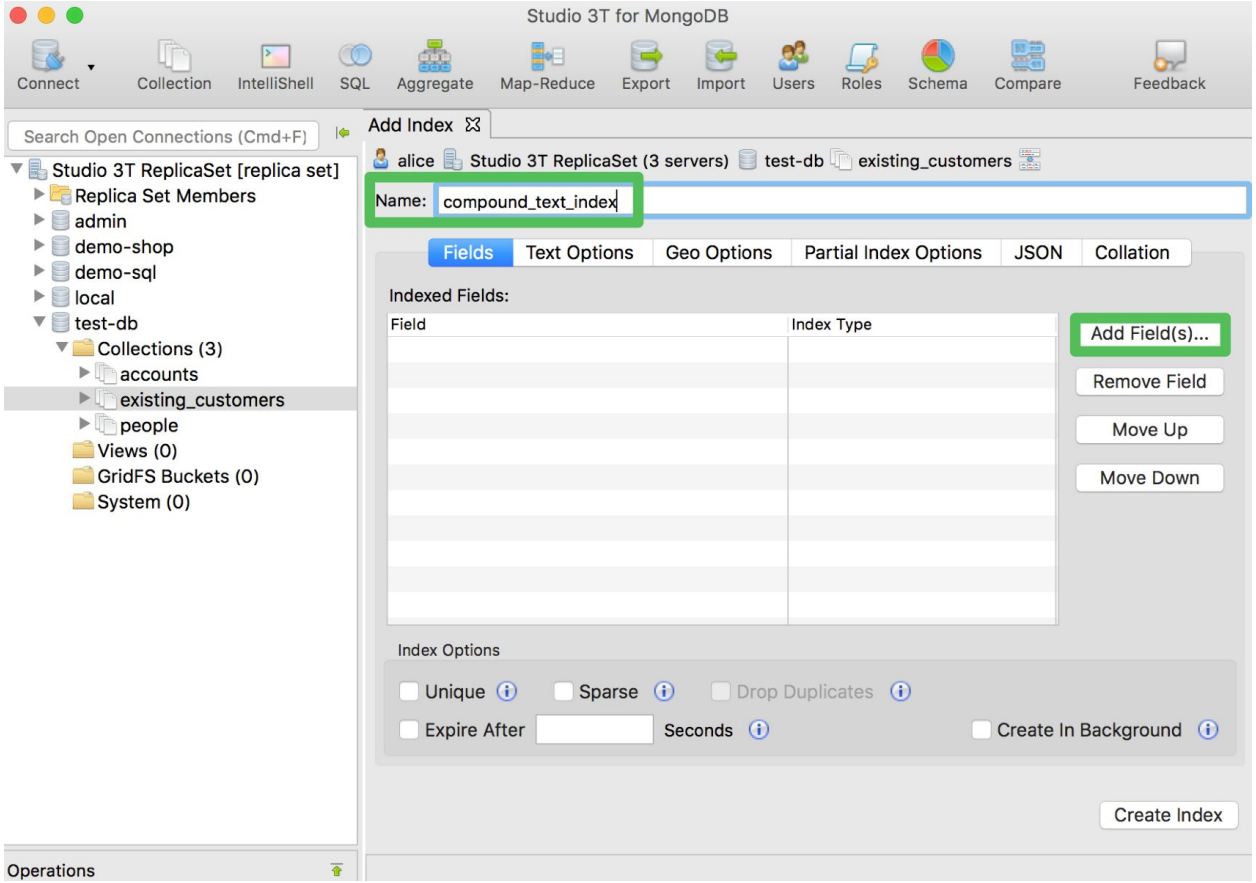
- Basic indexing
  - Speeds up queries on specific fields
  - Mostly used fields should be indexed
  - Index responsibly : Too many indices might slow down the database

- Text indexing
  - Faster text search queries on string content
  - Term lookup
  - Exact phrase lookup
  - Automatic relevance sorting

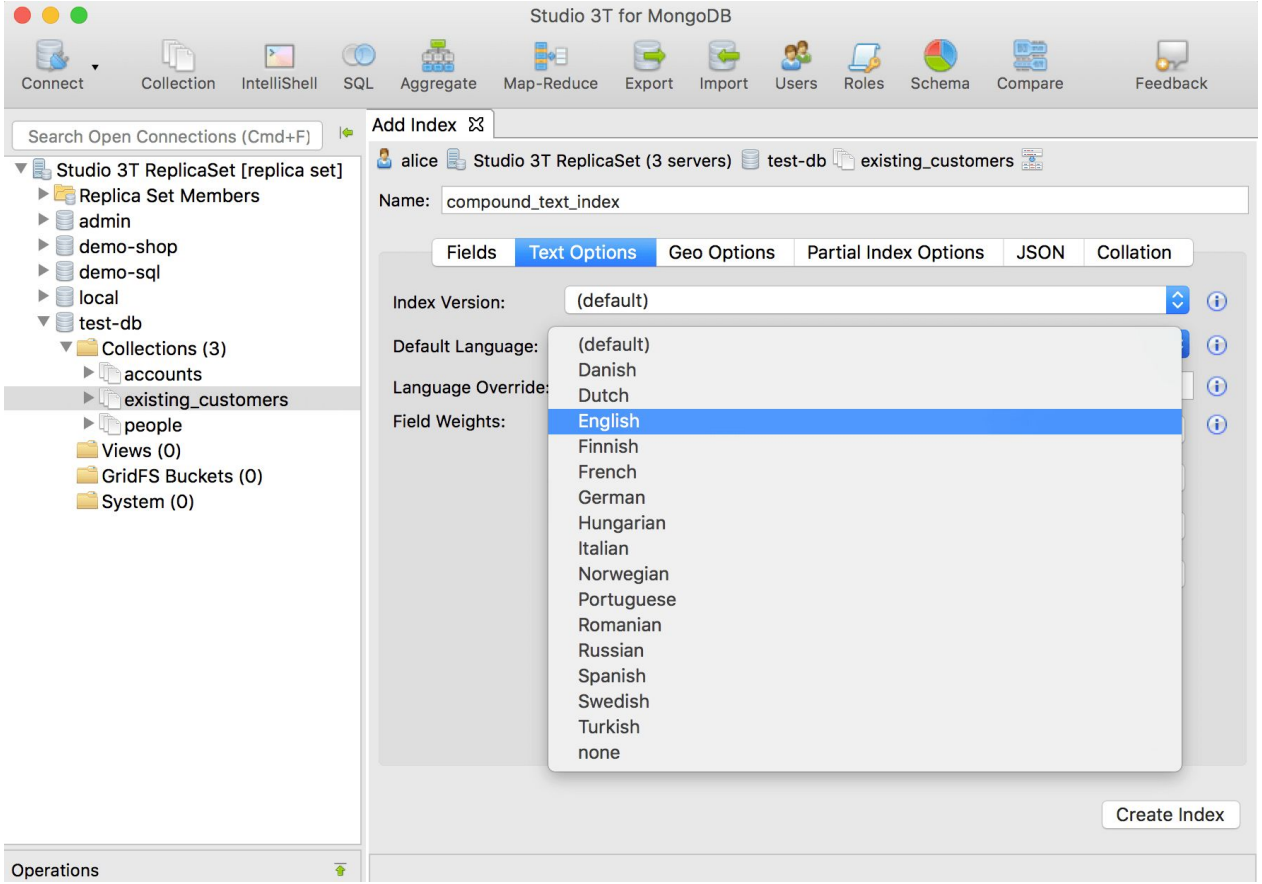
How to - Text indexing



How to - Text indexing



How to - Text indexing



Querying with text indices

- db.stores.find({ $text: { $search: "java coffee shop" } })
  - $text tokenizes the search string and performs a logical OR
  - Will search on all indexed fields
  - Results include a relevance score for each record
- db.stores.find({ $text: { $search: "java \"coffee shop\"" } })
  - Will match exact phrases “java” OR “coffee shop”
- db.stores.find({ $text: { $search: "java shop -coffee" } })
  - Will match (“java” OR “shop”) AND NOT(“coffee”)

Web app development with flask

A python-based web microframework, suitable for small-scale applications.

Flask Documentation

Demo TODO app

Jinja2 Template Engine Documentation

Resources

- MongoDB
- Flask
- Flask-RESTful
- Python

https://github.com/vasisouv/flask-mongo-snippets