Working with Databases in Python 3

Using a Local NoSQL Database - Mongita

Douglas Starnes
Author / Speaker

@poweredbyaltnet | linktr.ee/douglasstarnes

Overview

What is NoSQL?

MongoDB

Mongita

CRUD with Mongita

Refactor the SQLite command line demo to use Mongita

SQL vs. NoSQL

NoSQL does not replace SQL databases

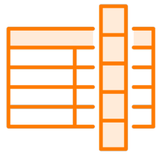
NoSQL compliments SQL databases

Some NoSQL databases include SQL like query languages

Relational and everything else

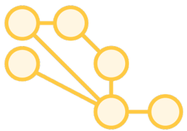
What is ‘Everything else’?

Document (MongoDB)



Column oriented (Cassandra)

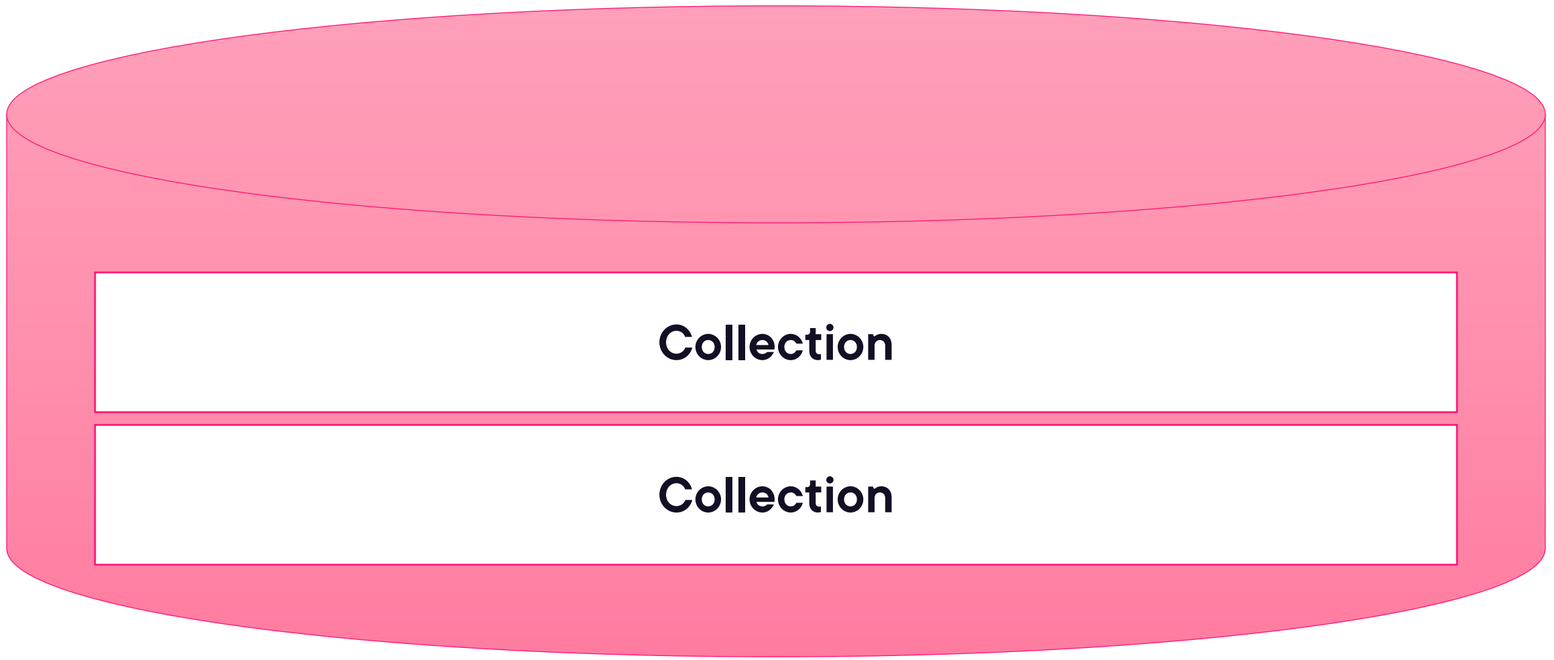
Key-value (Redis)



Graph (Neo4j)

Crash Course in MongoDB

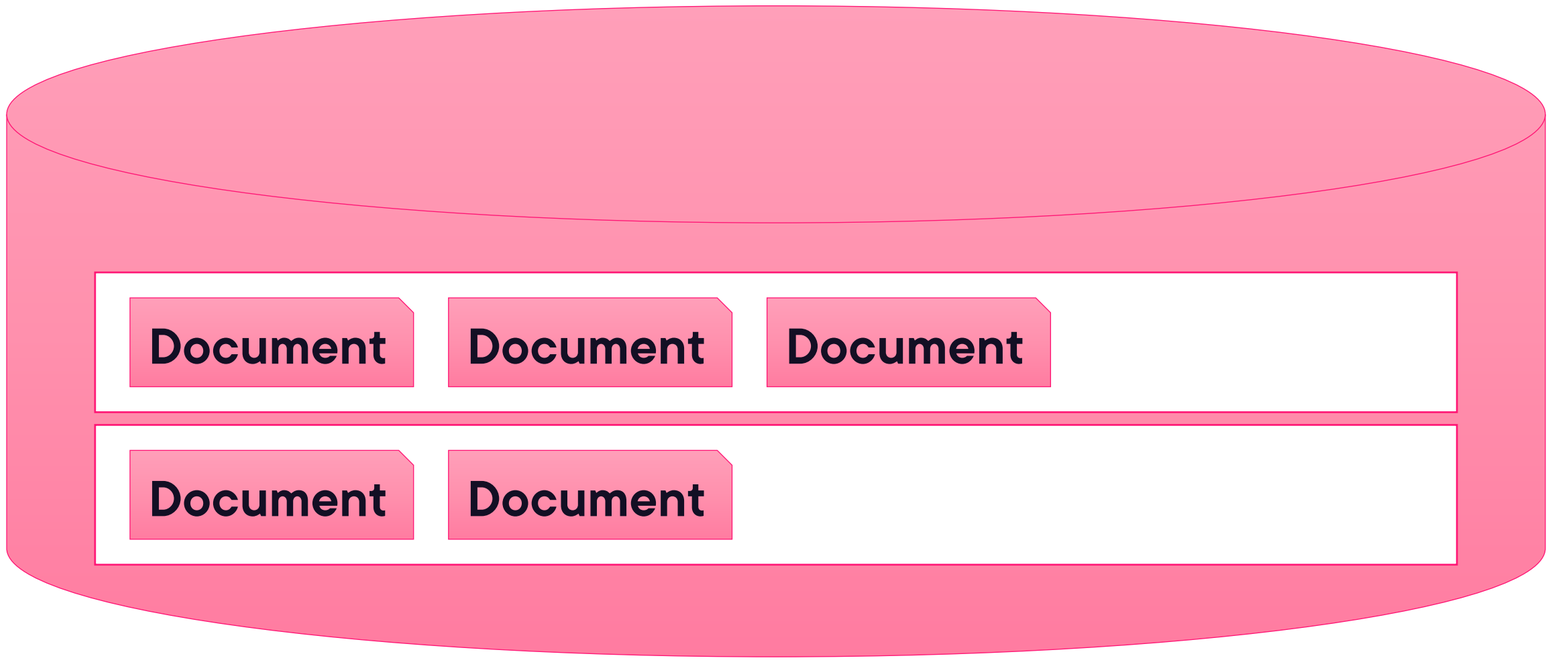
A database is a group of collections



Crash Course in MongoDB

A database is a group of collections

A collection is a group of documents

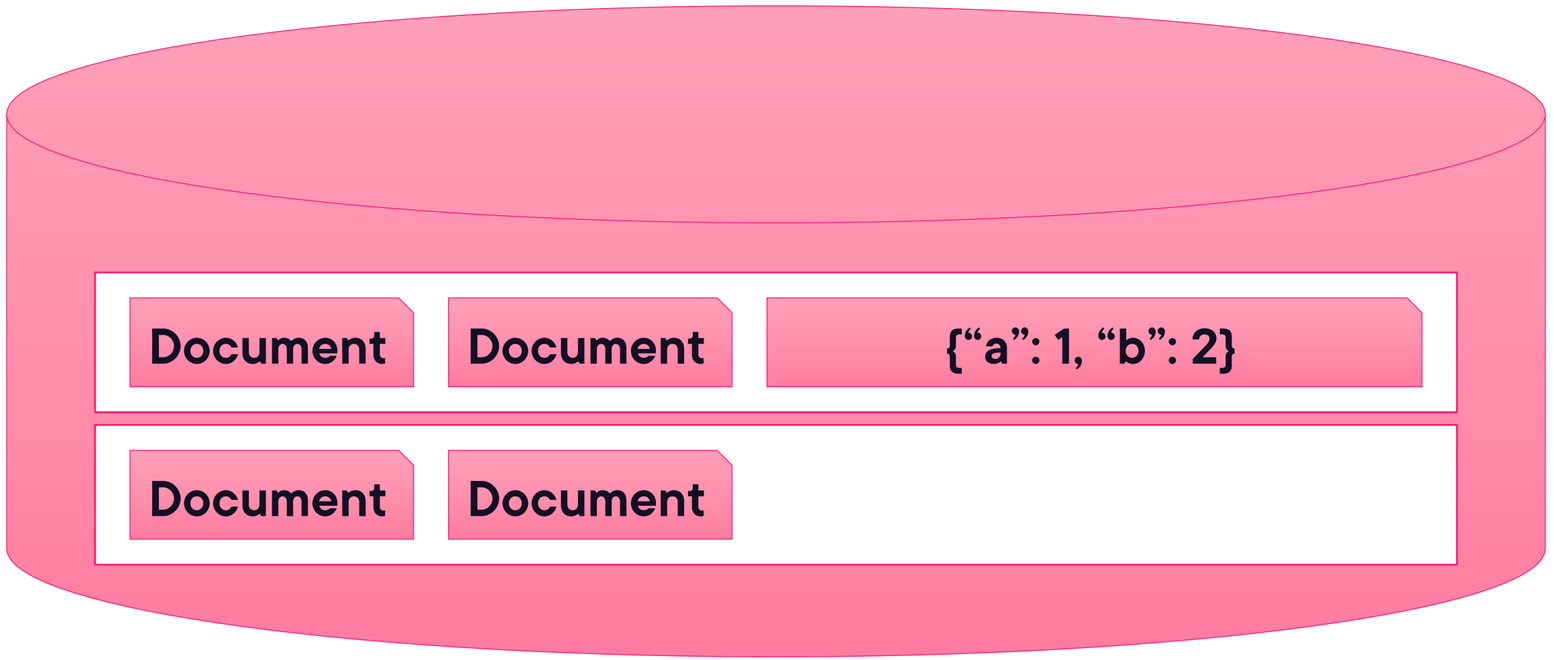


Crash Course in MongoDB

A database is a group of collections

A collection is a group of documents

A document stores data (conceptually) in JSON



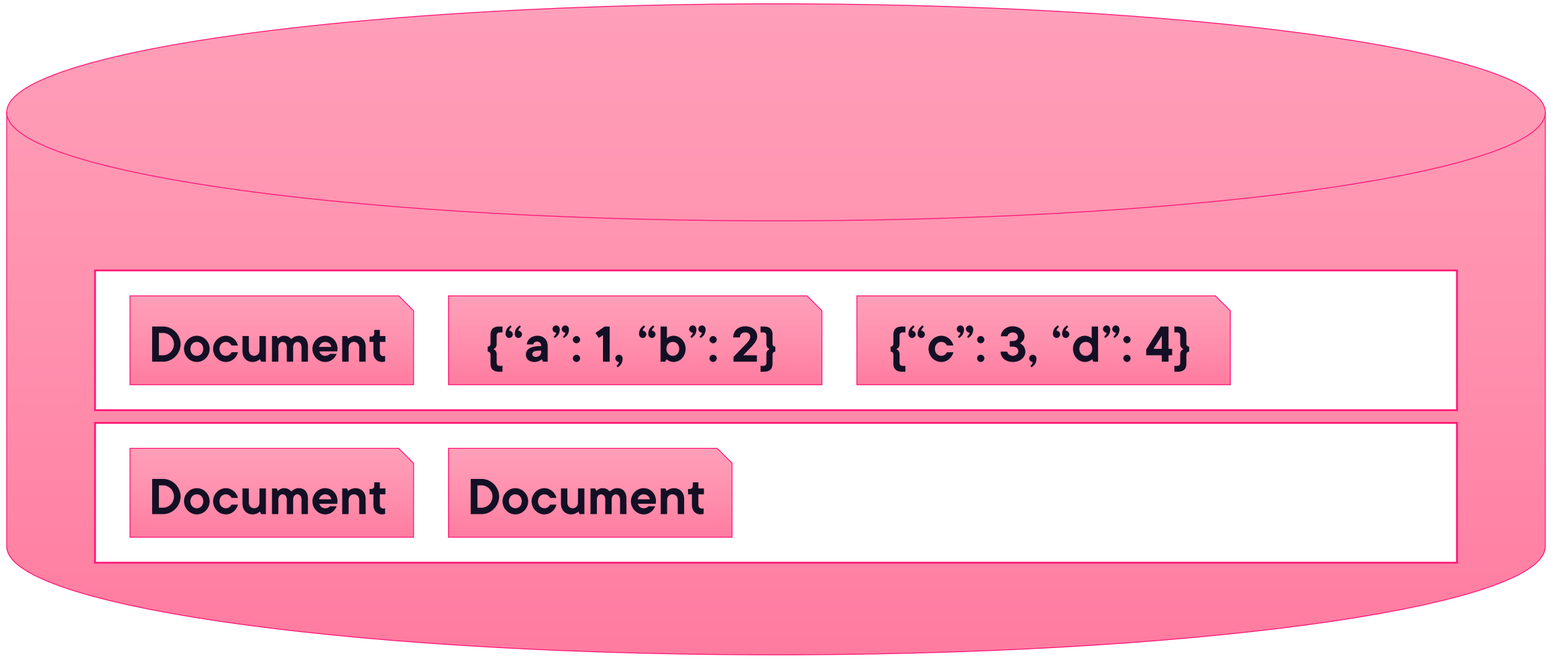
Crash Course in MongoDB

A database is a group of collections

A collection is a group of documents

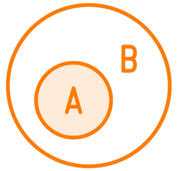
A document stores data (conceptually) in JSON

The documents are schemaless and the structure is not enforced



Mongita

Conceptually similar to SQLite, except for document databases



Python package that implements a subset of the `pymongo` API

Embedded database

Unit testing

Using `mongita`

```
from mongita import MongitaClientDisk
```

Using mongita

```
from mongita import MongitaClientDisk

client = MongitaClientDisk()
```

Using `mongita`

```
from mongita import MongitaClientDisk

client = MongitaClientDisk()

db = client.portfolio # get or create the portfolio database
```

Using `mongita`

```
from mongita import MongitaClientDisk

client = MongitaClientDisk()

db = client.portfolio # get or create the portfolio database

investments = db.investments # get or create the investments collection
```

Using mongita

```
from mongita import MongitaClientDisk

client = MongitaClientDisk()

db = client.portfolio # get or create the portfolio database

investments = db.investments # get or create the investments collection

# insert a document into the investments collection
investments.insert_one({“coin_id”: “bitcoin”, “currency”: “usd”, “amount”: 1.0})
```

Using mongita

```
from mongita import MongitaClientDisk

client = MongitaClientDisk()

db = client.portfolio # get or create the portfolio database

investments = db.investments # get or create the investments collection

# insert a document into the investments collection
investments.insert_one({“coin_id”: “bitcoin”, “currency”: “usd”, “amount”: 1.0})

investments.insert_many([
    {“coin_id”: “ethereum”, “currency”: “usd”, “amount”: 10.0},
    {“coin_id”: “solana”, “currency”: “usd”, “amount”: 25.0}
])
```

Using `mongita`

```
# get all documents with a coin_id of bitcoin
bitcoin_investments = investments.find({“coin_id”: “bitcoin”})

# get first document with a coin_id of bitcoin
first_bitcoin_investment = investments.find_one({“coin_id”: “bitcoin”})

# all documents in the investments collection
all_investments = investments.find({})

# an empty result set will return the None value
no_investments = investments.find({“coin_id”: “dogecoin”})
```

What about multiple filter documents?

Supported by pymongo, but not supported by Mongita

Using `mongita`

```
[
    {“coin_id”: “bitcoin”, “amount”: 1.0},
    {“coin_id”: “bitcoin”, “amount”: 0.5},
    {“coin_id”: “ethereum”, “amount”: 10.0}
]
```

Using mongita

```
[
    {“coin_id”: “bitcoin”, “amount”: 1.0},
    {“coin_id”: “bitcoin”, “amount”: 0.5},
    {“coin_id”: “ethereum”, “amount”: 10.0}
]

# increment all bitcoin investments by 0.1
updates = investments.update_many(
    {“coin_id”: “bitcoin”},
    {“$inc”: {“amount”: 0.1}}
)
```

Using mongita

```
[
    {“coin_id”: “bitcoin”, “amount”: 1.0},
    {“coin_id”: “bitcoin”, “amount”: 0.5},
    {“coin_id”: “ethereum”, “amount”: 10.0}
]

# increment all bitcoin investments by 0.1
updates = investments.update_many(
    {“coin_id”: “bitcoin”},
    {“$inc”: {“amount”: 0.1}}
)

# delete the first (and only) ethereum investment
deletes = investments.delete_one({“coin_id”: “ethereum"})
```

Summary

NoSQL is a compliment to SQL databases

There are many different NoSQL databases

Mongita

- Stores data in documents which are conceptually JSON objects
- Official Python API in the pymongo package
- Mongita implements a subset of pymongo
  - Embedded database

The `insert_one` and `find` methods

Filter and update documents