Getting Started with MongoDB

TCF IT Professional Conference
March 14, 2014

Michael P. Redlich
@mpredli
about.me/mpredli/
Who’s Mike?

- BS in CS from Rutgers
- “Petrochemical Research Organization”
- Ai-Logix, Inc. (now AudioCodes)
- Amateur Computer Group of New Jersey
- Publications
- Presentations
Objectives

- What is MongoDB?
- What is NoSQL?
- Getting Started with MongoDB
- Basic CRUD Operations
- Live Demos (yea!)
- MongoDB Resources
What is MongoDB? (I)

- "...an open-source document database that provides high performance, high availability, and automatic scaling."

MongoDB Web Site, http://www.mongodb.org/

- It’s name derived from “humongous”

- Written in C++
What is MongoDB? (2)

• “...an open-source database used by companies of all sizes, across all industries and for a wide variety of applications. It is an agile database that allows schemas to change quickly as applications evolve, while still providing functionality developers expect from traditional databases...”

What is NoSQL?

• Developed to address shortcomings of a traditional SQL relational database, namely:
  • big data
  • frequency of access to big data
  • performance and scalability
How is MongoDB Used?

Applications
- CRM, ERP, Collaboration, Mobile, BI

Data Management
- Online Data
  - MongoDB
  - RDBMS
- Offline Data
  - Hadoop
  - EDW

Infrastructure
- OS & Virtualization, Compute, Storage, Network
Who is Using MongoDB?
Features of MongoDB

• Document-Oriented Storage
• Full Index Support
• Replication and High Availability
• Auto-Sharding
• Querying
• Fast In-Place Updates
• Map/Reduce
• GridFS
• Professional Support by MongoDB
## Nomenclature (1)

<table>
<thead>
<tr>
<th>RDBMS</th>
<th>MongoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Database</td>
</tr>
<tr>
<td>Table</td>
<td>Collection</td>
</tr>
<tr>
<td>Row</td>
<td>Document</td>
</tr>
<tr>
<td>Index</td>
<td>Index</td>
</tr>
<tr>
<td>Join</td>
<td>Embedding &amp; Linking</td>
</tr>
<tr>
<td>Foreign Key</td>
<td>Reference</td>
</tr>
</tbody>
</table>
Nomenclature (2)
What is a Document?

- Basic unit of data
  - analogous to a row in a RDBMS
- An ordered set of fields (keys) with associated values stored in BSON format
  - similar to JSON
What is BSON?

• “...a binary-encoded serialization of JSON-like documents.”

  BSON Web Site, http://www.bsonspec.org/

• Binary JSON

• Designed to be lightweight, traversable, and efficient
What is a Collection?

• A group of documents
• analogous to a table in a RDBMS
• Schema-less
Advantages of Documents

• Documents correspond to native data types in many programming languages
• Embedded documents and arrays reduce the need for expensive joins
• Dynamic schema support fluent polymorphism
{ 
lastName: "Redlich",
firstName: "Michael",
email: "mike@redlich.net",
role: {
  officer: "President",
  sig: "Java Users Group"
}
}
Field Names

- Strings
- Cannot contain:
  - null
  - dots (.)
  - dollar sign ($)
- No duplicate field names
Conventions Used in This Presentation

- Command Prompt ($)
- MySQL prompt (mysql>)
- MongoDB prompt (>)
- Keywords (db, find(), etc.)
- Variables (variable)
Example Database

- ACGNJ Board of Directors:
  - lastName
  - firstName
  - roles (embedded documents)
  - tenure
Getting Started

- Download MongoDB
- Create a default data directory
  - /data/db
  - C:\data\db
- Create your first MongoDB database
Starting MongoDB

- Start an instance of the MongoDB server:
  
  $ mongod

- Start an instance of the MongoDB client (a JavaScript-based shell):
  
  $ mongo
• Show the list of shell commands:
  > help

• Show the list of databases:
  > show dbs

• Show the current database:
  > db
Mongo Shell (2)

• Specify the database to use or create:
  > use database

• Show the collections within the current database:
  > show collections

• Show the users within the database:
  > show users
Mongo Shell (3)

- Show the recent `system.profile` entries:
  
  ```
  > show profile
  ```

- Tab completion

- Command history
Primary Key

- Denoted by a special field, `_id`

- It can be generated:
  - Implicitly:
    - `{ _id : ObjectId(value) }`
  - Explicitly:
    - `{ _id : 2 }, { _id : "MPR" }`
ObjectIDs

- Default type for `_id`
- A 12-byte hexadecimal BSON type:
Live Demo!
Create
Create a Database

• Create a database in MySQL:

  mysql> CREATE DATABASE database;

• Create a database in MongoDB:

  > use database
Create a Collection

• Create a new table in MySQL:

```
mysql> CREATE TABLE table(column datatype,...);
```

• Create a new collection in MongoDB:

```
> db.collection.insert({field:value, .. })
```
Insert Data

• Insert a row in MySQL:
  
  > INSERT INTO table(column, ...)
  VALUES(value, ...);

• Insert a document in MongoDB:
  
  > db.collection.insert({field: value, ..})
Insert Data with Loops

- Insert multiple documents with an array:
  
  ```
  > for(int i = 0; i < j; ++i)
  db.collection.insert({field: array[i]});
  ```

- Insert multiple documents with variable:
  
  ```
  > for(int i = 0; i < j; ++i)
  db.collection.insert({field: i})
  ```
Live Demo!
Query (I)

- Retrieve all rows in MySQL:
  ```
  mysql> SELECT * FROM table;
  ```

- Retrieve all documents in MongoDB:
  ```
  > db.collection.find()
  ```
Query (2)

• Retrieve specified columns in MySQL:
  
  ```sql
  mysql> SELECT column1, column2 FROM table;
  ```

• Retrieve specified fields in MongoDB:
  
  ```javascript
  > db.collection.find({},
  {field1: true, field2: true})
  ```
Query (3)

- Retrieve specific rows in MySQL:
  ```sql
  mysql> SELECT * FROM table WHERE column = value;
  ```

- Retrieve specific documents in MongoDB:
  ```javascript
  > db.collection.find({field: value})
  ```
Query (4)

- Retrieve specific rows in MySQL:

  ```
  mysql> SELECT * FROM table WHERE column = value ORDER BY value ASC;
  ```

- Retrieve specific documents in MongoDB:

  ```
  > db.collection.find({field:value}).sort({field:1})
  ```
Query (5)

• Query for multiple documents (returns a cursor):
  > `db.collection.find()`

• Query for one document (returns a single document):
  > `db.collection.findOne()`
Query Selectors

- Scalar:
  - \$ne, \$mod, \$exists, \$type, \$lt, \$lte, \$gt, \$gte

- Vector:
  - \$in, \$nin, \$all, \$size
Query (6)

• Retrieve specific rows in MySQL:

```sql
mysql> SELECT * FROM table WHERE column != value;
```

• Retrieve specific documents in MongoDB:

```javascript
> db.collection.find({field: {$ne: value}})
```
Query (7)

- Retrieve specific rows in MySQL:
  
  ```sql
  mysql> SELECT * FROM table WHERE column1 = value OR column2 = value;
  ```

- Retrieve specific documents in MongoDB:
  
  ```javascript
  > db.collection.find({$or: [{field:value},{field:value}]})
  ```
Query (8)

> `db.members.aggregate({$project: {
          officer: "$roles.officer"
        }})`

> `db.members.find({tenure: {
          $gt: new ISODate("2014-12-31")
        }})`

> `db.members.find({"roles.officer": {
          $exists: true
        }}).sort({"roles.officer": 1})`
Query (9)

```javascript
> db.members.find({"roles.director":
  {$all:["Director"]}})

> db.members.find({"roles.committee":
  {$in:["Historian","Newsletter"]}})

> db.members.find({roles:{$size:3}})
```
Live Demo!
Update
Update (1)

• Update a row in MySQL:

```
mysql> UPDATE table SET column = value WHERE id = id;
```

• Update a document in a MongoDB:

```
> db.collection.update({_id:value},
  {$set:{field:value}}, {multi:true})
```
Update (2)

• Update a row in MySQL:

```
mysql> UPDATE table SET column1 = value WHERE column2 > value;
```

• Update a document in MongoDB:

```
> db.collection.update({field1: {$gt: value}},{$set:{field2:value}},{multi:true})
```
Update (3)

- Update a document using `findOne()`:

```javascript
> redlich = db.members.findOne({lastName: "Redlich"})

> redlich.roles = [{sig:"Java Users Group"}]

> db.members.update({lastName: "Redlich"}, redlich)
```
Atomic Update

Operators

- **Scalar:**
  - `$inc`, `$set`, `$unset`

- **Vector:**
  - `$push`, `$pop`, `$pull`, `$pushAll`,
    `$pullAll`, `$addToSet`
Update (4)

```
> db.members.update({lastName: "Redlich"}, {$set: {
  "ISODate": "2016-12-31"}})

> db.members.update({"roles.sig"}, {$set:{"roles.sig":"JUG"}})
```
Delete (1)

• Delete all rows in MySQL:
  ```sql
  mysql> DELETE FROM table;
  ```

• Delete all documents in MongoDB:
  ```javascript
  > db.collection.remove()
  ```
• Delete specific rows in MySQL:

```
mysql> DELETE FROM table WHERE column = value;
```

• Delete specific documents in MongoDB:

```
> db.collection.remove({field:value})
```
Delete (2)

• Delete a MySQL database

  ```
  mysql> DROP DATABASE database;
  ```

• Delete a MongoDB database

  ```
  > use database
  > db.dropDatabase()
  ```
Backup/Restore
Export (1)

- Export a collection to a JSON file
- Ensure `mongod` is running

```
$ mongoexport --db database --collection collection --out path/filename.json
```
Export (2)

- Export a collection to a CSV file
- Ensure `mongod` is running
- A list of fields is required

```
$ mongoexport --db database --collection collection --fields field1,field2,... --csv --out path/filename.json
```
Import

• Import a collection from a JSON, CSV, or TSV file

• Ensure `mongod` is running

```bash
$ mongoimport --db database --collection collection < path/filename.json
```
Dump

• Dump a specified MySQL database:

```
$ mysqldump -u root --opt database
> path.filename.sql
```

• Dump all MongoDB databases:

• Ensure `mongod` is not running

```
$ mongodump --dbpath /data/db --out path
```
Live Demo!
Package Components

(1)

• Core Processes
  • **mongod** - core DB process
  • **mongos** - controller & query router (sharding)
  • **mongo** - interactive JavaScript-based shell
• Binary Import and Export
  • mongodump - creates BSON dump files
  • mongorestore - restores BSON dump files
  • bsondump - converts BSON to JSON
  • mongooplog - streams oplog entries
Package Components

(3)

- Data Import and Export
  - `mongoimport` - imports JSON, CSV, or TSV data formats
  - `mongoexport` - exports to JSON, CSV, or TSV data formats
Package Components

(4)

• Diagnostic Tools
  • `mongostat` - captures database operations by type (insert, query, etc.)
  • `mongotop` - tracks read/write activity
  • `mongosniff` - provides tracing/sniffing view into database activity
  • `mongoperf` - performance testing tool
Package Components

(5)

• GridFS
  • `mongoiles` - provides a command-line interaction to a GridFS storage system
MongoDB Resources (1)
MongoDB Resources

(2)

- mongodb.org
- docs.mongodb.org
- mongodb.org/books
- mongodb.com/products/mongodb
- mongodb.com/reference
- bsonspec.org
- education.mongodb.com
Upcoming Events (1)

- Trenton Computer Festival
  - March 14-15, 2014
  - tcf-nj.org

- Emerging Technologies for the Enterprise
  - April 22-23, 2014
  - phillyemergingtech.com
Upcoming Events (2)

Trenton Computer Festival

EMERGING TECHNOLOGIES
FOR THE ENTERPRISE

SOLD OUT
Thanks!

mike@redlich.net
@mpredli
javasig.org