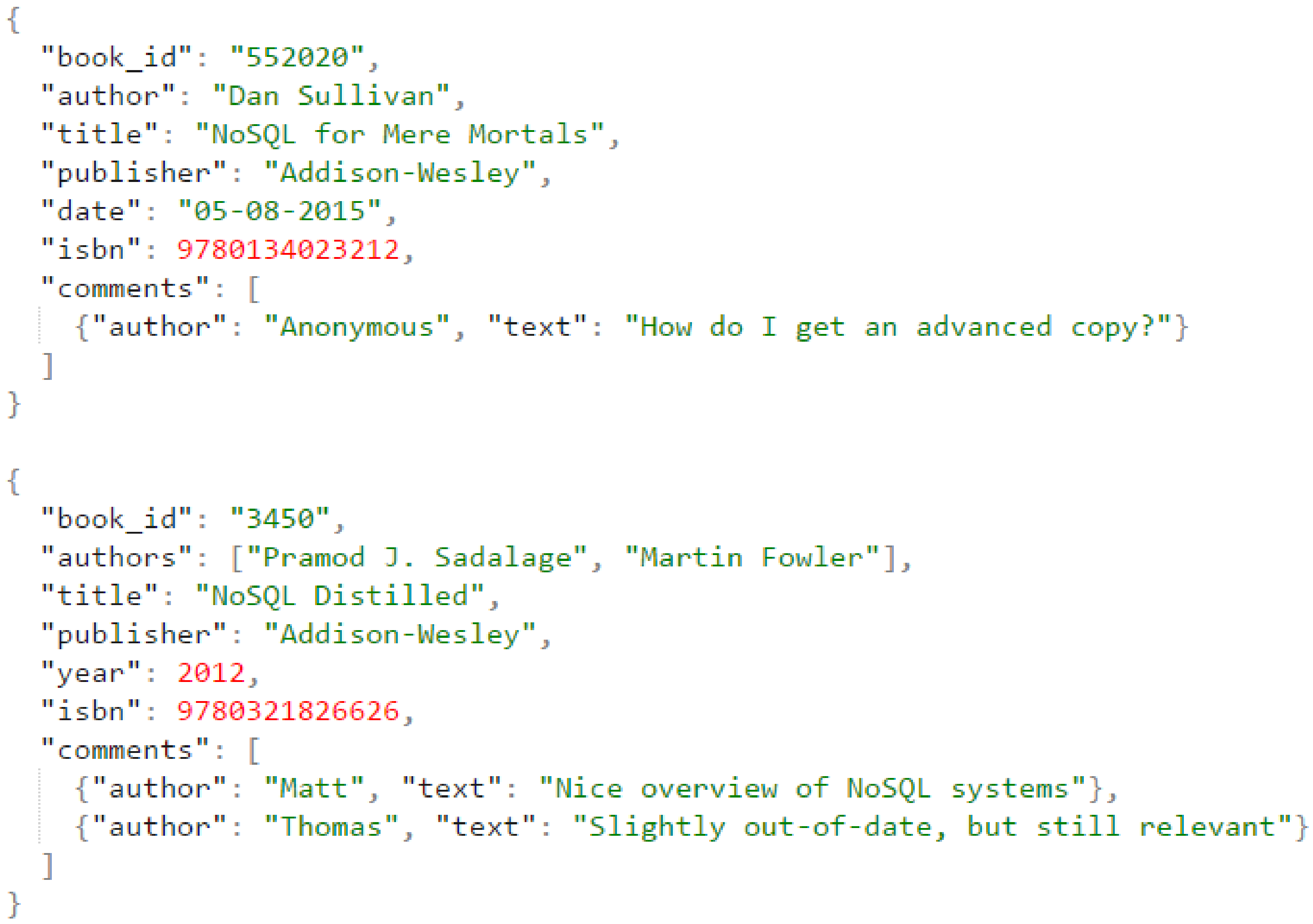
MongoDB Roadmap

- Data model
  - JSON syntax
  - Semi-structured data
- Query language
- Inserts, updates, deletes
- Replication and “sharding”
- “Eventual” consistency

Recall: Sample Documents for Queries



```
{
  "book_id": "552020",
  "author": "Dan Sullivan",
  "title": "NoSQL for Mere Mortals",
  "publisher": "Addison-Wesley",
  "date": "05-08-2015",
  "isbn": 9780134023212,
  "comments": [
    {"author": "Anonymous", "text": "How do I get an advanced copy?"}
  ]
}

{
  "book_id": "3450",
  "authors": ["Pramod J. Sadalage", "Martin Fowler"],
  "title": "NoSQL Distilled",
  "publisher": "Addison-Wesley",
  "year": 2012,
  "isbn": 9780321826626,
  "comments": [
    {"author": "Matt", "text": "Nice overview of NoSQL systems"},
    {"author": "Thomas", "text": "Slightly out-of-date, but still relevant"}
  ]
}
```

Recall: Find functions

```
db.collection.find({query},{projection})
```

```
db.collection.findOne({query},{projection})
```

Example:

```
db.posts.find({"author" : "Dan Sullivan"}, {"title" : 1})
```

```
Result: { "_id" ObjectId("5537dae716fb8743d12c5a60"),
                "title" : "NoSQL for Mere Mortals"}
```

FindOne

```
db.books.findOne({}, {"book_id" : 1, "title" : 1, "_id" : 0})
```

```
Result: {"book_id" : "552020",
              "title" : "NoSQL for Mere Mortals"}
```

```
db.books.findOne({"publisher" : "Addison-Wesley"},
{"title" : 1, "_id" : 0})
```

```
Result: {"title" : "NoSQL for Mere Mortals"}
```

Query operators

- **$lt** – Less than
- **$let** – Less than or equal to
- **$gt** – Greater than
- **$gte** – Greater than or equal to
- **$in** – Query for values of a single key
- **$or** – Logical or
- **$and** – Logical and
- **$not** - Negation

Range Query

```
db.books.find({"year" :  {"$gte" : 2012, "$lte" : 2015}})
```

Result:

```
{   "book_id": "3450",
      "authors": ["Pramod J. Sadalage", "Martin Fowler"],
      "title": "NoSQL Distilled",  "publisher": "Addison-Wesley",
      "year": 2012,
      "isbn": 9780321826626,
      "comments": [
            {"author": "Matt", "text": "Nice overview of NoSQL systems"},
            {"author": "Thomas", "text": "Slightly out-of-date, but still
             relevant"}]
  }
```

In, Or Queries

```
db.books.find({"isbn": {"$in": [9876543210, 0123456789]}})
```

Result: empty (there were no books with either ISBN)

```
db.books.find({"$or": [{"author" : "Dan Sullivan"},
                                   {title: "NoSQL for Mortals"}]})
```

Result:

```
{   "book_id" : "552020",  "author" : "Dan Sullivan",
     "title" : "NoSQL for Mere Mortals",
     "publisher" : "Addison-Wesley",  "date" : "05-08-2015",
      "isbn" : 9780134023212,
      "comments“ : [ {"author" : "Anonymous", "text" : "How do I get
                           my advanced copy?"} ]
}
```

Negation Query

```
db.books.find({"book_id" : {"$ne" : 552020}})
```

Result:

```
{   "book_id" : "3450",
      "authors" : ["Pramod J. Sadalage", "Martin Fowler"],
      "title" : "NoSQL Distilled",  "publisher": "Addison-Wesley",
      "year" : 2012,
      "isbn" : 9780321826626,
      "comments" : [
           {"author" : "Matt", "text": "Nice overview of NoSQL systems"},
           {"author" : "Thomas", "text": "Slightly out-of-date, but still
            relevant"}]
  }
```

Querying Arrays

```
db.books.find({"authors" : "Martin Fowler"}, {"authors" : 1})
```

Result:

```
{ "authors" : [ "Pramod J. Sadalage", "Martin Fowler" ] }
```

```
db.books.find({"authors“ : ["Martin Fowler", "Pramod J.
Sadalage"]}, {"authors" : 1})
```

Result: empty (there were no authors listed in this order)

```
db.books.find({"authors": {$all: ["Pramod J. Sadalage",
"Martin Fowler"]}}, {"authors" : 1})
```

Result:

```
{ "authors" : [ "Pramod J. Sadalage", "Martin Fowler" ] }
```

Querying Objects

```
db.books.find({"comments.author" : "Anonymous"},
                                 {"comments.text" : 1})
```

Result:

```
{ "comments" : [ { "text" : "How do I get an advanced copy?"} ] }
```

```
db.books.find({"comments.author" : "Matt",
"comments.text" : "Nice overview of nosql systems"}
{title : 1}))
```

Result: empty (there were no comments.text with this exact match)

Limits, Skips, Sorts, Counts

- db.books.find().limit(10)
  - Limits the number of results to 10
- db.books.find().skip(3)
  - Skips the first three results and returns the rest
- db.books.find().sort({"author" : 1, "title" : -1})
  - Sorts by author ascending (1) and title descending (-1)
- db.books.find().count()
  - Counts the number of documents in the books collection

Inserts

```
doc = { "book_id" : "3450",
          "authors" : ["Pramod J. Sadalage", "Martin Fowler"],
          "title" : "NoSQL Distilled",  "publisher" : "Addison-Wesley",
          "year" : 2012,
          "isbn" : 9780321826626,
          "comments" : [
             {"author" : "Matt", "text": "Nice overview of NoSQL systems"},
             {"author" : "Thomas", "text": "Slightly out-of-date, but still
              relevant"}]
          }
db.books.insert(doc)
```

Result: `WriteResult({ "nInserted" : 1 })`

Updates and Deletes

```
db.books.update({"book_id" : "552020"}, {"price" : 35.20})
```

Result:

```
WriteResult({ "nMatched" : 0, "nUpserted" : 0, "nModified" : 0 })
```

```
db.books.update({"book_id" : "552020"}, {"price" : 35.20},
{ upsert: true } )
```

Result:

```
WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0 })
```

```
db.books.remove({"book_id" : “552020”})
```

Result:

```
WriteResult({ "nRemoved" : 1 })
```

Replacements

```
doc = { "book_id" : "3450",
          "authors" : ["Pramod J. Sadalage", "Martin Fowler"],
          "title" : "NoSQL Distilled",
          "publisher" : "Addison-Wesley",
          "year" : 2012,
          "isbn" : 9780321826626
        }
db.books.update({"book_id" : "3450"}, doc)
```

Result:

```
        WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

MongoDB Design Goals

- Want a data management system with properties:
  - Flexible schema (= semi-structured data model)
  - Highly-scalable (= support millions of transactions per second)

- To achieve goals, willing to give up:
  - Complex queries: e.g., give up on joins
  - Multi-document transactions
  - ACID guarantees: e.g., eventual consistency OK

Terminology

- **Replication** = Create multiple copies of each database partition. Replication can be synchronous or asynchronous. Spread queries across these replicas. Goals: scalability and availability.

- **Sharding** = horizontal partitioning by some key, and storing partitions on different servers. Data is de-normalized to avoid cross-shard operations (no distributed joins). Split the shards as data volumes or access grows. Goals: massive scalability.

Two-Phase Commit = Too Slow

- Phase 1:
  - Coordinator sends “Prepare to Commit”
  - Replicas make sure they can do so no matter what (write the action to a log to tolerate failure)
  - Replicas reply “Ready to Commit”

- Phase 2:
  - If all replicas ready, coordinator sends “Commit”
  - If any replicas failed, coordinator sends “Abort”

“Eventual” Consistency

- CAP Theorem: Trade-off between system availability, data consistency and tolerance to network partitions. You can only have 2/3 properties (Brewer, 2000)
- Eventual consistency = relaxed consistency = system always accepts writes, but reads may not reflect the latest updates
- Write conflicts will eventually propagate throughout the system. “Eventually” is undefined (sometime in the future)
- Eventual consistency implemented using vector clocks
- Approach pioneered by Amazon with Dynamo (2007)
- Adopted by MongoDB and majority of NoSQL systems

Vector Clocks

- A data item D has a set of [server, version] pairs
  where server = server name that wrote D
  and version = the version of D written by that server

- Suppose D([S1, v1]), [S2, v2]), then D represents
  version v1 for S1, version v2 for S2.

- If server Si updates D, then:
  – If (Si, vi) exists, it must increment vi to vi+1
  – Otherwise, it must create new entry (Si, v1)

Vector Clock Example

1. Client 1 writes data item D at server SX: **D = D([SX,V1])**
2. Client 2 reads **D([SX,V1])**, updates D, and this update is handled by server **SX**: **D = D([SX,V2])** (Note: [SX,V1] is garbage collected)
3. Client 3 reads **D([SX,V2])**, updates D and this update is handled by server **SY**: **D = D([SX,V2], [SY,V1])**
4. Client 4 reads **D([SX,V2])** (i.e. most recent write had not yet propagated), updates D and this update is handled by server **SZ**: **D = D ([SX,V2], [SZ,V1])**
5. Client 5 reads **D([SX,V2], [SY,V1])** from one replica and **D([SX,V2], [SZ,V1])** from a different replica: **Conflict!**

Detecting Conflicts

- Vector clocks let us detect conflicts. How? Need to understand what it means for a version to be derived from another version

- A data item D is an *ancestor* of D' if for all [S, v] ∈ D there exists [S,v'] ∈ D' s.t. $v \leq v'$

- Otherwise, D and D' are on parallel branches, and it means they have a conflict that needs to be reconciled by the application

In-class Exercise

D	D’	Conflict?	Newest Version
([SX,v3])	([SX,v5])	No	([SX,v5])
([SX,v3],[SY,v6])	([SX,v3],[SY,v6],[SZ,v2])		
([SX,v3],[SY,v10])	([SX,v3],[SY,v6],[SZ,v2])	Yes	N/A
([SX,v3],[SY,v10])	([SX,v3],[SY,v20],[SZ,v2])		
([SX,v3],[SY,v6])	([SX,v3],[SZ,v2])		

Quiz 7

Q1 (6 points): Consider the following JSON document that describes our class:

```
{
  "_id" : "33",
  "course" { "code" : cs327e, "title": “Elements of Databases”}
   "year" : 2015,
   "semester" : "Spring"
   "instructor" : “Shirley Cohen”,
   "prerequisites" : ["cs303"],
   ratings: nill
   last_modified: "04-22-2015"
}
```

a) find all the syntax errors in the JSON document and correct them.

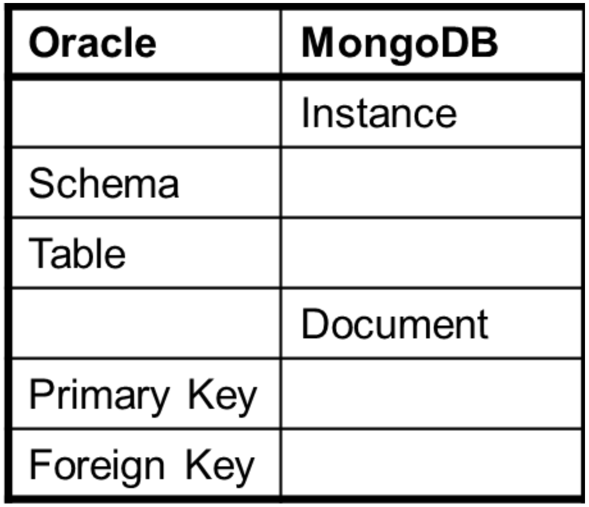
b) add another element for the number of students enrolled in the class. There are 66.

c) add a nested object with the TA's name (Yuming Sheng), her office hours times (Fridays 2-4pm), and location (TA Station Desk 5).

Quiz 7 (cont.)

Q2 (2 points): Explain the term “semi-structured data” and briefly describe its significance.

Q3 (2 points): Give analogous concepts between Oracle and MongoDB by filling out the table below. If no analog exists, write “none”.



Oracle	**MongoDB**
	Instance
Schema	
Table	
	Document
Primary Key	
Foreign Key	

Next Week

- Monday: Lighting Talks
- Wednesday: Review for Final