

MongoDB

Copyright © 2022 by
Robert M. Dondero, Ph.D
Princeton University

Objectives

- The lecture will:
 - Provide *a taste of* the **MongoDB** DBMS
 - Give you enough information about **MongoDB** to:
 - Help you decide if you want to use it in your project
 - Help you get started with it

MongoDB

- ***MongoDB***
 - **Who:** 10gen, later MongoDB Inc.
 - **When:** 2007
 - **Why:**
 - Document-oriented database
 - Originally: part of a cloud-as-service platform
 - Later: open source flagship product

Motivation

- Why study MongoDB?
 - Popular
 - The most popular **document store** DBMS
 - And more generally...
 - The most popular non-relational (**NoSQL**) DBMS
 - Different
 - Contrasts with relational DBMSs
 - Reasonable for COS 333 project
 - Especially if storing data that is hierarchical to arbitrary depths

Installing MongoDB

- Linux (Debian, Ubuntu)
 - At a bash shell prompt:
 - `sudo apt-get mongodb`

Installing MongoDB

- Mac
 - Install Homebrew
 - Browse to <https://brew.sh/> and follow instructions
 - At a bash shell prompt:
 - `brew tap mongodb/brew`
 - `brew install mongodb-community@4.2`

Installing MongoDB

- MS Windows
 - Browse to <http://mongodb.com/try/download/community>
 - Click on Download
 - In Windows Explorer, double click on the .msi file that was downloaded to launch the installer
 - Use the default installation settings

Installing MongoDB

- MS Windows (cont.)
 - Create a data directory
 - `mkdir C:\data\db`
 - Add `C:\Program Files\MongoDB\server\4.4\bin` to your PATH environment variable

Starting the MongoDB Server

- Linux (without transaction support)
 - Starts automatically at login

Starting the MongoDB Server

- Mac (without transaction support)
 - At a bash shell prompt
 - `mongod -config /usr/local/etc/mongod.conf`

Starting the MongoDB Server

- Mac (with transaction support)
 - At a bash shell prompt
 - `mongod -config /usr/local/etc/mongod.conf -replSet r1`
 - At another bash shell prompt:
 - `mongo`
 - In the resulting MongoDB shell
 - `rs.initiate()`

Starting the MongoDB Server

- MS Windows (without transaction support)
 - At a command prompt:
 - `mongod`

Starting the MongoDB Server

- MS Windows (with transaction support)
 - At a command prompt:
 - `mongod -replSet r1`
 - At another command prompt:
 - `mongo`
 - In the resulting MongoDB shell
 - `rs.initiate()`

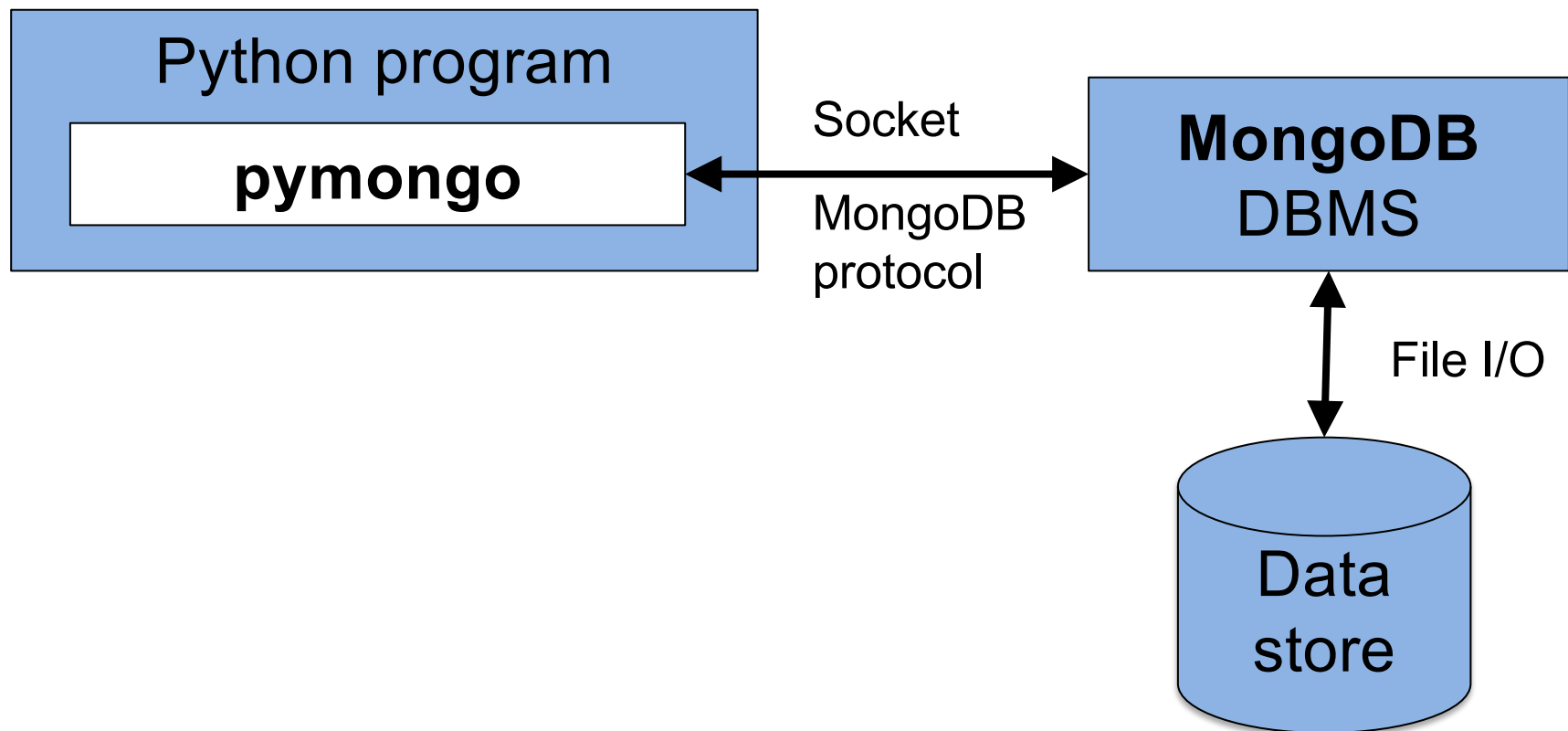
Starting the MongoDB Server

- DBMS listens for connections at a default port 27017
- MongoDB supports authentication, but not by default

The MongoDB Shell

- Linux, Mac, and MS Windows:
 - At bash shell / command prompt:
 - `mongo`
 - (Enter JavaScript statements)
 - `exit`

MongoDB via Python



Installing the MongoDB Driver

- Linux, Mac, and MS Windows
 - At a bash shell prompt:
 - Activate your cos333 virtual environment
 - `python -m pip install pymongo`

The MongoDB Data Model

- The MongoDB data model

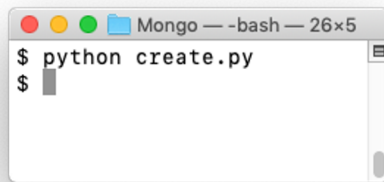
Relational Model	MongoDB
Database	Database
Table	Collection
Row	Document

MongoDB Pgmming in Python

- ***Document***
 - **Internally:**
 - A fragment of ***JavaScript Object Notation (JSON)***
 - (JavaScript and JSON covered later in course)
 - **Externally in Python:**
 - A primitive object (`int`, `float`, `bool`, `str`)
 - Or a `dict` object containing `dict` or `list` or primitive objects,
 - Or a `list` object containing `dict` or `list` or primitive objects
 - With nesting to any level

MongoDB Pgmming in Python

- See **Mongo/create.py**



- DBMS runs as a distinct process on a specified computer (localhost)
- DBMS listens for connections at a known port (27017)

MongoDB Pgmming in Python

- See **Mongo/create.py** (cont.)
 - Create client
 - Index into client to get database

MongoDB Pgmming in Python

- See **Mongo/create.py** (cont.)
 - Index into pennyDB to get bookCollection
 - Corresponds to combination of books and authors tables

MongoDB Pgmming in Python

- See **Mongo/create.py** (cont.)
 - zipcodeCollection
 - Corresponds to zipcodes table
 - `insert_one()` returns a result
 - Result contains an `inserted_id` field
 - `inserted_id` field is document reference

MongoDB Pgmming in Python

- See **Mongo/create.py** (cont.)
 - customerCollection
 - Corresponds to customers table
 - Contains references to documents in zipcodeCollection

MongoDB Pgmming in Python

- See **Mongo/create.py** (cont.)
 - orderCollection
 - Corresponds to orders table

MongoDB Pgmming in Python

- Data relationships: books and authors
 - Relational DB
 - Book data is stored in books table
 - Author data is stored in authors table
 - Each rows in books table is related to row(s) in authors table by isbn
 - MongoDB
 - Book data is stored in bookCollection
 - Author data is merged into bookCollection
 - Each book document contains a list/array of authors
 - Relationship is implemented via **denormalization**

MongoDB Pgmming in Python

- Data relationships: customers and zipcodes
 - Relational DB
 - Customer data is stored in customers table
 - Zipcode data is stored in zipcodes table
 - Each row in customers table is related to a row in zipcodes table by zipcode
 - MongoDB
 - Customer data is stored in customerCollection
 - Zipcode data is stored in zipcodeCollection
 - Each customer document is related to a zipcode document by document reference
 - Relationship is implemented via **references**

MongoDB Pgmming in Python

- Data relationships: books, customers, orders
 - Relational DB
 - Book data is stored in books table
 - Customer data is stored in customers table
 - Order data is stored in orders table
 - Each row in books is related to row(s) in orders by isbn
 - Each row in customers is related to row(s) in orders by isbn
 - MongoDB
 - Book data is stored in bookCollection
 - Customer data is stored in customerCollection
 - Order data is stored in orderCollection
 - Each document in bookCollection is related to document(s) in orderCollection by isbn
 - Each document in customerCollection is related to document(s) in orderCollection by isbn
 - Relationship is implemented via **data**

MongoDB Pgmming in Python

- Data relationships in MongoDB:
 - **Option 1:** denormalization
 - **Option 2:** references
 - **Option 3:** data

MongoDB Pgmming in Python

- See **Mongo/display.py** (cont.)

```
Mongo — -bash — 102x32
$ python display.py

books
-----
{'_id': ObjectId('5f3349a6d8f0622de1fa8dee'), 'isbn': '123', 'title': 'The Practice of Programming', 'authors': ['Kernighan', 'Pike'], 'quantity': 500}
{'_id': ObjectId('5f3349a6d8f0622de1fa8def'), 'isbn': '234', 'title': 'The C Programming Language', 'authors': ['Kernighan', 'Ritchie'], 'quantity': 800}
{'_id': ObjectId('5f3349a6d8f0622de1fa8df0'), 'isbn': '345', 'title': 'Algorithms in C', 'authors': ['Sedgewick'], 'quantity': 650}
-----

zipcodes
-----
{'_id': ObjectId('5f3349a6d8f0622de1fa8df1'), 'zipcode': '08540', 'city': 'Princeton', 'state': 'NJ'}
{'_id': ObjectId('5f3349a6d8f0622de1fa8df2'), 'zipcode': '02138', 'city': 'Cambridge', 'state': 'MA'}
{'_id': ObjectId('5f3349a6d8f0622de1fa8df3'), 'zipcode': '02142', 'city': 'Cambridge', 'state': 'MA'}
-----

customers
-----
{'_id': ObjectId('5f3349a6d8f0622de1fa8df4'), 'custid': '111', 'custname': 'Princeton', 'address': {'street': '114 Nassau St', 'zipcode': ObjectId('5f3349a6d8f0622de1fa8df1')}}
{'_id': ObjectId('5f3349a6d8f0622de1fa8df5'), 'custid': '222', 'custname': 'Harvard', 'address': {'street': '1256 Mass Ave', 'zipcode': ObjectId('5f3349a6d8f0622de1fa8df2')}}
{'_id': ObjectId('5f3349a6d8f0622de1fa8df6'), 'custid': '333', 'custname': 'MIT', 'address': {'street': '292 Main St', 'zipcode': ObjectId('5f3349a6d8f0622de1fa8df3')}}
-----

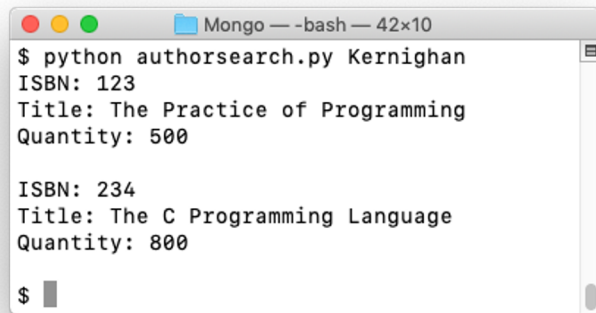
orders
-----
{'_id': ObjectId('5f3349a6d8f0622de1fa8df7'), 'isbn': '123', 'custid': '222', 'quantity': 20}
{'_id': ObjectId('5f3349a6d8f0622de1fa8df8'), 'isbn': '345', 'custid': '222', 'quantity': 100}
{'_id': ObjectId('5f3349a6d8f0622de1fa8df9'), 'isbn': '123', 'custid': '111', 'quantity': 30}
$
```

MongoDB Pgmming in Python

- See **Mongo/display.py** (cont.)
 - `find()` returns an iterable object
 - Each iteration yields a document
 - Each document is a Python dict
 - Each document has a unique `_id`
 - Each customer document references a zipcode document

MongoDB Pgmming in Python

- See **Mongo/authorsearch.py**

A terminal window titled "Mongo - bash - 42x10" showing the execution of a Python script. The prompt is "\$". The command entered is "python authorsearch.py Kernighan". The output shows two book records. The first record has ISBN: 123, Title: The Practice of Programming, and Quantity: 500. The second record has ISBN: 234, Title: The C Programming Language, and Quantity: 800. The prompt "\$" is visible at the bottom left of the terminal window.

```
$ python authorsearch.py Kernighan
ISBN: 123
Title: The Practice of Programming
Quantity: 500

ISBN: 234
Title: The C Programming Language
Quantity: 800

$
```

- `collection.find()` method
- MongoDB supports search through documents

MongoDB Pgmming in Python

- See [Mongo/order.py](#)

```
Mongo — -bash — 102x34
$ python create.py
$ python order.py 123 222
$ python display.py

-----
books
-----
{'_id': ObjectId('5f334c210253ed5a20036352'), 'isbn': '123', 'title': 'The Practice of Programming', 'authors': ['Kernighan', 'Pike'], 'quantity': 500}
{'_id': ObjectId('5f334c210253ed5a20036353'), 'isbn': '234', 'title': 'The C Programming Language', 'authors': ['Kernighan', 'Ritchie'], 'quantity': 800}
{'_id': ObjectId('5f334c210253ed5a20036354'), 'isbn': '345', 'title': 'Algorithms in C', 'authors': ['Sedgewick'], 'quantity': 650}
-----
zipcodes
-----
{'_id': ObjectId('5f334c210253ed5a20036355'), 'zipcode': '08540', 'city': 'Princeton', 'state': 'NJ'}
{'_id': ObjectId('5f334c210253ed5a20036356'), 'zipcode': '02138', 'city': 'Cambridge', 'state': 'MA'}
{'_id': ObjectId('5f334c210253ed5a20036357'), 'zipcode': '02142', 'city': 'Cambridge', 'state': 'MA'}
-----
customers
-----
{'_id': ObjectId('5f334c210253ed5a20036358'), 'custid': '111', 'custname': 'Princeton', 'address': {'street': '114 Nassau St', 'zipcode': ObjectId('5f334c210253ed5a20036355')}}
{'_id': ObjectId('5f334c210253ed5a20036359'), 'custid': '222', 'custname': 'Harvard', 'address': {'street': '1256 Mass Ave', 'zipcode': ObjectId('5f334c210253ed5a20036356')}}
{'_id': ObjectId('5f334c210253ed5a2003635a'), 'custid': '333', 'custname': 'MIT', 'address': {'street': '292 Main St', 'zipcode': ObjectId('5f334c210253ed5a20036357')}}
-----
orders
-----
{'_id': ObjectId('5f334c210253ed5a2003635b'), 'isbn': '123', 'custid': '222', 'quantity': 21}
{'_id': ObjectId('5f334c210253ed5a2003635c'), 'isbn': '345', 'custid': '222', 'quantity': 100}
{'_id': ObjectId('5f334c210253ed5a2003635d'), 'isbn': '123', 'custid': '111', 'quantity': 30}
$
```

MongoDB Pgmming in Python

- See **Mongo/order.py** (cont.)
 - `collection.update_one()` method
 - MongoDB supports complex queries and various kinds of updates

MongoDB Pgmming in Python

- See [Mongo/purchase.py](#)

```
Mongo — -bash — 104x35
$ python create.py
$ python purchase.py 123 222
Transaction committed
$ python display.py

books
-----
{'_id': ObjectId('5f4080b934ae0de8ff82675a'), 'isbn': '123', 'title': 'The Practice of Programming', 'authors': ['Kernighan', 'Pike'], 'quantity': 499}
{'_id': ObjectId('5f4080b934ae0de8ff82675b'), 'isbn': '234', 'title': 'The C Programming Language', 'authors': ['Kernighan', 'Ritchie'], 'quantity': 800}
{'_id': ObjectId('5f4080b934ae0de8ff82675c'), 'isbn': '345', 'title': 'Algorithms in C', 'authors': ['Sedgewick'], 'quantity': 650}
-----

zipcodes
-----
{'_id': ObjectId('5f4080b934ae0de8ff82675d'), 'zipcode': '08540', 'city': 'Princeton', 'state': 'NJ'}
{'_id': ObjectId('5f4080b934ae0de8ff82675e'), 'zipcode': '02138', 'city': 'Cambridge', 'state': 'MA'}
{'_id': ObjectId('5f4080b934ae0de8ff82675f'), 'zipcode': '02142', 'city': 'Cambridge', 'state': 'MA'}
-----

customers
-----
{'_id': ObjectId('5f4080b934ae0de8ff826760'), 'custid': '111', 'custname': 'Princeton', 'address': {'street': '114 Nassau St', 'zipcode': ObjectId('5f4080b934ae0de8ff82675d')}}
{'_id': ObjectId('5f4080b934ae0de8ff826761'), 'custid': '222', 'custname': 'Harvard', 'address': {'street': '1256 Mass Ave', 'zipcode': ObjectId('5f4080b934ae0de8ff82675e')}}
{'_id': ObjectId('5f4080b934ae0de8ff826762'), 'custid': '333', 'custname': 'MIT', 'address': {'street': '292 Main St', 'zipcode': ObjectId('5f4080b934ae0de8ff82675f')}}
-----

orders
-----
{'_id': ObjectId('5f4080b934ae0de8ff826763'), 'isbn': '123', 'custid': '222', 'quantity': 21}
{'_id': ObjectId('5f4080b934ae0de8ff826764'), 'isbn': '345', 'custid': '222', 'quantity': 100}
{'_id': ObjectId('5f4080b934ae0de8ff826765'), 'isbn': '123', 'custid': '111', 'quantity': 30}
$
```

MongoDB Pgmming in Python

- See **Mongo/purchase.py** (cont.)
 - MongoDB 4.0 and newer supports transactions

MongoDB Pgmming in Python

- See [Mongo/recovery.py](#)

```
Mongo -- -bash -- 38x28
$ python create.py
$ python recovery.py
Transaction 0 committed.
Transaction 1 committed.
Simulated failure with i = 2
Transaction rolled back.
Transaction 3 committed.
Transaction 4 committed.
Transaction 5 committed.
Transaction 6 committed.
Transaction 7 committed.
Transaction 8 committed.
Simulated failure with i = 9
Transaction rolled back.
Simulated failure with i = 10
Transaction rolled back.
Transaction 11 committed.
Simulated failure with i = 12
Transaction rolled back.
Transaction 13 committed.
Transaction 14 committed.
Transaction 15 committed.
Transaction 16 committed.
Transaction 17 committed.
Transaction 18 committed.
Simulated failure with i = 19
Transaction rolled back.
$
```

```
Mongo -- -bash -- 102x33
$ python display.py
-----
books
-----
<class 'pymongo.cursor.Cursor'>
{'_id': ObjectId('60260a63a777d356bb385c8e'), 'isbn': '123', 'title': 'The Practice of Programming', 'authors': ['Kernighan', 'Pike'], 'quantity': 485}
{'_id': ObjectId('60260a63a777d356bb385c8f'), 'isbn': '234', 'title': 'The C Programming Language', 'authors': ['Kernighan', 'Ritchie'], 'quantity': 800}
{'_id': ObjectId('60260a63a777d356bb385c90'), 'isbn': '345', 'title': 'Algorithms in C', 'authors': ['Sedgewick'], 'quantity': 650}
-----
zipcodes
-----
{'_id': ObjectId('60260a63a777d356bb385c91'), 'zipcode': '08540', 'city': 'Princeton', 'state': 'NJ'}
{'_id': ObjectId('60260a63a777d356bb385c92'), 'zipcode': '02138', 'city': 'Cambridge', 'state': 'MA'}
{'_id': ObjectId('60260a63a777d356bb385c93'), 'zipcode': '02142', 'city': 'Cambridge', 'state': 'MA'}
-----
customers
-----
{'_id': ObjectId('60260a63a777d356bb385c94'), 'custid': '111', 'custname': 'Princeton', 'address': {'street': '114 Nassau St', 'zipcode': ObjectId('60260a63a777d356bb385c91')}}
{'_id': ObjectId('60260a63a777d356bb385c95'), 'custid': '222', 'custname': 'Harvard', 'address': {'street': '1256 Mass Ave', 'zipcode': ObjectId('60260a63a777d356bb385c92')}}
{'_id': ObjectId('60260a63a777d356bb385c96'), 'custid': '333', 'custname': 'MIT', 'address': {'street': '292 Main St', 'zipcode': ObjectId('60260a63a777d356bb385c93')}}
-----
orders
-----
{'_id': ObjectId('60260a63a777d356bb385c97'), 'isbn': '123', 'custid': '222', 'quantity': 35}
{'_id': ObjectId('60260a63a777d356bb385c98'), 'isbn': '345', 'custid': '222', 'quantity': 100}
{'_id': ObjectId('60260a63a777d356bb385c99'), 'isbn': '123', 'custid': '111', 'quantity': 30}
$
```

MongoDB Pgmming in Python

- See **Mongo/recovery.py** (cont.)
 - Transactions work!

Stopping the MongoDB Server

- Linux
 - Stops automatically at logout
- Mac
 - At a bash shell prompt
 - Type Ctrl-c to abort `mongod` command
- MS Windows
 - At a command prompt
 - Type Ctrl-Fn-Pause to abort `mongod` command

MongoDB Assessment

- MongoDB assessment
 - (pro) Fast
 - (In old versions) no transactions => no locking, no commits, no rollbacks
 - (pro) Simple
 - No table/collection joins
 - (pro) Flexible data model
 - Good for storing hierarchical data
 - No (actually, optional) schemas
 - (pro) Uses OO model
 - No impedance mismatch

MongoDB Assessment

- MongoDB assessment (cont.)
 - (con) Less robust
 - (In old versions) no transactions => no locking, no commits, no rollbacks
 - (con) Maybe less convenient
 - No table/collection joins => extra logic in client
 - (con) Less data integrity
 - No (actually, optional) schemas

Summary

- The lecture has:
 - Provided *a taste of* the **MongoDB** DBMS
 - Given you enough information about **MongoDB** to:
 - Help you decide if you want to use it in your project
 - Help you get started with it