MongoDB

and an introduction to unstructured (NoSQL) databases

Frank Burkholder
Michael Dyer

galvanize

Objectives

1. Understand what unstructured data is in terms of a database
2. Explain how MongoDB keeps track of unstructured data
3. Be able to do some simple CRUD commands in the Mongo Shell
4. Use PyMongo in the assignment to hold web-scraped data

Unstructured data

Unstructured data refers to information that doesn't reside in a traditional row-column database.

- Each row of the database can have its own fields, and from row to row those fields can be different.
- There is no pre-defined schema (columns or fields have a certain type).
- Each object can be completely different from all others and can change basically arbitrarily over time.
- Each document can have or not have whatever fields are appropriate for that particular document
- Opposite of a structured database (SQL is an example structured)
- Web-scraping data is often unstructured

MongoDB

- It’s a document-oriented (each row is a document) No-SQL database
- No schema, No joins, No transactions
- JSON-like objects form the data model
    - Each document is made up of key-value pairs (analogous to columns, but defined within document)
- CURSOR: When you ask MongoDB for data, it returns a pointer to the result set called a cursor.
    - Actual execution is delayed until necessary
    - Like a generator, the cursor points to current values but after those values are retrieved it’s exhausted.

SQL (structured) vs. Mongo organization

SQL

customers

cust_id	cust_name	cust_state
1	Kayla	CO
2	Erich	CO

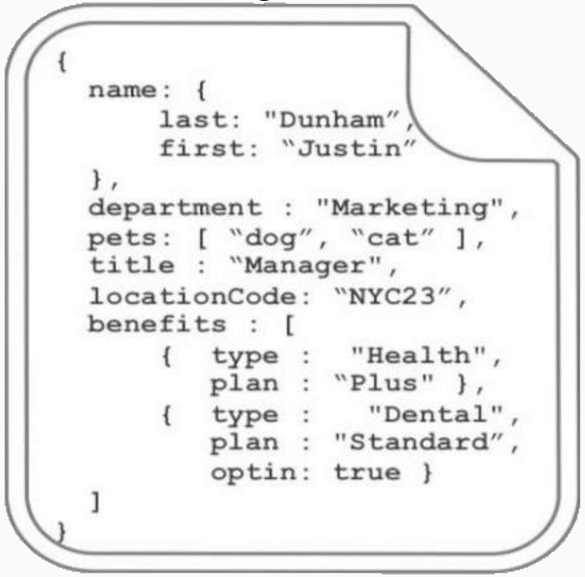
products

prod_id	description	price
1	skis	$300
2	goggles	$75

purchases

cust_id	prod_id	date
1	1	10/30
1	2	11/14

MongoDB



```
{
  name: {
      last: "Dunham”,
      first: “Justin”
  },
  department : "Marketing",
  pets: [ “dog”, “cat” ],
  title : “Manager",
  locationCode: “NYC23”,
  benefits : [
      {  type :  "Health",
         plan : “Plus" },
      {  type :   "Dental",
         plan : "Standard”,
         optin: true }
  ]
}
```

CRUD a simple database in the Mongo Shell

For any database, you need to be able to **C**reate, **R**ead, **U**pdate, and **D**elete documents. You're going to want to find documents, too.

Let's do this in Mongo

Start your Mongo docker container, and access the Mongo Shell.

If you don't remember how, go to the `docker_mongodb.md` guide in the `docker` repo.

CRUD a simple database in the Mongo Shell

```
// Create a new database called class_db
use class_db

// View all the existing collections (tables)
db.getCollectionNames()

// Create a collection (table) and insert records into them
db.teachers.insert({name: 'E-Rich', age: 25, facial_hair: 'clean'})
db.teachers.insert({name: 'Frank', age: 21, friends: ['Adam', 'Cully']})
db.teachers.insert({name: 'Neil', age: 55, friends: ['Barack Obama', 'Kanye']})

// We can then view the first document in our table like so...
db.teachers.findOne()
db.teachers.findOne().pretty()

// Or we can view all our documents at once...
db.teachers.find().pretty()

// Or maybe we want to return the first 2 documents
db.teachers.find().limit(2).pretty()
```

CRUD a simple database in the Mongo Shell

```
// Generally speaking we run a .find({}, {}) where the first set of brackets
// dictates which documents get returned (like a WHERE clause) and the second
// set of brackets dictates which fields should be returned
db.teachers.find({name: 'Neil'}).pretty()
db.teachers.find({name: 'Neil'}, {friends: true}).pretty()
// By default, the id is always returned but we can turn that off
db.teachers.find({name: 'Neil'}, {friends: 1, _id: false}).pretty()

// Return only the friends entry from each document
db.teachers.find({}, {friends: true})
```

Mongo Shell Commands

galvanize

Command	About
help	Show help.
show dbs	Show databases
use <db>	Switch to database, create if non-existent
show collections	Show collections in database
db.collection.insertOne()	Inserts a document into a collection.
db.collection.insertMany()	Inserts multiple documents into a collection.
db.collection.find()	Selects documents in a collection based on the filter and returns a cursor to the selected documents.
db.collection.updateOne()	Updates a single document within the collection based on the filter.
db.collection.updateMany()	Updates all documents within the collection that match the filter.
db.collection.replaceOne()	Replaces a single document within the collection based on the filter.
db.collection.deleteOne()	Removes a single document from a collection based on the filter.
db.collection.deleteMany()	Removes all documents that match the filter from a collection.

Mongo shell command reference: https://docs.mongodb.com/manual/reference/mongo-shell/

PyMongo - interfacing with Mongo using Python

```
$ conda install pymongo
$ docker start mongoserver
```

in your Python script:

```python
from pymongo import MongoClient
client = MongoClient('localhost', 27017)
db = client['database_name']
table = db['collection_name']
```

Syntax is similar, but not exactly the same, as the Mongo shell. Examples:

- W3Schools: https://www.w3schools.com/python/python_mongodb_getstarted.asp
- MongoDB/Python docs: https://api.mongodb.com/python/current/tutorial.html
- **Zetcode:** http://zetcode.com/python/pymongo/
- See the `docker_mongo.db` guide in the docker repo.

Objectives

1. Understand what unstructured data is in terms of a database
2. Explain how MongoDB keeps track of unstructured data
3. Be able to do some simple CRUD commands in the Mongo Shell
4. Use PyMongo in the assignment to hold web-scraped data