NoSQL and MongoDB

- Objectives
  - Discover the advantages of NoSQL and MongoDB
  - See why the industry is moving towards NoSQL
  - Use PyMongo to access and update MongoDB
  - Learn MongoDB’s query language

History: Starting from RDBMSs is an axiom

- Starting from an RDBMS continues to be an axiom of software development.

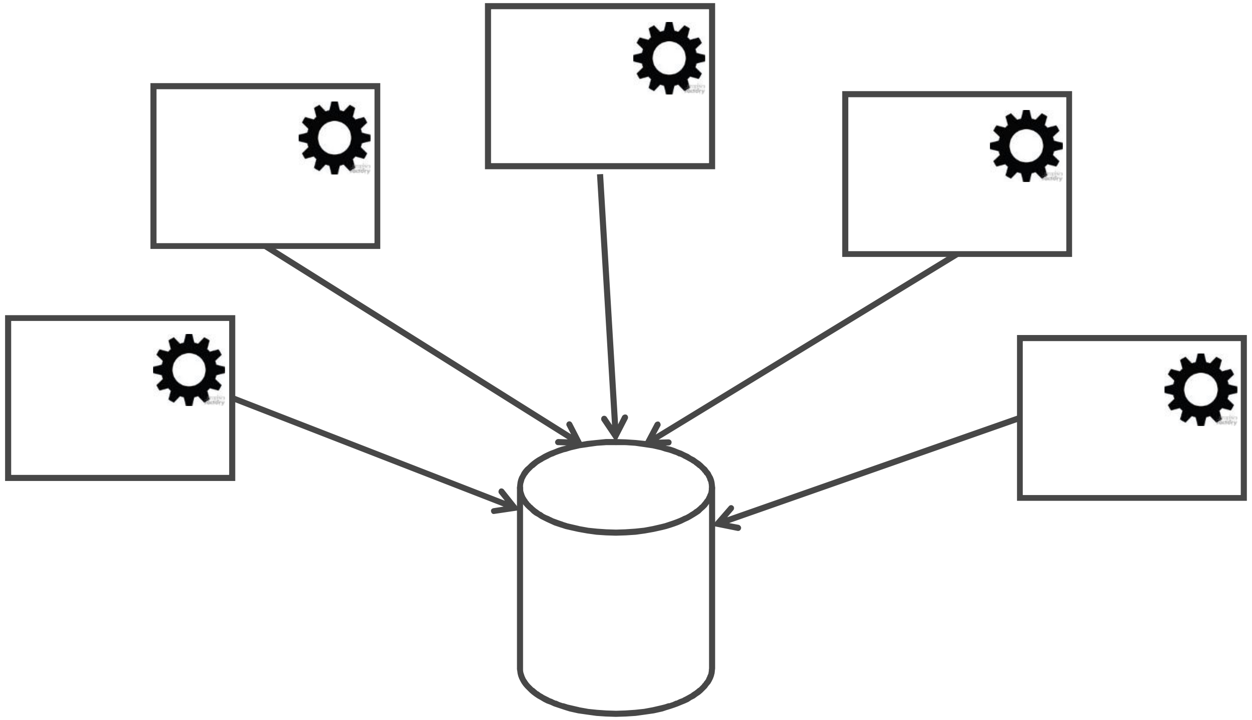
When was the last time you consciously evaluated alternatives to an RDBMS?

History: Why has SQL persisted so long?

- **Experience**: Industry experience in reliably running RDBMSs is useful.
- **Tooling**: Many many tools speak SQL and understand RDBMSs.
- **ACID**: RDBMSs typically provide app safety via ACID properties.
- **DBAs**: There can be a professional divide between DBAs and developers.
- **Concurrency**: They tame the challenges of concurrency and failure.
- **Integration DBs** (1): RDBMSs have been used in large organizations as integration layers between many enterprise apps.

History: Why has SQL persisted so long?

- The integration database:



History: The industry is moving away from integration dbs

Integration database have issues:

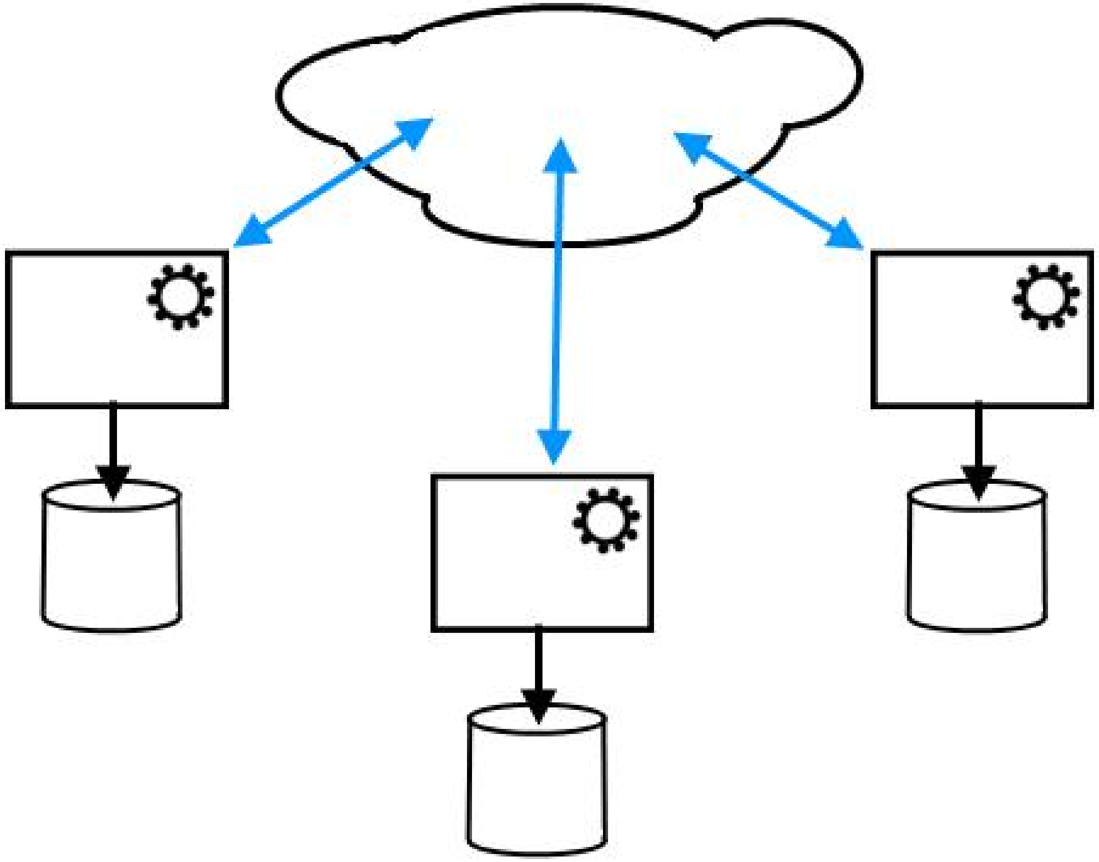
- Their schemas are often more complex (dramatically more so) than databases built for a single application (application databases).
- Complex coordination is required for every little change (innovation through change committees anyone?)
- They drive many important applications, can you scale them?
- They play a central role in “the rise of the DBA”.

History: Application database

- Many large IT groups are moving away from integration databases towards application databases (1) + services and SOA.
  - services (SOA) provide another way to design systems:
  - you access data through the service layer
  - application databases provide data for a single service

History: The industry is moving away from integration dbs

- The application database + services solution:



MongoDB: MongoDB is serious business

- Some companies using MongoDB:
  - http://www.mongodb.org/about/production-deployments/

craigslist

foursquare™

DEVELOPMENTOR

facebook.

CERN

github
SOCIAL CODING

sourceforge

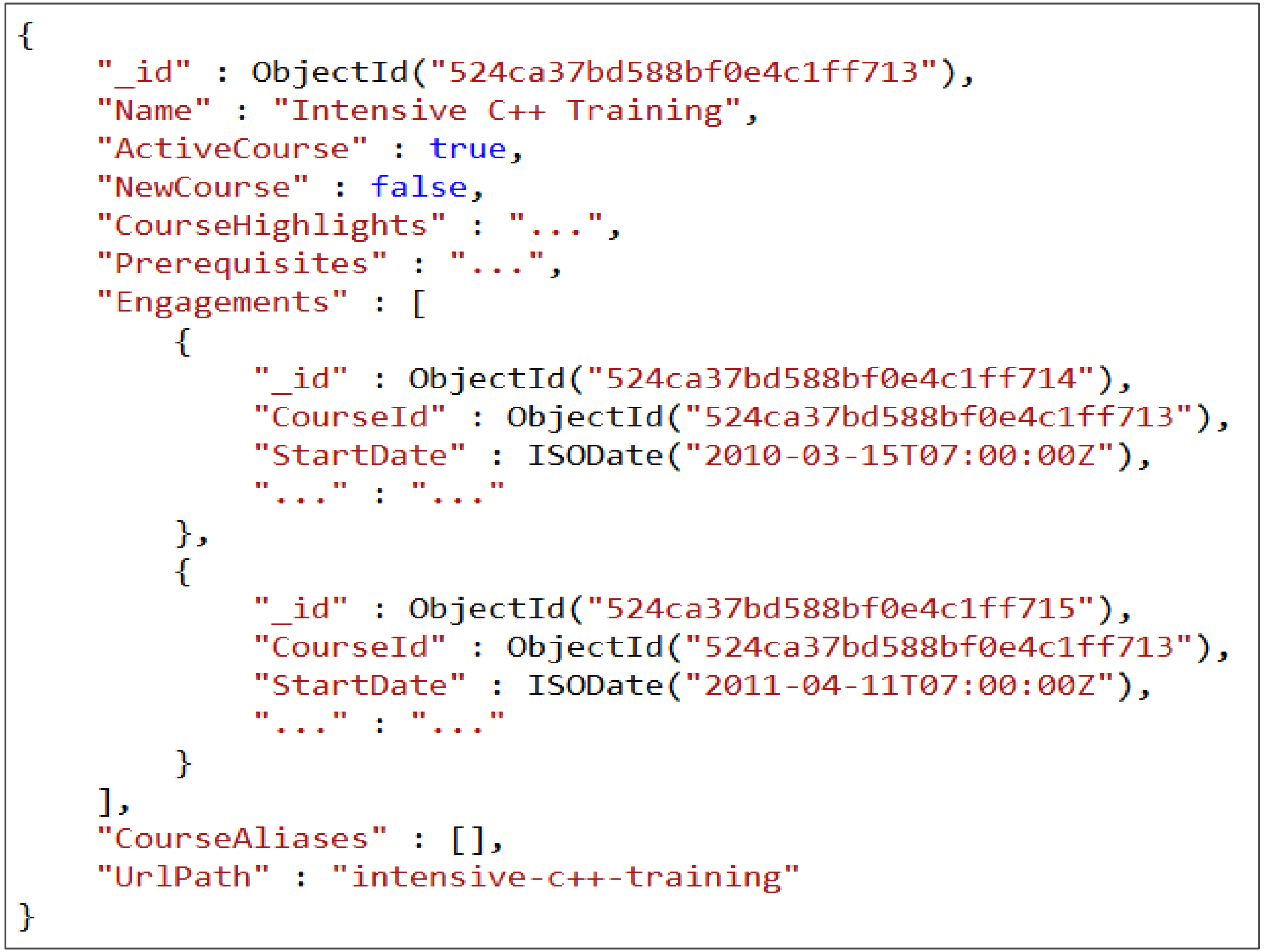
MTV
MUSIC TELEVISION

bit.ly

Disney

wordnik
All the words.

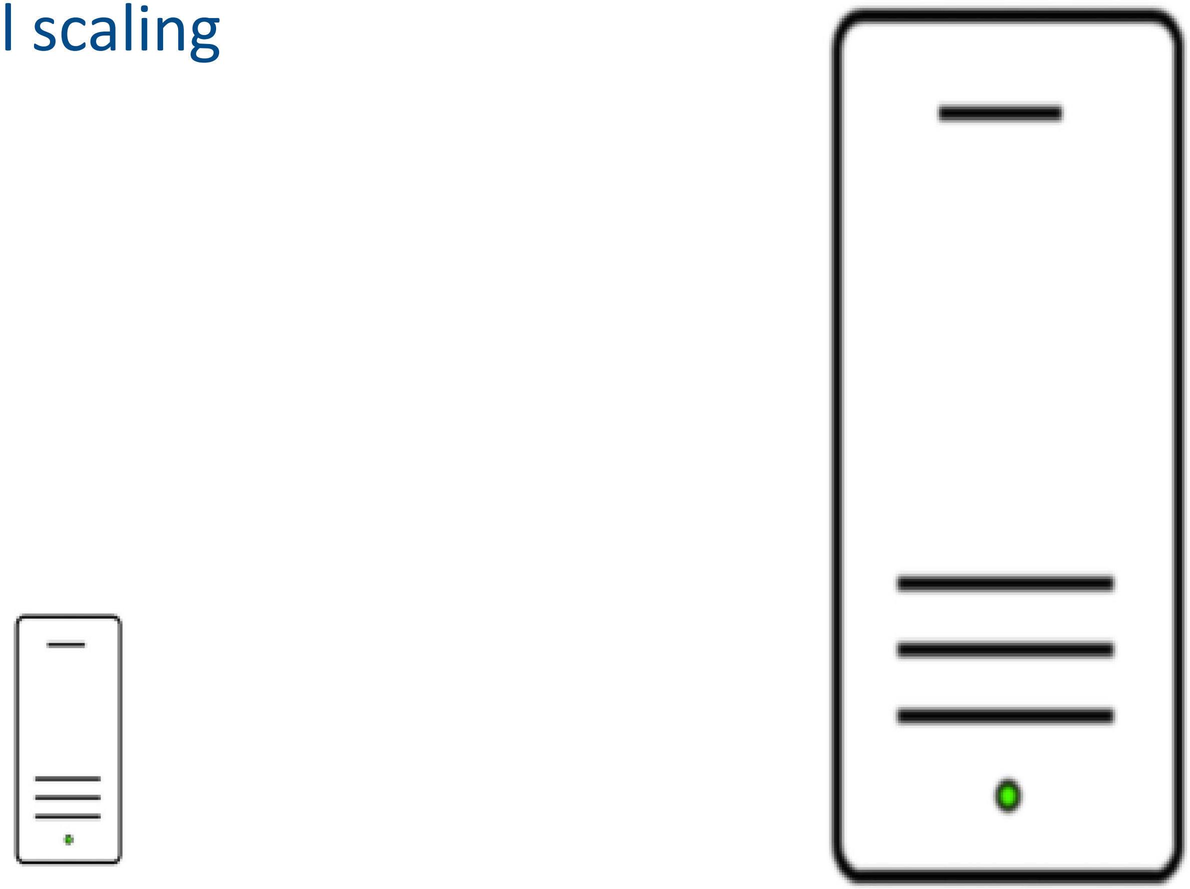
NoSQL: Document DBs - how do they store data?



```
{
    "_id" : ObjectId("524ca37bd588bf0e4c1ff713"),
    "Name" : "Intensive C++ Training",
    "ActiveCourse" : true,
    "NewCourse" : false,
    "CourseHighlights" : "...",
    "Prerequisites" : "...",
    "Engagements" : [
        {
            "_id" : ObjectId("524ca37bd588bf0e4c1ff714"),
            "CourseId" : ObjectId("524ca37bd588bf0e4c1ff713"),
            "StartDate" : ISODate("2010-03-15T07:00:00Z"),
            "..." : "..."
        },
        {
            "_id" : ObjectId("524ca37bd588bf0e4c1ff715"),
            "CourseId" : ObjectId("524ca37bd588bf0e4c1ff713"),
            "StartDate" : ISODate("2011-04-11T07:00:00Z"),
            "..." : "..."
        }
    ],
    "CourseAliases" : [],
    "UrlPath" : "intensive-c++-training"
}
```

Scaling: RDBMS

- Typically vertical scaling

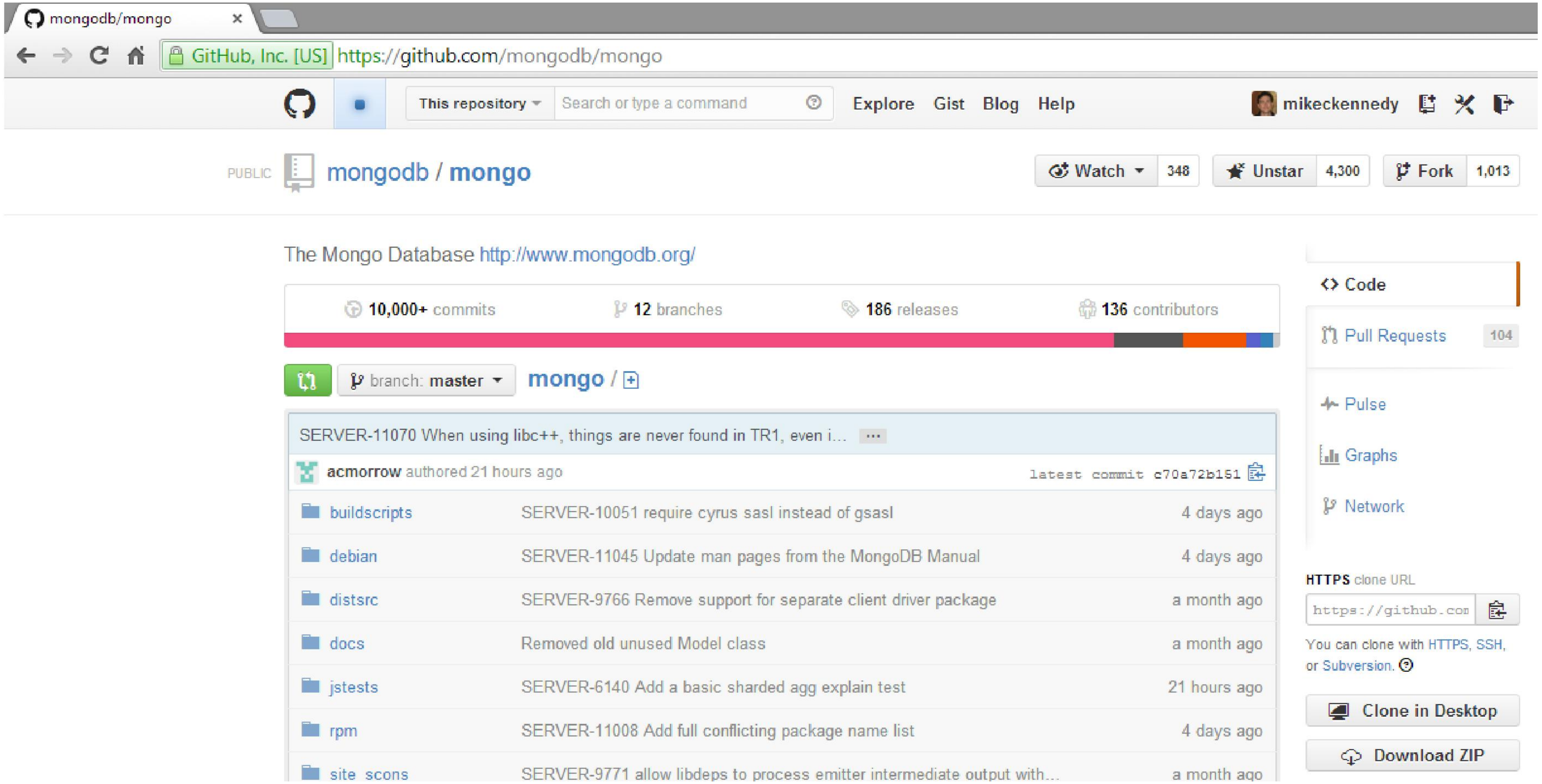


Start here
$ / perf.

Scale here
$$$$ / perf.

MongoDB: Open source

- Source code (server and drivers) on github
- Free download (pay for support)



The Mongo Database http://www.mongodb.org/

10,000+ commits · 12 branches · 186 releases · 136 contributors

SERVER-11070 When using libc++, things are never found in TR1, even i… …

acmorrow authored 21 hours ago — latest commit c70a72b161

buildscripts	SERVER-10051 require cyrus sasl instead of gsasl	4 days ago
debian	SERVER-11045 Update man pages from the MongoDB Manual	4 days ago
distsrc	SERVER-9766 Remove support for separate client driver package	a month ago
docs	Removed old unused Model class	a month ago
jstests	SERVER-6140 Add a basic sharded agg explain test	21 hours ago
rpm	SERVER-11008 Add full conflicting package name list	4 days ago
site_scons	SERVER-9771 allow libdeps to process emitter intermediate output with…	a month ago

You can clone with HTTPS, SSH, or Subversion.

MongoDB: Getting MongoDB

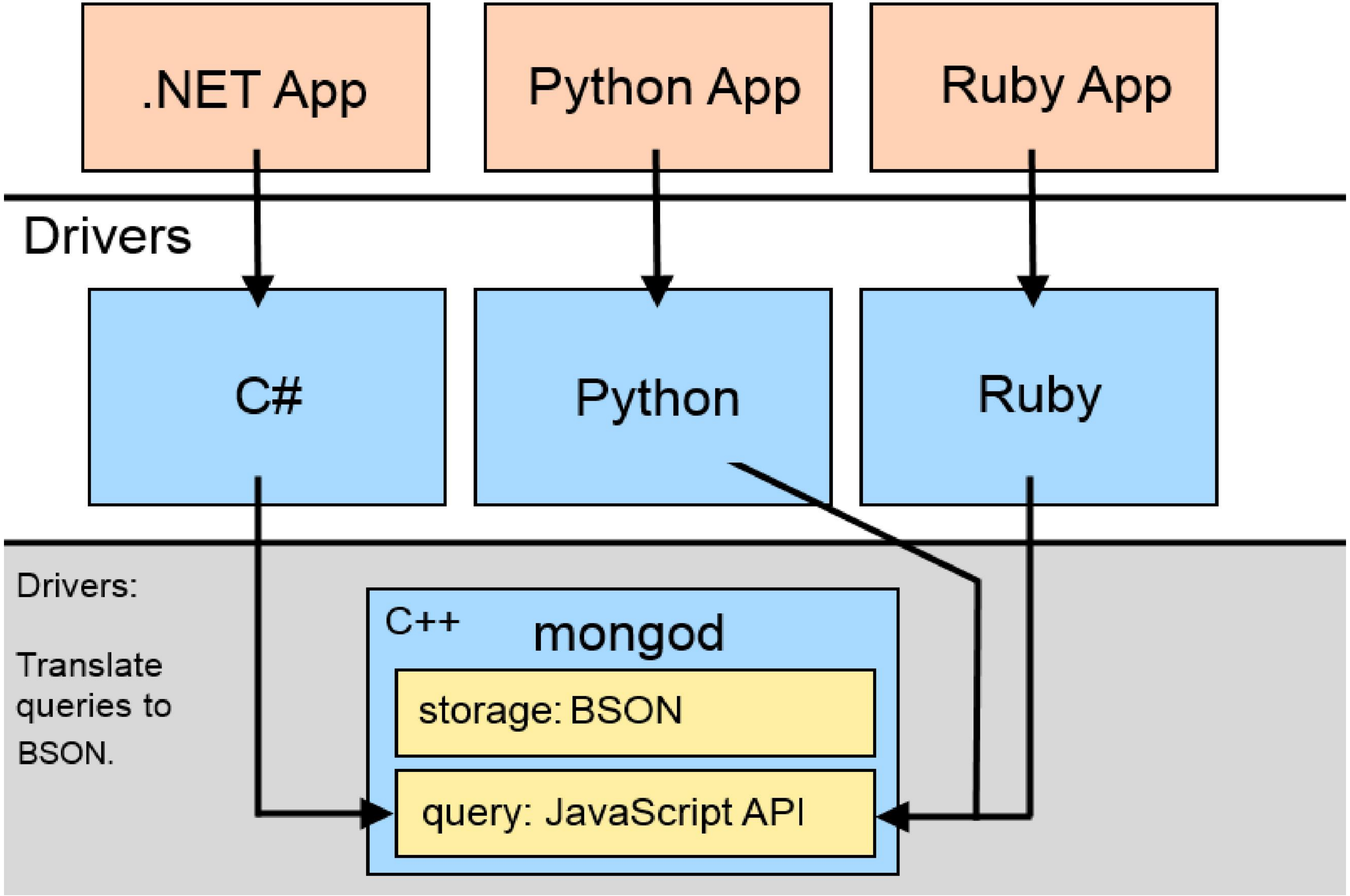
- Download your OS’s version here
  - free, no registration required,
  - http://www.mongodb.org/downloads (64-bit is preferred)
- Install MongoDB by decompressing it
  - you may want to set it to start up as a Windows Server or system daemon
- Create a data folder and log folder
- Start the server

Estimated time: First time, less than 10 min, with experience, less than 3 min.

MongoDB: Management tools

- **Robomongo**
  - Free / Open-source, Windows / OS X / Linux
  - http://robomongo.org/
- **Mongo.exe** – MongoDB’s native shell
  - Free / Open-source, Windows / OS X / Linux

The internals of MongoDB



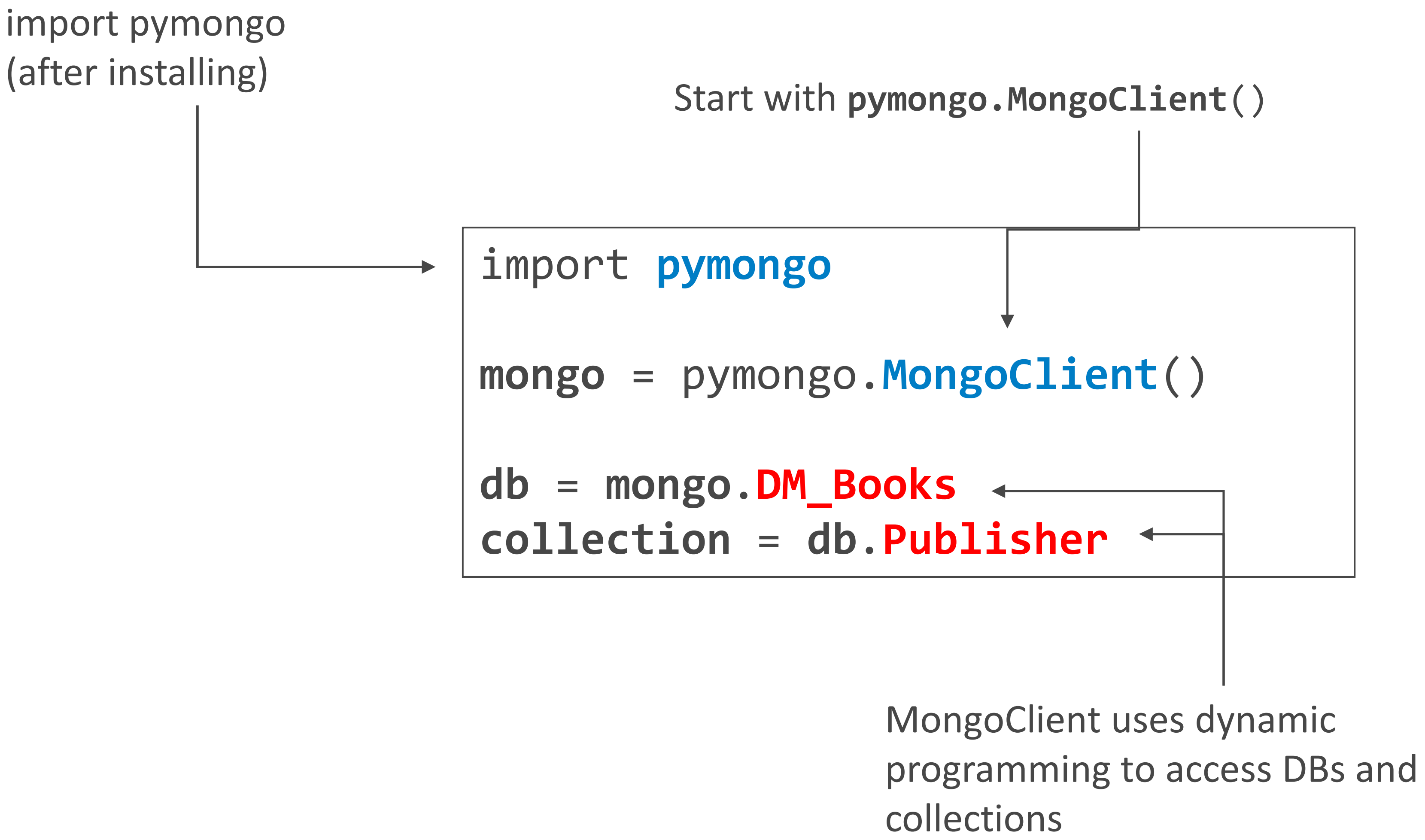
Drivers:

Translate queries to BSON.

PyMongo

- MongoDB has an official Python driver
  - **pymongo**: https://pypi.python.org/pypi/pymongo
- Tutorials and documentation from MongoDB
  - Python Language Center http://docs.mongodb.org/ecosystem/drivers/python/
- Open-source on Github
  - https://github.com/mongodb/mongo-python-driver
- Supports
  - Python 3 and Python 2
  - Windows, OS X, Linux
- Installing pymongo from the installers on PyPI is preferred

PyMongo [connecting]



import pymongo
(after installing)

Start with **pymongo.MongoClient()**

```
import pymongo

mongo = pymongo.MongoClient()

db = mongo.DM_Books
collection = db.Publisher
```

MongoClient uses dynamic programming to access DBs and collections

Note: If **DM_Books** or **Publisher** does not exist, this is how you create them.

Querying for documents: find, find_one

- For simple queries, we use find and specify prototype dictionaries.

```
set = db.Collection.find( { '_id': 1 } )
# Think: SELECT * FROM Collection WHERE _id = 1

set = db.Collection.find( {'username': "mkennedy" } )
# Think: SELECT * FROM Collection WHERE username = 'mkennedy'

doc = db.Collection.find_one(
        {'username': "pierre", 'passwordhash': "B8E28A21" } )
# Think: SELECT TOP(1) * FROM Collection
#        WHERE username = 'pierre' AND ' passwordhash = 'B8E28A21'
```

Querying for documents: results

The return value from find is a cursor.

```
query = db.Collection.find( {'Category': 'NoSQL'} )

# Get the number of records with count
numOfRecords = query.count()

# Pull all documents into memory as a list of dicts
memoryList = list(query)

# Stream documents to the app via for/in loops
for doc in query:
    print(doc['Title'])
```

PyMongo [viewing results]

- Highly-nested data can be pretty printed for readability

```
import pprint

doc = db.Collection.find_one( {'Category': 'NoSQL'} );

pprint.pprint(doc)
# prints

{
 'Author': 'Joe Vitale',
 'ISBN': '0759614318',
 'Published': datetime.datetime(2001, 1, 1, 8, 0),
 'Publisher': ObjectId('5258672c3a93bb21980ffa8d'),
 'Category': 'NoSQL',
 'Title': 'Spiritual Databases: A ...',
 '_id': ObjectId('525867633a93bb2198137c81')
}
```

Operators: Introduction

- How would you express this as a prototypical dictionary?
  - `SELECT * FROM Users WHERE RegistrationDate > @date`
- You cannot, which is why we need **$operators**. e.g.

```
db.Users.find( { 'registrationDate': {'$gt': date } } )
```

Operators: Inequalities and existence

```
db.Users.find( {'registrationDate': {'$gt': date } } )
```

- **$gt** - greater than
- **$gte** - greater than or equal to
- **$lt** - less than
- **$lte** - less than or equal to
- **$ne** - not equal
- **$exists** - the field exists in this document

Operators: Inequalities and existence

- Operator example, find non-null email addresses:

```
db.Users.find( {'email': {'$ne': None } } )
```

Operators: $or and $and

- Often you need to combine two filters using OR or AND:

```
db.Users.find( {'$and':
       [
              {'email': {'$ne': None } },
              {'registrationDate': {'$gt': date } }
       ]
} )
```

- Note: $and and $or expect an array of conditions.

Operators: inside arrays ($in, $all)

- It is very common to traverse a weak foreign key using two queries.

```
{ // category
   _id: "science",
   bookIds: [1, 5, 93, 20, 11]
   // more items
}
```

```
cat = db.Categories.find( {'_id': 'science' } )
books = db.Books.find( {'_id': {'$in': cat.bookIds } } )
```

Updating documents: Entire documents

- We can treat MongoDB as an ORM.
  - get document
  - make changes
  - save document back to DB

```
user = db.Users.find_one( { '_id': 72 } )
user['hasPaid'] = True
user['expirationDate'] = newEndDate
db.Users.save(user)
```

Updating documents: By field, atomically

- **$set** atomically updates the document without retrieving it.

```
db.Users.update(
{ '_id': 72 },
{ '$set':
   {
      'hasPaid': True,
      'expirationDate' = newEndDate
   }
} )
```

Updating multiple records

- By default, the update() method updates a single document. If the multi option is set to true, the method updates all documents that match the query criteria.

```
db.Users.update(
  { 'hasPaid': True },
  { '$set': { expirationDate = newEndDate },
  { 'multi': True }
)
```

Warning: This goes against your intuition.
SQL updates everything that matches by default.

Deleting / removing documents

- Delete documents with db.collection.remove:

```
# remove all non paying users.
db.Users.remove( { 'hasPaid': False} )
```

Note: For large deletion operations, it may be more efficient to copy the documents that you want to keep to a new collection and then use drop() on the original collection.

Summary

- MongoDB is a cluster-friendly, scalable database
- Simpler programming models and application DBs are leading developers down the NoSQL path
- PyMongo is the official MongoDB driver for Python
- MongoDB has a prototypical document-based query language