

MongoDB

Hands on test drive
Dan Van Valin & Kevin McGinty



What is MongoDB?

- NoSQL



What is MongoDB?

- No SQL
- Collections of Documents



Stores BSON Documents

```
{
```

```
  "_id" : ObjectId("5932cdb9c72a3fb0c0ecbd75"),  
  "title" : "The Incredibles",  
  "plot" : "A family of undercover superheroes, while trying to live the  
quiet suburban life, are forced into action to save the world.",  
  "actors" : [ "Craig T. Nelson", "Samuel L. Jackson" ],  
  "details" : {  
    "year" : "2004",  
    "rated" : "PG",  
    "genre" : "Animation",  
    "director" : "Brad Bird"
```



Stores BSON Documents

```
{
```

```
  "_id" : ObjectId("5932cdb9c72a3fb0c0ecbd75"),  
  "title" : "The Incredibles",  
  "plot" : "A family of undercover superheroes, while trying to live the  
quiet suburban life, are forced into action to save the world.",  
  "actors" : [ "Craig T. Nelson", "Samuel L. Jackson" ],  
  "details" : {  
    "year" : "2004",  
    "rated" : "PG",  
    "genre" : "Animation",  
    "director" : "Brad Bird"
```



Stores BSON Documents

```
{
```

```
  "_id" : ObjectId("5932cdb9c72a3fb0c0ecbd75"),  
  "title" : "The Incredibles",  
  "plot" : "A family of undercover superheroes, while trying to live the  
quiet suburban life, are forced into action to save the world.",  
  "actors" : [ "Craig T. Nelson", "Samuel L. Jackson" ],  
  "details" : {  
    "year" : "2004",  
    "rated" : "PG",  
    "genre" : "Animation",  
    "director" : "Brad Bird"
```



Stores BSON Documents

```
{
```

```
  "_id" : ObjectId("5932cdb9c72a3fb0c0ecbd75"),  
  "title" : "The Incredibles",  
  "plot" : "A family of undercover superheroes, while trying to live the  
quiet suburban life, are forced into action to save the world.",  
  "actors" : [ "Craig T. Nelson", "Samuel L. Jackson" ],  
  "details" : {  
    "year" : "2004",  
    "rated" : "PG",  
    "genre" : "Animation",  
    "director" : "Brad Bird"
```

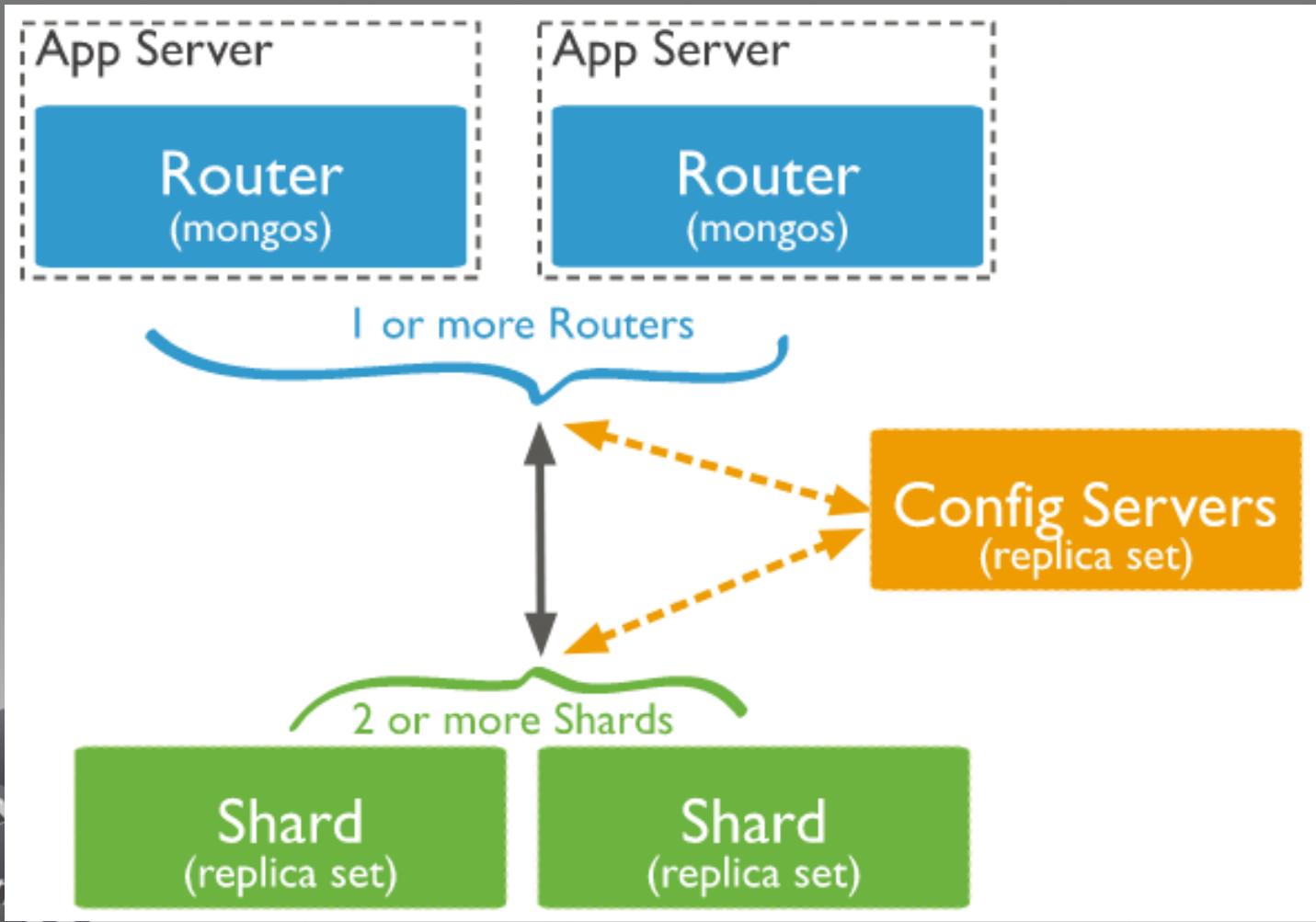


What is MongoDB?

- No SQL
- Collections of Documents
- Horizontal Scaling



Horizontal Scaling: Sharding

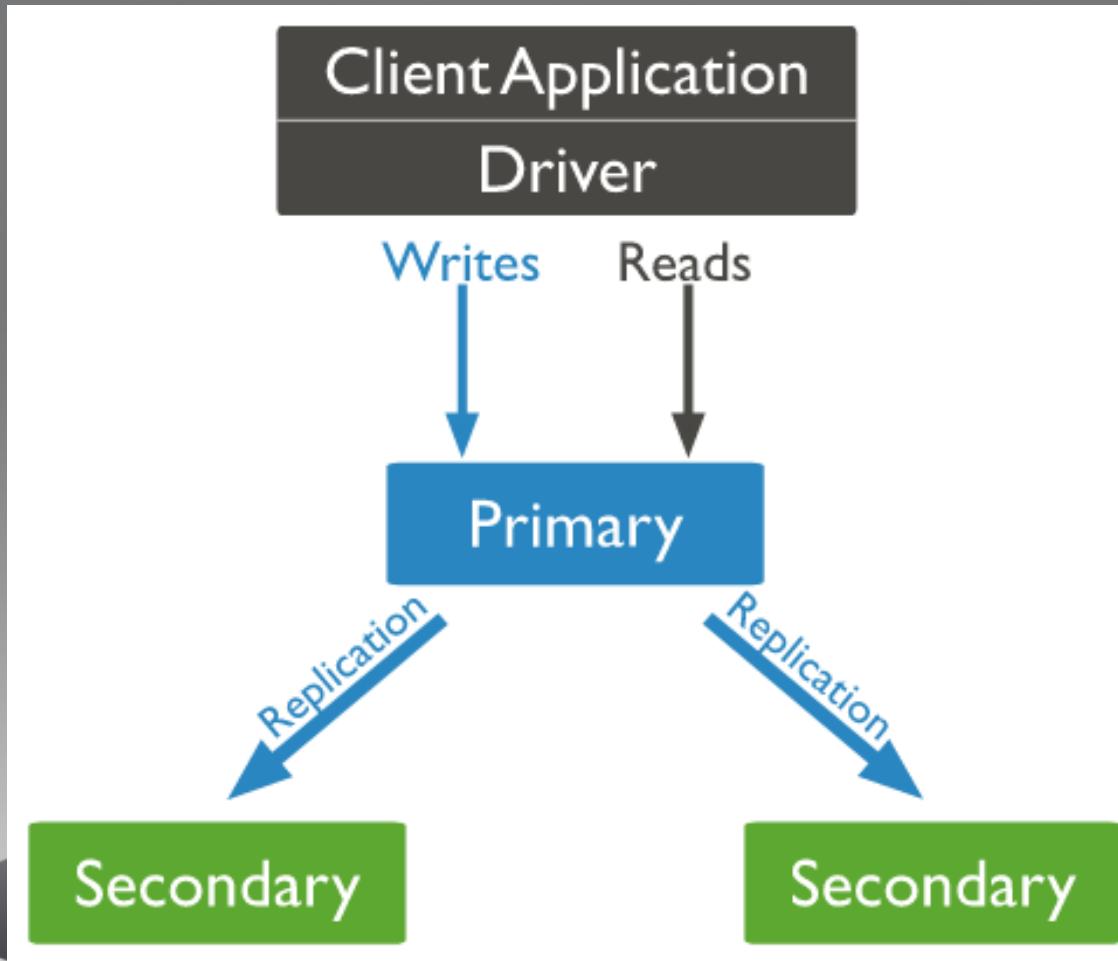


What is MongoDB?

- No SQL
- Collections of Documents
- Horizontal Scaling
- High Performance
- High Availability



High Availability: Replica Sets



<https://docs.mongodb.com/manual/replication/>



Starting MongoDB

- Start the server:

mongod

- Start the command shell (from IntelliJ Terminal helps):

mongo

- In the shell, display the mongo version running:

db.version()



Try Out The Shell

Some Simple Commands

List your databases:

show dbs

Start using a database (or create one) :

use <db name>

List the collections in the database:

show collections



Querying Documents

`db.<collection>.find()`
– queries everything

Check to see if you have the movies data inserted:

`db.movies.find()`
Should display 12 documents.



Shell Operators

Find can be followed with these shell operators:

.count()

.pretty()

.limit(rows)

.skip(rows)

.sort({"key": order})

order:1 for ascending, -1 for descending



Querying Documents

`db.<collection>.find(selector, projection)`

- Selector and projection look like javascript objects
- The simplest selector just matches object fields
- Projection limits the portion of the document returned
- List fields to display or remove: 1=display, 0=remove

`db.movies.find({title: "Batman"}, {plot: 1})`

- Returns the plot for Batman
- Also returns the `_id` field



Selectors

Search value can also be an operator:

\$lt less than

\$lte less than or equal to

\$gt greater than

\$gte greater than or equal to

\$ne not equal



db.movies.find({ title: { \$gt: "Spider" } })

Querying Documents

Conjunctions can be used with an array:

\$or

\$and

db.movies.find(

```
{$or: [ { title: "Batman" },  
        { title: "The Incredibles" } ] } )
```



Querying Documents

You can reference embedded documents with dot notation:

```
db.movies.find({"details.rated": "PG"})
```



Querying Exercise

1. Find all movie titles in our database with the actor Ian McKellen
2. List the names of directors for all movies
3. List the title and imdbRating for movies since 2000 or that have a rating greater than 8

Example: db.movies.find({ imdbRating: { \$gt: 8 } },

{title: 1, _id: 0})

Operators: \$lt, \$lte, \$gt, \$gte, \$ne



Solutions

1. Find all movie titles in our database with the actor Ian McKellen

```
> db.movies.find({actors: "Ian McKellen"}, {title: 1, _id: 0})
```

```
{ "title" : "X-Men" }
```

```
{ "title" : "The Hobbit: An Unexpected Journey" }
```

```
>
```



Solutions

2. List the names of directors for all movies

```
> db.movies.find({}, {"details.director": 1, _id: 0})  
[{"details": {"director": "Peter Jackson"}, ...}, {"details": {"director": "Bryan Singer"}, ...}, {"details": {"director": "Tim Burton"}, ...}, {"details": {"director": "Jon Favreau"}, ...}, {"details": {"director": "Patty Jenkins"}, ...}, {"details": {"director": "George Lucas"}, ...}, {"details": {"director": "Alex Proyas"}, ...}, {"details": {"director": "Sam Raimi"}, ...}, {"details": {"director": "Tim Story"}, ...}, {"details": {"director": "Peter Jackson"}, ...}, {"details": {"director": "Joss Whedon"}, ...}, {"details": {"director": "Brad Bird"}, ...}]
```



Solutions

3. List the title and imdbRating for movies since 2000 or that have a rating greater than 8

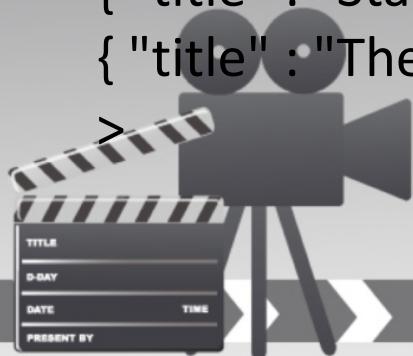
```
> db.movies.find({$or: [{"details.year":{$gte: 2000}},  
                        {imdbRating: {$gt: 8}}]},  
                  {title: 1, imdbRating: 1, _id: 0})
```

```
{ "title" : "The Lord of the Rings: The Fellowship of the Ring",  
  "imdbRating" : 8.8 }
```

```
{ "title" : "Wonder Woman", "imdbRating" : 8.4 }
```

```
{ "title" : "Star Wars: Episode IV - A New Hope", "imdbRating" : 8.7 }
```

```
{ "title" : "The Avengers", "imdbRating" : 8.1 }
```



Updates

Save: create or overwrite a document:

```
db.<collection>.save(document)
```

Update: a portion of a document:

```
db.<collection>.update(criteria, { op : portion },  
parameters)
```

Remove: remove documents:

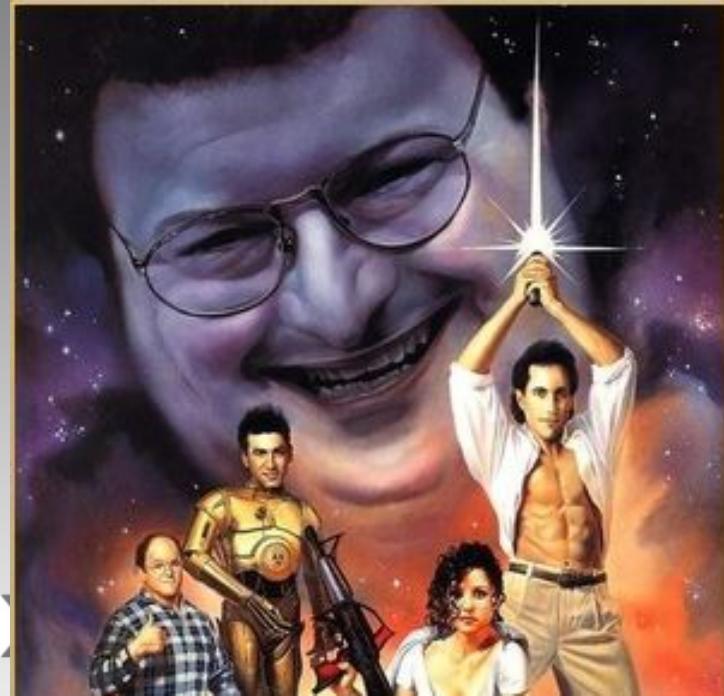
```
db.<collection>.remove(criteria, optional_limit)
```



Updates

`db.movies.update(selector, document with optional operator)`

```
db.movies.update({title: "Batman"},  
{$set: {boxOffice: "$251,188,924"}})
```



Array Operators

\$push

Adds an item to an array.

\$pop

Removes an item.

\$addToSet

Adds elements to an array only if they do not already exist in the set.



Updates Exercises

1. Add “Jack Palance” to the actors for Batman.
2. Add “Action” to the genre for “The Incredibles”.
3. Add “copies” with a value of 3 and “rented” with a value of 0 to every movie in the database.

Example: db.movies.update({title: "Batman"},

{\$set: {boxOffice: "\$251,188,924"}}, {multi: false})

Operators: \$set, \$push, \$pop, \$addToSet



Solutions

1. Add “Jack Palance” to the actors for Batman.

```
db.movies.update({title:"Batman"},  
{$push: {actors: "Jack Palance"}})
```



Solutions

2. Add “Action” to the genre for “The Incredibles”.

```
db.movies.update( { title:"The Incredibles" },  
{ $push: { "details.genre" : "Action" } } )
```



Solutions

3. Add “copies” with a value of 3 and “rented” with a value of 0 to every movie in the database.

```
db.movies.update({}, {$set: {copies: 3, rented: 0}},  
{multi: true})
```



Other Operators

\$inc

Increments the value of the field by the specified amount.

\$mul

Multiplies the value of the field by the specified amount.

\$unset

Removes the specified field from a document.

\$rename

Renames a field.



Atomic Updates

```
db.movies.findAndModify( {query: {selectors},  
    update:{operators}});
```

- Select the row to update and then update in a single operation



Other Features

- Validation
- Aggregation/Map Reduce
- Text Search
- Capped Collections
- Geospatial Indices and Queries
- Free training



MEAN example

- MongoDB
- Express
- Angular
- Node



Angular view - Creating

```
<div>
  <input type="text" placeholder="Enter a Todo"
         ng-keypress="save($event)" ng-model="todoInput">
</div>
```



Angular Controller - Creating

```
todosFactory.saveTodo({  
    todo: $scope.todoInput,  
    isCompleted: false  
}).then(function(data) {  
    $scope.todos.push(data.data);  
});  
$scope.todoInput = '';
```



Express Server - Creating

```
router.post('/api/todos', function(req, res) {  
  db.todos.insert(req.body, function(err, data) {  
    console.log(err);  
    res.json(data);  
  });  
});
```



Angular view - Update

```
<div ng-repeat="todo in todos | orderBy: '!highPriority'">
  ...
    <span ng-class="{'glyphicon glyphicon-star': todo.highPriority,
                  'glyphicon glyphicon-star-empty': !todo.highPriority}"
          ng-click="updatePriority($event, todo)">
    </span>
  ...
</div>
```



Angular Controller - Update

```
$scope.updatePriority = function (todo) {
    // Toggle priority
    var priority = !todo.highPriority;

    todosFactory.updateTodo({
        _id: todo._id,
        highPriority: priority,
        isCompleted: todo.isCompleted,
        todo: todo.todo
    }).then(function (data) {
        if (data.data.ok) {
            todo.highPriority = priority;
        } else {
            alert('Oops something went wrong!');
        }
    });
};

});
```



Express Server - Update

```
router.put('/api/todos', function(req, res) {  
  // Remove the ID, because you can't "update" an ID  
  
  var id = req.body._id;  
  delete req.body["_id"];  
  
  db.todos.update({  
    _id: mongojs.ObjectId(id)  
, req.body, {}, function(err, data) {  
    console.log(err);  
    res.json(data);  
  })  
});
```

