Database used: MongoDB

URL: <https://www.mongodb.org/>

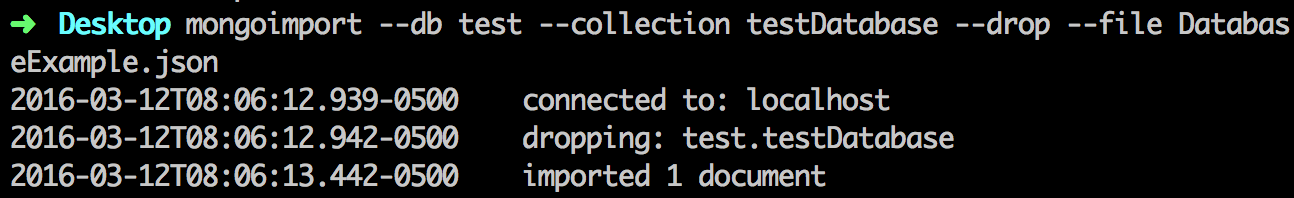
Dataset used:

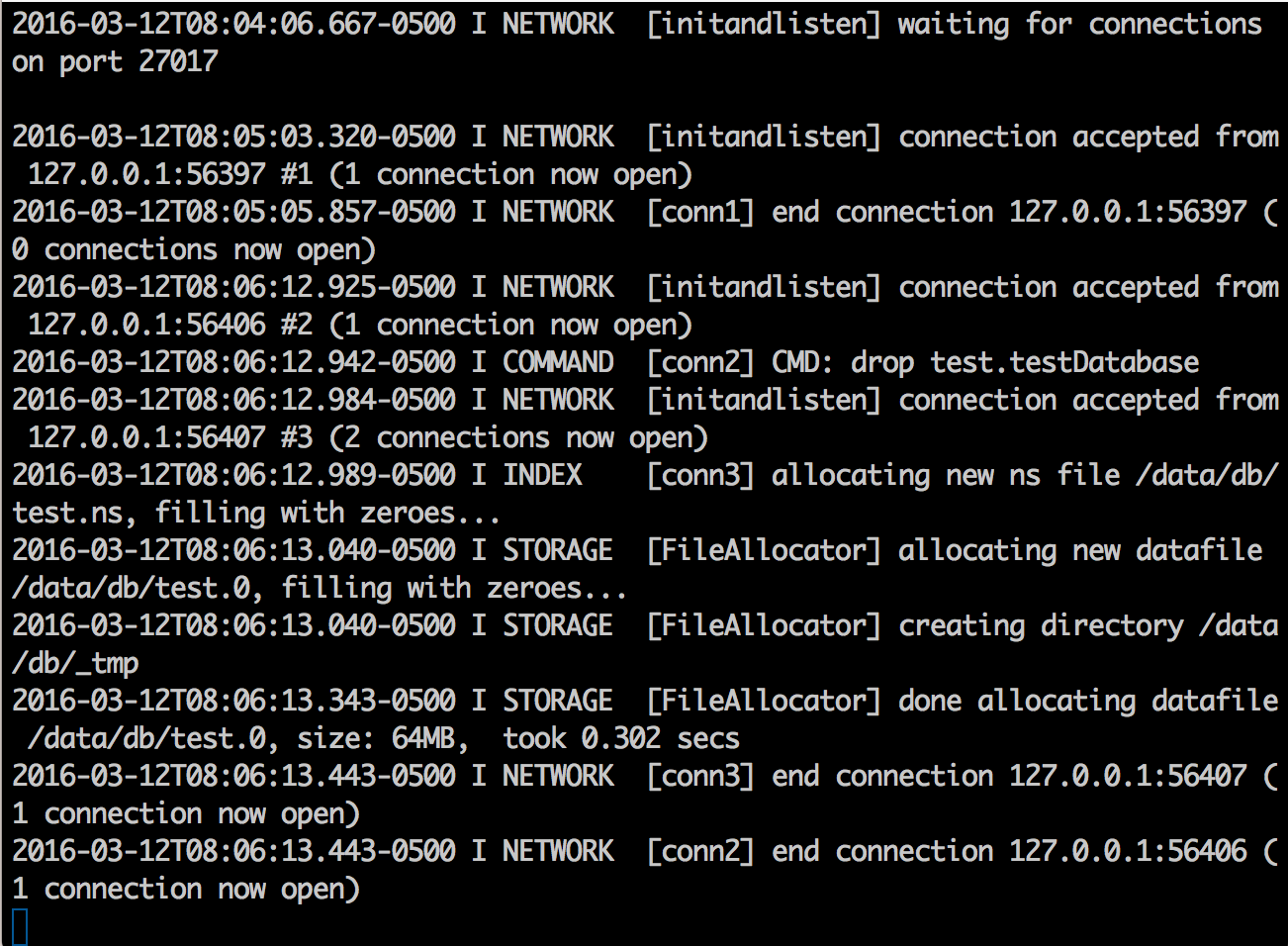
Enron Email Dataset (423MB)

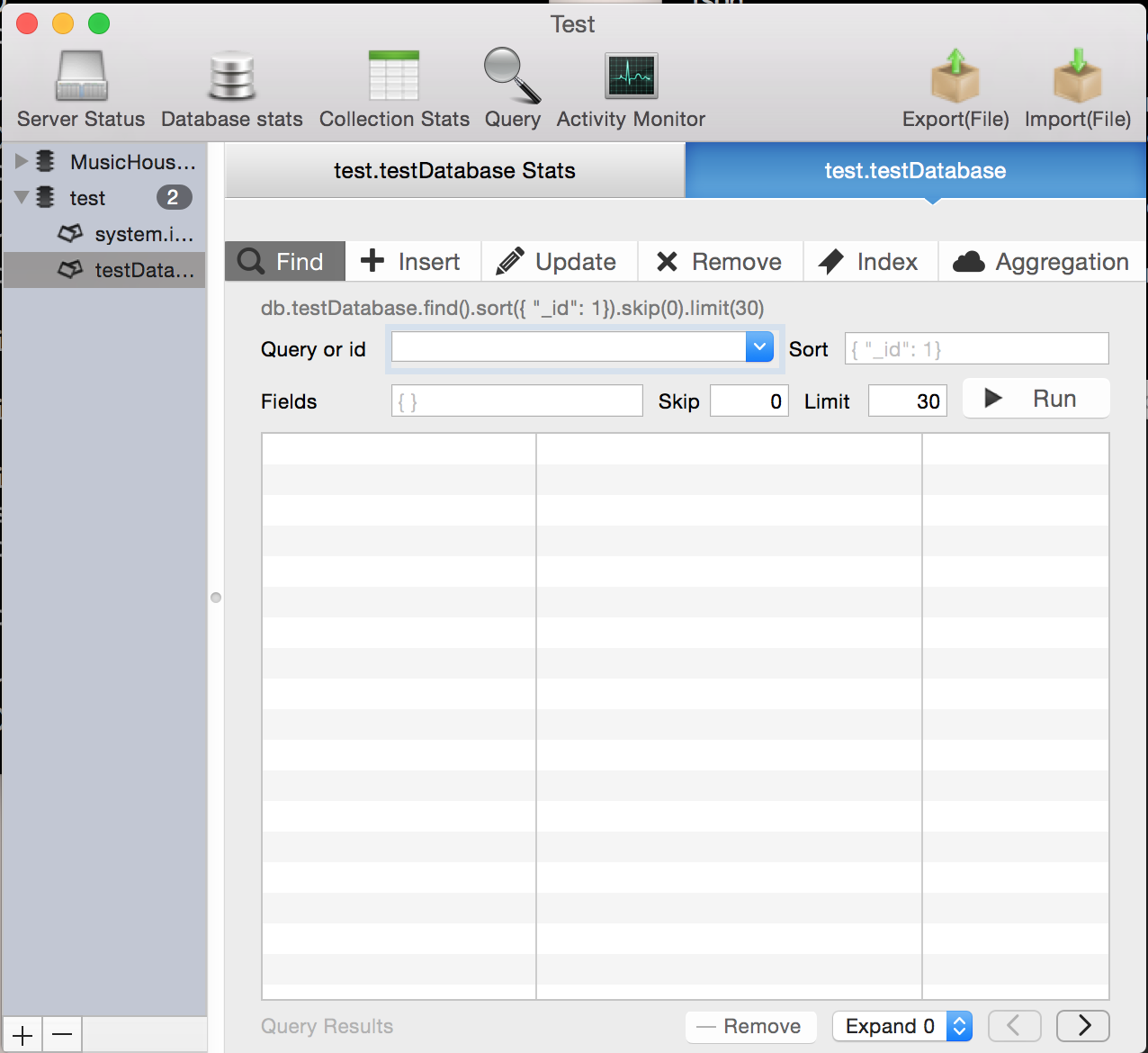
URL: <http://www.cs.cmu.edu/~enron/>

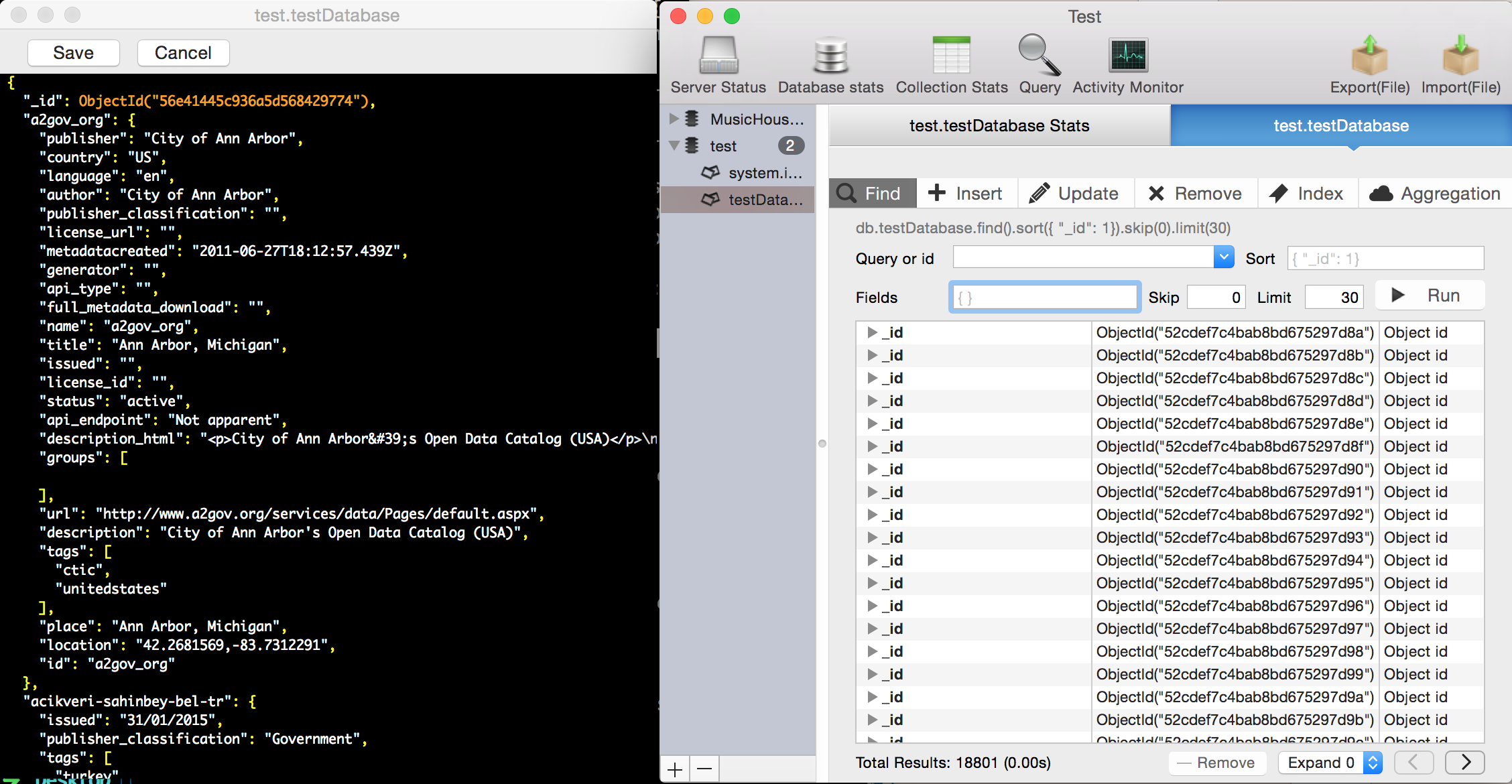
Companies.zip (50MB)

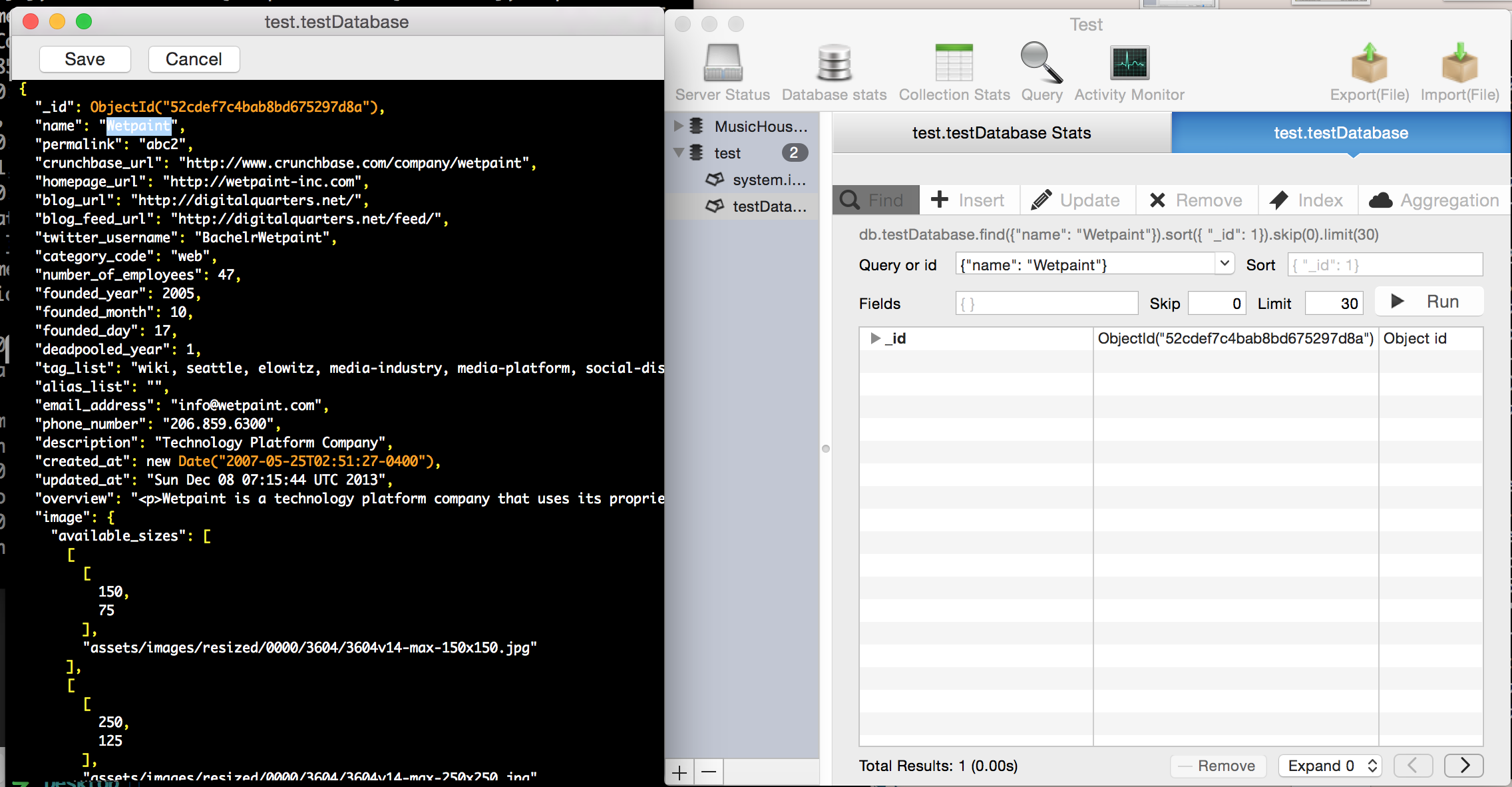
ScreenShot:











Experiment:

Searching:

1.

SQL: SELECT \* FROM ‘ testDatabase ’

MongoDB: db.testDatabase.find()

2.

SQL: SELECT \* FROM ‘testDatabase’ WHERE ‘name’ = ‘Wetpaint’

MongoDB: db. testDatabase.find({name’:" Wetpaint’"})

3.

MongoDB: db. testDatabase.find({"relationships.is\_past": "True"})

4.

MongoDB :db. testDatabase.insert({name:"Company", post\_text:"This is a sample post", post\_privacy:"public", post\_likes\_count:0})

Tabular format runtime:

|  |  |
| --- | --- |
| Query1 | 0.12s |
| Query2 | 0.04s |
| Query3 | 0.04s |
| Query4 | 0.03 |

Analysis:

MongoDB isn't like a traditional relational database. It's noSQL or document based, it provides weak consistency guarantees, and it doesn't have to guarantee consistency like SQL, which makes it really fast for all kind of query.

MongoDB provides a lot of pros, while it still has some cros in practical using, like:

1. No rollbacks. Your code must function without rollbacks. Check all programmatic conditions before performing the first database write operation. Order your write operations such that the most important operation occurs last.

2. Explicit locking. Your code may explicitly lock objects when performing operations.

3. Do not support traditional Database concept like: Transition, which is sometime really useful in Database; Relationship, which is the way we do search in Database like MySql.

4. Data size in MongoDB is typically higher due to e.g. each document has field names stored it.

5. Less flexibity with querying, because there is no such functions like Joins.

As we can see, the speed of MongoDB is really fast for normal query.