Thinksheet

Mongo v3.6

Basic Commands

Start local database server *bash shell*

```
$ mongod --dbpath data/db
```

Import data *bash shell*

```
$ mongoimport -d <db> -c <col> --drop -f ~/data.json
```

Start mongo client - connect to local server *bash shell*

```
$ mongo
```

Show databases *mongo shell*

```
> show dbs
```

Use database named "mydb" *mongo shell*

```
> use mydb
```

Show collections in current database *mongo shell*

```
> show collections
```

Delete database *mongo shell*

```
> db.dropDatabase();
```

CRUD operations

Creating Documents

Insert one or more new documents into a collection

`.insert(<doc or array of docs> [,opts])`

- *doc:* is an object literal

Example:

Insert an author with name, email, language and active status

```
db.authors.insert({
  name:"Joe", email:"joe@email.com",
  createdAt: ISODate("2017-01-01"),
  books: [{title: "Draft"}, {title: "Memoirs"}]
})
```

See also: insert(), insertOne(), insertMany()

Reading Documents

Find and filter one or more documents from collection.

`.find(query, [,projection])`

- Use *query object* to filter the selection
- Use *projection* to specify the fields to return

Examples:

Find all documents where `email` is `joe@email.com`

```
db.authors.find( {email:"joe@email.com"} )
```

Find document by _id

```
db.authors.find({
  _id: ObjectId("507c35dd8fada716c89d0013")
})
```

Find all authors where name is 'Joe' AND email is 'joe@email.com'

```
db.authors.find({
  name:"Joe",
  email:"joe@email.com",
})
```

Find all authors; return only the name and _id fields (aka projection)

```
db.authors.find( {}, {name: 1})
```

See also: find(), findOne()

Query Selectors

Query selectors provide additional ways to filter data

Find in range

`$gte` - greater than or equal to
`$lte` - less than or equal to

```
db.authors.find({
  createdAt: {
    $gte: ISODate("2017-01-01"),
    $lte: ISODate("2017-12-31")
  }
})
```

Find authors with one of several names

```
db.authors.find({
  name: { $in: ["Joe", "Chris", "Tauhida"] }
})
```

See also: $eq, $gt, $lt, $in, $and, $exists, $elemMatch

Query Results

Database queries return a cursor with zero or more results. You can do things like sort, limit, and count.

Examples:

Get the 10 most recent authors

```
db.authors.find().
  sort( {createdAt: -1} ).
  limit(10)
```

See also: sort(), limit(), min(), max(), skip(), count()

Updating documents

Update one or more documents matching query

`.update( <query>, update: {}, options: {})`

- *query*: filter determining which documents to update
- *update*: operations to apply to update
- *options*: { upsert: boolean, multi: boolean}
  - *upsert*: if true, then will create a document if no match found
  - multi: if true, updates 1 or more documents

Example:

Update email of user with a string of the user's id

```
db.authors.update({
    _id: ObjectId("507c35dd8fada716c89d0013")
  },{
    email: "newaddress@email.com"
});
```

Update ONE document matching query

`.findAndModify({query, sort, update, new, upsert})`

- *sort:* if multiple document found, sort to determine which is updated
- *update*: operations to apply to update
- new: if true, returns modified document rather than original
- *upsert*: if true, then will create a document if no match found

Example:

Increment user's score by one

```
db.authors.findAndModify({
  query: { name: "Joe" },
  sort: { rating: 1 },
  update: { $inc: { score: 1 } },
  upsert: true
});
```

Update a subdocument in an array using the positional operator

- *$ operator*: identifies an element in an array to update

```
db.authors.update({
  "books.title" : "Draft"
},{
  $set: {
    "books.$.title" : "My Autobiography"
  }
})
```

Compare `.update()` versus `.findAndModify()`

	`.update()`	`.findAndModify()`
Update multiple docs	Yes. Use `multi: true`	Only one doc
Supports sorting	No	Yes
Returns modified doc	No - returns WriteResult	Yes. Use `new: true`

See also: update(), updateOne(), updateMany(), findOneAndUpdate(), findAndModify(), findOneAndReplace(), replaceOne(),

Deleting Documents

Permanently delete one or more documents from a collection

`.deleteOne( <query>)`
`.deleteMany( <query> )`

- *query*: filter determining which documents to update

Examples:

Delete a single document by id

```
db.authors.deleteOne({
  _id: ObjectId("5964243c0a607805d078a9a3")
})
```

Deleting where email is empty

```
db.authors.deleteMany({
  email: {$exists: false}
})
```

See also: deleteOne(), deleteMany(), remove(), findOneAndDelete()