Introduction to Database Systems: CS305 PyMongo

Oliver Bonham-Carter
Hang Zhao

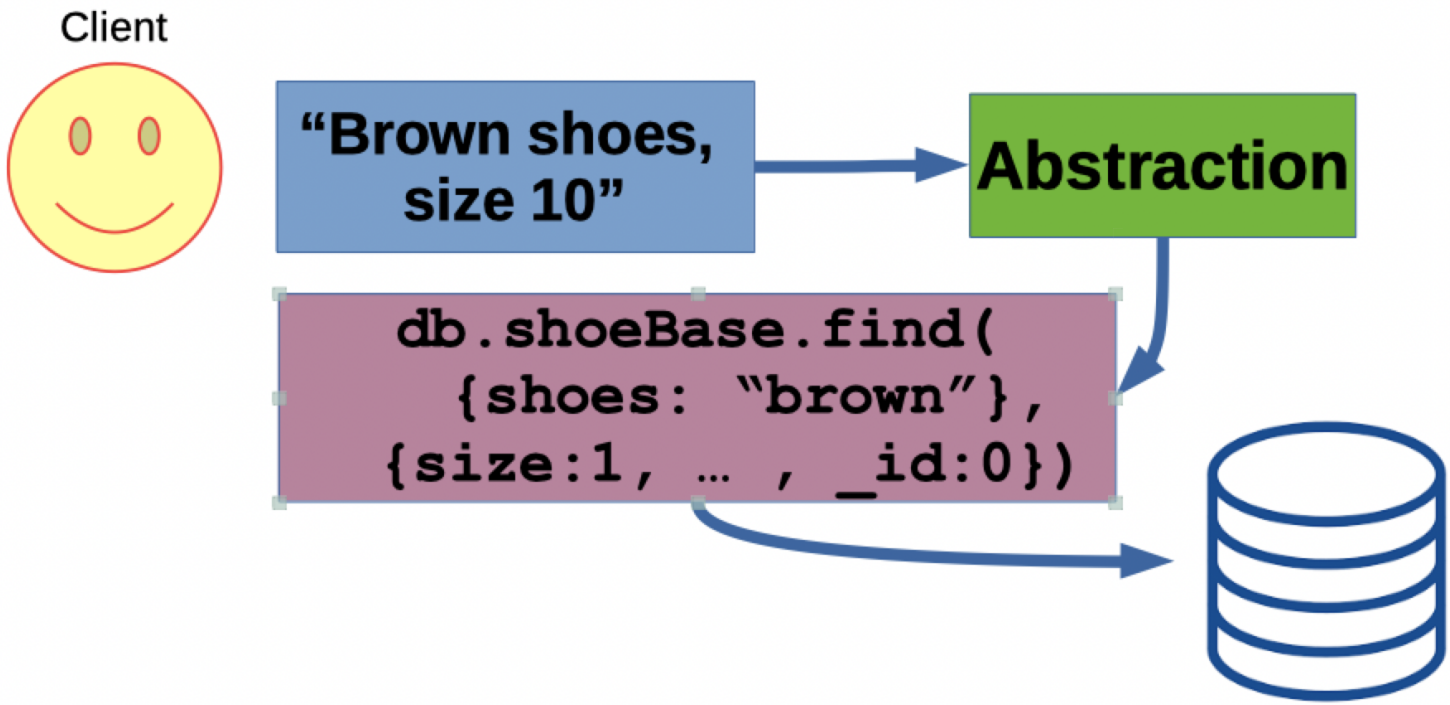
21 November 2023

And now, back to Docker with MongoDB

Abstraction

Abstraction

To make some process *abstract* is to hide (automate) some of the details that serve to complicate the process. The idea behind *abstraction* here is to create a more user-friendly experience by removing some of the the complexities of using Mongo databases.



Mongo Container Commands

Start the bash (make sure the Docker MongoDB container is running!)

```
sudo docker exec -it mongodb bash
```

Below are commands to enter when inside the container

Download updated package information with apt

```
apt-get update
```

Install an editor, Python3, Pip

```
apt-get install nano
apt-get install vim
apt-get install python3-pip
apt-get install python3-venv
```

Use Pip to install *pymongo*

```
pip install pymongo
```

Tools

Create file in; `/mongodata` and `locate` in container; `/data/db`

You could use *Nano* to begin coding (or VSCode – just be sure you are in the correct directory when you work!!)

```
nano pymongoDemo.py
```

After coding, exit Nano and run your code

```
python3 pymongoDemo.py
```

Main Nano Menu Items

```
^G Get Help    ^O WriteOut    ^R Read File
^X Exit        ^J Justify     ^W Where is

^Y Prev Pg     ^K Cut Text    ^C Cur Pos
^V Next Pg     ^U UnCut Text  ^T To Spell
```

- Control-O :: ^O : Save
- Control-X :: ^X : Exit

Boilerplate code

Create file in; /mongodata and locate in container; /data/db

```
#!/usr/bin/env python3

# libraries
from pymongo import MongoClient
import string

# creating connections for communicating with MongoDB
client = MongoClient('localhost:27017')
db = client.mongodemo # The name of the collection is mongodemo

# Define functions here!

# User Interaction

print("\t [+] Data BEFORE addition")
read() # call read function

print("\t [+] Insert some data")
insert() # call insert function()

print("\t [+] Data AFTER addition")
read() # call read function to view the changes

print("\t [+] Update Data")
update() # call update to ask for new information to replace existing

print("\t [+] Data AFTER Update")
read() # call read function to view the changes
```

Read Function

```
def read():
   """ function to read records from mongo db """
   try:
      empCol = db.Employee.find()
      print("\n Found: all data from DataEmployee \n")

      for emp in empCol:
         print(f"\t [+] {emp}")

   except Exception as e:
      print(str(e))
# end of read()
```

Insert

```
def insert():
   """ Function to insert data into mongo db """
   employeeId = input('Enter Employee id :')
   employeeFirstName = input('Enter FirstName :')
   employeeLastName = input('Enter LastName :')
   employeeAge = input('Enter age :')
   employeeCountry = input('Enter Country :')

   # insert the data into the base
   try:
      db.Employee.insert_one(
      {
      "id": employeeId,
      "firstName":employeeFirstName,
      "lastName":employeeLastName,
      "age":employeeAge,
      "country":employeeCountry
      })
      print("\nInserted data successfully\n")

   except Exception as e:
      print(str(e))
# end of insert()
```

Update

```
def update():
   """ Function to update record to mongo db """
   print("  Update:")
   try:
      employeeId = input('  Enter Employee id :')
      employeeFirstName = input('  Enter FirstName :')
      employeeLastName = input('  Enter LastName :')
      employeeAge = input('  Enter age :')
      employeeCountry = input('  Enter Country :')

      # update the record with the new information
      db.Employee.update_one(
      {"id": employeeId},
      {
      "$set": {
      "firstName":employeeFirstName,
      "lastName":employeeLastName,
      "age":employeeAge,
      "country":employeeCountry
      }})

      print("\nRecords updated successfully. \n")

   except Exception as e:
      print(str(e))
   # end of update()
```

How do we find this database using Mongosh?

Useful commands for inside the container

Load the MongoDB Shell

```
mongosh
```

How do we find this database using Mongosh?

Useful commands for inside the container

From Inside the MongoDB Shell: List collections

```
show collections
```

```
mongodemo> show collections
Employee
```

List the databases

```
show dbs
```

```
mongodemo> show dbs
admin      40.00 KiB
config     96.00 KiB
local      72.00 KiB
mongodemo  72.00 KiB
people     44.00 KiB
test       84.00 KiB
```

Engaging the database made from PyMongo

```
test> show dbs
admin       40.00 KiB
config      96.00 KiB
local       72.00 KiB
mongodemo   72.00 KiB
people      44.00 KiB
test        84.00 KiB
test> use mongodemo
switched to db mongodemo
```

```
mongodemo> db.Employee.find({},{})
[
  {
    _id: ObjectId("655a8bffe91ed8a46e0ec3cd"),
    id: '1007',
    firstName: 'James-ish',
    lastName: 'Bondy',
    age: '25',
    country: 'UK'
  },
  {
    _id: ObjectId("655a8d35c2be1fc6eca177c9"),
    id: '9999',
    firstName: 'Jane',
    lastName: 'Blond',
    age: '25',
    country: 'UK'
  },
```

Choose *mongodemo* (where PyMongo placed all data)

```
use mongodmo
```

Do a *catch-all* query

```
db.Employee.find({},{})
```

Consider this ...

- Can you make a database in MongoDB?
- Can you create code to manage the data you use to populate this database?
- Can you think of applications for your code to other areas?