An Evening With MongoDB
An Evening With MongoDB
An Intro to MongoDB

@aaronheckmann

Evening with MongoDB Charlotte
Data model
Data model

• Document oriented
Data model

• vs Relational
Data model

• Relational

<table>
<thead>
<tr>
<th>Product</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td>name</td>
<td>created_at</td>
</tr>
</tbody>
</table>
### Data model

- **Relational**

<table>
<thead>
<tr>
<th>Product</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>name</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>created_at</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product_Attribute</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>product_id</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>key</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>val</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data model

- Relational
- assembly required

<table>
<thead>
<tr>
<th>Product</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td>name</td>
<td>created_at</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product_Attribute</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td>product_id</td>
<td>key</td>
<td>val</td>
</tr>
</tbody>
</table>
Data model

• Relational
  • assembly required
  • app model != storage model
Data model

• Document oriented
Data model

• Document oriented
• JSON
Data model

- Document oriented
- JSON-like
Data model

- Document oriented
- JSON-like
- BSON
Data model

- Document oriented
- JSON-like
  - BSON
    - basically typed JSON
    - number, string, binary, array, etc
Data model

Document oriented

{
    _id:     ObjectId("..")
 , name:   "Panthers T-shirt"
 , created_at:  ISODate("2012-08-15T15:42:09.195Z")
}

Wednesday, August 15, 12
Data model

Document oriented

{
  _id: ObjectId("..")
  , name: "Panthers T-shirt"
  , created_at: ISODate("2012-08-15T15:42:09.195Z")
  , props: [{ key: 'string', val: anything }]
}
Data model

• Document oriented
• Ad-hoc query support
Data model

Ad-hoc query support
Data model

Ad-hoc query support

• find, findOne
Data model

Ad-hoc query support

- find, findOne
- accept a conditions object
  - regular expressions
  - numbers
  - strings
  - etc
Data model

Ad-hoc query support

- `find`, `findOne`
- accept a conditions object
  - regular expressions
  - numbers
  - strings
  - etc
- rich operators
  - `$lt`, `$gt`, `$in`, `$ne`, ...
Data model

- Document oriented
- Ad-hoc query support
- Secondary indexing
Data model

Secondary indexing
Data model

Secondary indexing

• `createIndex()`
Data model

Secondary indexing

- `createIndex()`
- accepts an object
Data model

Secondary indexing

- `createIndex()`
- accepts an object
- options
  - unique, sparse, 2d, expiresAfterSeconds
Data model: gains
Data model: gains

• Dynamic schemas
Data model: gains

- Dynamic schemas
- Data modeled directly to app
Data model: gains

- Dynamic schemas
- Data modeled directly to app
- Retain ad-hoc queries
Data model: gains

- Dynamic schemas
- Data modeled directly to app
- Retain ad-hoc queries
- Retain secondary indexing
Data model: gains

- Dynamic schemas
- Data modeled directly to app
- Retain ad-hoc queries
- Retain secondary indexing
- Productivity
Data model: losses
Data model: losses

- Joins
Data model: losses

- Joins
- Multi-collection transactions
Data model: losses

- Joins
- Multi-collection transactions
  - use document level $atomics
Replication

- distribute data across multiple machines
Replication

- distribute data across multiple machines
- redundancy
Replication

- distribute data across multiple machines
- redundancy
- auto-failover
Replication

• read from master or replicas
Replication

- read from master or replicas
- all writes go to master
Replication

- read from master or replicas
- all writes go to master
  - configurable
    - `getLastError { w: 'majority' }`
Sharding
Sharding

- scale horizontally
Sharding

• scale horizontally
• range based partition mechanism
Sharding

- scale horizontally
- range based partition mechanism
- shard key
Sharding

- scale horizontally
- range based partition mechanism
- shard key
- apps talk to shard set-ups the same way
Mongo 2.2

- Concurrency improvements
Mongo 2.2

- Concurrency improvements
- db level locking
Mongo 2.2

- Concurrency improvements
- db level locking
- more yielding
Mongo 2.2

- Concurrency improvements
- Tag aware sharding
Mongo 2.2

- Concurrency improvements
- Tag aware sharding
- TTL collections
Mongo 2.2

• Concurrency improvements
• Tag aware sharding
• TTL collections
  • `ensureIndex({ date: 1 }, { expiresAfterSeconds: 60*15 })`
  • cannot be compound
  • cannot be used on capped collections
Mongo 2.2

- Concurrency improvements
- Tag aware sharding
- TTL collections
- Aggregation ...
MongoDB’s New Aggregation Framework

@aaronheckmann
Evening with MongoDB Charlotte
Aggregation in 2.0?
Map Reduce
Map Reduce

A LARGE hammer
Map Reduce

A LARGE hammer

- Complex analytics on big data
Map Reduce

A LARGE hammer

- Complex analytics on big data
- Distributed computing on clusters of machines
Problems
Problems

• Too complex
Problems

• Too complex
• Requires writing Javascript
Problems

• Too complex
• Requires writing Javascript
• Expensive
New Aggregation Framework
New Aggregation Framework

• Declarative
New Aggregation Framework

- Declarative
- C++ implementation
New Aggregation Framework

- Declarative
- C++ implementation
- Expression evaluation
New Aggregation Framework

• Declarative
• C++ implementation
• Expression evaluation
• Framework
Pipeline
Pipeline

• Think Unix pipes
Pipeline

- Think Unix pipes
- Single collection
Pipeline

- Think Unix pipes
- Single collection
- Series of operations
The Aggregation Command

db.runCommand({
    aggregate: "article",
    pipeline: [ {$op1, $op2, ...} ]
});
The Aggregation Command

- Takes two arguments

```javascript
db.runCommand({
    aggregate: "article",
    pipeline: [{ $op1, $op2, ... }]
});
```
The Aggregation Command

- Takes two arguments
  - `aggregate`: name of collection

```javascript
db.runCommand({
  aggregate: "article",
  pipeline: [ {$op1, $op2, ...} ]
});
```
The Aggregation Command

• Takes two arguments
  • `aggregate`: name of collection
  • `pipeline`: array of operations

```javascript
db.runCommand({
  aggregate: "article",
  pipeline: [
    {$op1, $op2, ...}]
});
```
Aggregation helper

db.article.aggregate(
    { $pipeline_op1 }
    , { $pipeline_op2 }
    , { $pipeline_op3 }
    , { $pipeline_op4 }
    , ...
);

Wednesday, August 15, 12
Pipeline Operations

- $match
  - query predicate - `coll.find(predicate)`
Pipeline Operations

- $match
  - query predicate - coll.find(predicate)
- $project
Pipeline Operations

- $match
  - query predicate - `coll.find(predicate)`
- $project
  - reshapes results
  - include / exclude fields
  - computed fields
Pipeline Operations

- $match
  - query predicate - `coll.find(predicate)`
- $project
  - reshapes results
- $unwind
  - hands out array elements one at a time in the context of their surrounding documents
Pipeline Operations

- **$match**
  - query predicate - `coll.find(predicate)`

- **$project**
  - reshapes results

- **$unwind**
  - hands out array elements one at a time

- **$group**
  - aggregates docs into buckets defined by a key
Pipeline Operations

- $group aggregation expressions
  - _id is the group key
  - $sum
  - $avg
  - $push, $addToSet
  - more..
    - $min, $max, $first, $last
Pipeline Operations

- **$sort**
  - sorts documents

- **$limit**
  - caps the number of documents

- **$skip**
  - steps over the specified number of documents
Computed Expressions

- Available in $project operations
Computed Expressions

• Available in $project operations
• Prefix expression language
  • Add two fields:
    • $add: ["$field1", "$field2"]
Computed Expressions

- Available in $project operations
- Prefix expression language
  - Add two fields:
    - $add: ["$field1", "$field2"]
  - Provide a value for a missing field:
    - $ifNull: ["$field1", "$field2"]
Computed Expressions

- Available in `$project` operations
- Prefix expression language
  - Add two fields:
    - `$add: ["$field1", "$field2"]`
  - Provide a value for a missing field:
    - `$ifNull: ["$field1", "$field2"]`
  - Nesting:
    - `$add: ["$field1", $ifNull: ["$field2", "$field3"]]

Wednesday, August 15, 12
Computed Expressions

- String functions
  - toUpper, toLower, substr, strcasecmp
Computed Expressions

- String functions
  - toUpper, toLower, substr, strcasecmp
- Date field extraction and arithmetic
  - Get year, month, day, hour, etc, from dates
Computed Expressions

- **String functions**
  - `toUpper`, `toLower`, `substr`, `strcasecmp`

- **Date field extraction and arithmetic**
  - Get year, month, day, hour, etc, from dates

- **Ternary conditional**
  - Return one of two values based on a predicate
Usage Tips

- Use $match as early as possible
Usage Tips

• Use $match as early as possible
• $sort (memory)
Usage Tips

- Use `$match` as early as possible
- `$sort` (memory)
- `$group`
  - like `$sort` but not as much memory is needed
Sharding support

• Mongos
Sharding support

- Mongos
  - forwards ops up to first $group or $sort to shards
Sharding support

- Mongos
  - forwards ops up to first `$group` or `$sort` to shards
  - combines shard server results and continues
Questions?

http://docs.mongodb.org/manual/release-notes/2.2/
http://docs.mongodb.org/manual/reference/aggregation/
http://www.mongodb.org/downloads

@aaronheckmann
@mongodb
Thank You!

http://docs.mongodb.org/manual/release-notes/2.2/
http://docs.mongodb.org/manual/reference/aggregation/
http://www.mongodb.org/downloads

@aaronheckmann
@mongodb