MongoDB Outline

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

Additional Resources on MongoDB

- http://docs.mongodb.org/manual/
  - Main source on MongoDB, but hard to read
- http://www.rfc-editor.org/rfc/rfc7159.txt
  - Authority on JSON
- http://www.mongodb.com/presentations/
  - Presentations on MongoDB

Reading: textbook chapter 9

Data Model

- A MongoDB instance contains a number of databases. A **database** holds a set of collections. A **collection** holds a set of documents.

- A document = JSON object = set of unordered key-value pairs = nested or not = schema-less

- Other NoSQL document stores: CouchDB, Couchbase, SimpleDB, Terrastore

MongoDB as Semi-structured Data

- Relational databases have rigid schema
  - Schema evolution is costly
- MongoDB is flexible: semi-structured data model
  - Store and query data in JSON

- Warning: not normal form. Not even 1NF!

JSON Syntax


```
{
  "business_id": "vcNAWiLM4dR7D2nwwJ7nCA",
  "full_address": "4840 E Indian School Rd\nSte 101\nPhoenix, AZ 85018",
  "open": true,
  "categories": [
    "Doctors",
    "Health & Medical"
  ],
  "city": "Phoenix",
  "review_count": 9,
  "name": "Eric Goldberg, MD",
  "neighborhoods": [],
  "longitude": -111.983758,
  "state": "AZ",
  "stars": 3.5,
  "latitude": 33.499313,
  "type": "business",
  "hours": {
    "Tuesday": {
      "close": "17:00",
      "open": "08:00"
    },
    "Friday": {
      "close": "17:00",
      "open": "08:00"
    }
  }
}
```


Basic constructs

- Base values
  number, string, boolean, null
- Objects { }
  sets of key-value pairs
- Arrays [ ]
  lists of values

JSON describes the content

JSON Terminology

- JSON object: set of unordered elements
- elements: key/value pairs
- keys: “business_id”, “full_address”, “open”, …
- keys must be unique within an object
- values: true, 9, “AZ”, [“Doctors”, “Health & Medical”]
- values can contain objects
- empty value: null, [] (or simply omit element)

well-formed JSON object: elements surrounded by curly braces

Comparison

MongoDB	Oracle
Database	Schema
Collection	Table
Document	Record
_id field	Primary Key

More syntax: _id and references

```
{
   "_id" : "555",
   "name" : "Jane"
}
{
   "_id" : "444",
   "name" : "Sarah",
   "mother" : "555"
}
```

MongoDB documents in a collection must have unique identifier

Documents can be referenced using unique identifier

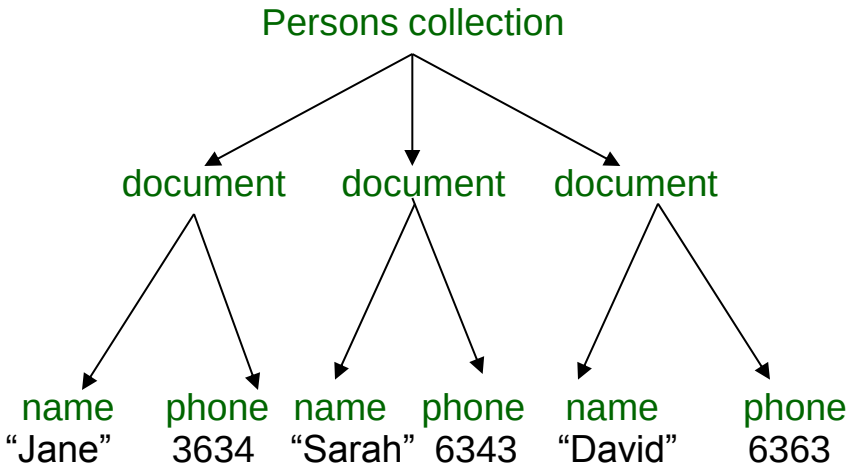
References in JSON are just syntax

JSON as *Data*

- JSON is self-describing
- Keys become part of the data
  - Relational schema: persons(name, phone)
  - In JSON “name”, “phone” are part of the data, and are repeated many times
- Consequence: JSON is much more flexible
- JSON = semi-structured data

Mapping Relational Data to JSON

Canonical mapping:



Persons Table

Name	Phone
Jane	3634
Sarah	6343
David	6363

Persons Collection

```
{ "name" : "Jane",
  "phone" : 3634 }
{ "name" : "Sarah",
  "phone" : 6343 }
{ "name" : "David",
  "phone" : 6363 }
```

Mapping Relational Data to JSON

Natural mapping:

Customers

Id	Name	Phone
10	Jane	3634
12	Sarah	6343

Orders

Id	Cust_Id	Date	Product
505	10	04-20-15	Apple Watch
500	10	04-19-15	iPhone6
100	12	04-01-14	MacBook

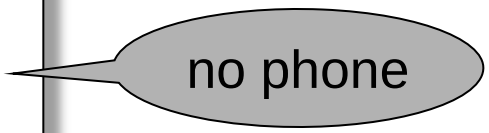
Customers

```
{
   "_id" : 10,
   "name" : "Jane",
   "phone" : 3634,
   "orders" :  [{"_id" : 505,
                 "date" : "04-20-15",
                 "product" : "Apple Watch"},
                 {"_id" : 500,
                 "date" : "04-19-15",
                 "product" : "iPhone6" }]
}
{
   "_id" : 12,
   "name" : "Sarah",
   "phone" : 6343,
   "orders" :  [{"_id" : 100,
                 "date" : "04-01-14",
                 "product" : "MacBook"}]
}
```

JSON is Semi-structured Data

- Missing elements:

```
{“name” : “Jane”,
  “phone” : 3634
 }
 {
  “name” : “Jim”
 }
```



- Could represent in a table with nulls:

name	phone
Jane	1234
Jim	-

JSON is Semi-structured Data

- Repeated elements:

```
{
   “name” : “Jane”,
   “phones” : [3634, 2345]
 }
```

Two phones

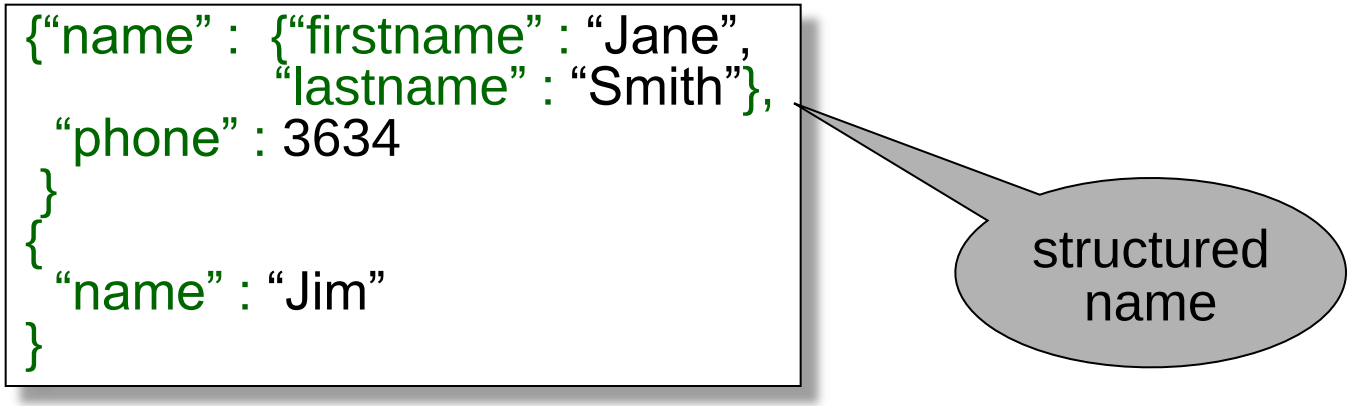
- Difficult with tables:

name	phone	
Jane	3456	2345

???

JSON is Semi-structured Data

- Elements with different types in different documents



- Nested objects (no 1NF)
- Heterogeneous documents:
  - collection contains documents with structured names and unstructured names

Typical MongoDB Applications

- Web applications
  - Content management systems
  - Ecommerce
  - Event logging
- Evolving schema
  - Quickly add a new element
- Data exchange
  - Take the data, don’t worry about schema

Approaches to JSON Processing

- Via API
  - Called DOM
  - Navigate, update the JSON arbitrarily
  - BUT: memory bound

- Via some query language:
  - MongoDB query language
  - Stand-alone processing in shell OR embedded in client-side program

MongoDB Operations

Will discuss next:

- query language
- insert, update, and delete

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"}
  ]
}
```

Find

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

- {query} = the search criteria
- {projection} = the fields to display

- Notice the use of “{“ and “}”

Find

```
db.books.find()
```

Result: all documents in book collection

```
db.posts.find({"author" : "Dan Sullivan"},
{"title" : 1, "book_id" : 1, " _id " : 0})
```

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

```
db.books.find({"author" : "Dan Sullivan"},
{"title" : 1, "_id" : 0})
```

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

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"}]
 }
```

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"}]
 }
```

Or Queries

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

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

```
db.books.find({"$or": [{"author": "Dan Sulivan"},
                       {isbn: 9780134023212}]})
```

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
                          an advanced copy?"} ]
}
```

Next Class

- Remainder of query language
- Replication and “sharding”
- Quiz 7