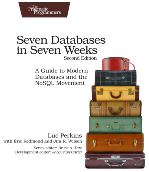
6.2: MongoDB Commands



Instructor: Dr. GP Saggese - gsaggese@umd.edu

- **References**:
  - All concepts in slides
  - MongoDB tutorial
  - Web
    - https://www.mongodb.com/
    - Official docs
    - pymongo
  - Seven Databases in Seven Weeks, 2e

CRUD Operations

- CRUD = Create, Read, Update, Delete
- **Create**

  ```
  db.collection.insert(<document>)
  db.collection.update(<query>, <update>, {upsert: true})
  ```

  - Upsert = update (if exists) or insert (if it doesn't)
- **Read**

  ```
  db.collection.find(<query>, <projection>)
  db.collection.findOne(<query>, <projection>)
  ```

- **Update**

  ```
  db.collection.update(<query>, <update>, <options>)
  ```

- **Delete**

  ```
  db.collection.remove(<query>, <justOne>)
  ```

- Details here

SCIENCE ACADEMY

Create Operations

- `db.collection` specifies the collection (like an SQL table) to store the document

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

  - Without `_id` field, MongoDB generates a unique key

    ```
    db.parts.insert({type: "screwdriver", quantity: 15})
    ```

  - Use `_id` field if it has a special meaning

    ```
    db.parts.insert({\_id: 10, type: "hammer", quantity: 1})
    ```

- Update 1 or more records in a collection satisfying **query**

  ```
  db.collection.update(<query>, <update>, {upsert: true})
  ```

- Update an existing record or create a new record

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

- A more modern OOP-like syntax than the COBOL / FORTRAN-inspired SQL

SCIENCE ACADEMY

Read Operations

- `find` provides functionality similar to `SQL SELECT` command

  `db.collection.find(<query>, <projection>).cursor`

  with:

  - = WHERE condition
  - = fields in result set

- `db.parts.find({parts: "hammer"}).limit(5)`

  - Return cursor to handle a result set
  - Can modify the query to impose limits, skips, and sort orders
  - Can specify to return the 'top' number of records from the result set

- `db.collection.findOne(<query>, <projection>)`

SCIENCE ACADEMY

More Query Examples

- Mongo has a functional programming flavor
  - E.g., composing operators, like $or

SQL	Mongo
`SELECT * FROM users WHERE age>33`	`db.users.find({age: {$gt: 33}})`
`SELECT * FROM users WHERE age!=33`	`db.users.find({age: {$ne: 33}})`
`SELECT * FROM users WHERE name LIKE "%Joe%"`	`db.users.find({name: /Joe/})`
`SELECT * FROM users WHERE a=1 and b='q'`	`db.users.find({a: 1, b: 'q'})`
`SELECT * FROM users WHERE a=1 or b=2`	`db.users.find({$or: [{a: 1}, {b: 2}]})`
`SELECT * FROM foo WHERE name='bob' and (a=1 or b=2 )`	`db.foo.find({name: "bob", $or: [{a: 1}, {b: 2}]})`
`SELECT * FROM users WHERE age>33 AND age<=40`	`db.users.find({'age': {$gt: 33, $lte: 40}})`

Query Operators

Command	Description
$regex	Match by any PCRE-compliant regular expression string (or just use the // delimiters as shown earlier)
$ne	Not equal to
$lt	Less than
$lte	Less than or equal to
$gt	Greater than
$gte	Greater than or equal to
$exists	Check for the existence of a field
$all	Match all elements in an array
$in	Match any elements in an array
$nin	Does not match any elements in an array
$elemMatch	Match all fields in an array of nested documents
$or	or
$nor	Not or
$size	Match array of given size
$mod	Modulus
$type	Match if field is a given datatype
$not	Negate the given operator check

SCIENCE
ACADEMY

Update Operations

- `db.collection.insert(<document>)`
  - Omit the `_id` field to have MongoDB generate a unique key
    ```
    db.parts.insert({type: "screwdriver", quantity: 15})
    db.parts.insert({\_id: 10, type: "hammer", quantity: 1})
    ```
- `db.collection.save(<document>)`
  - Updates an existing record or creates a new record
- `db.collection.update(<query>, <update>, {upsert: true})`
  - Will update 1 or more records in a collection satisfying query
- `db.collection.findAndModify(<query>, <sort>, <update>, <new>, <fields>, <upsert>)`
  - Modify existing record(s)
  - Retrieve old or new version of the record

Delete Operations

- `db.collection.remove(<query>, <justone>)`
  - Delete all records from a collection or matching a criterion
  - `<justone>` specifies to delete only 1 record matching the criterion
- Remove all records in `parts` with `type` starting with h
  - `db.parts.remove(type: /h/ )`
- Delete all documents in the `parts` collection
  - `db.parts.remove()`

SCIENCE ACADEMY

MongoDB Features

- Document-oriented NoSQL store
- Rich querying
    - Full index support (primary and secondary)
- Fast in-place updates
- Agile and scalable
    - Replication and high availability
    - Auto-sharding
    - Map-reduce functionality
- Scale horizontally over commodity hardware
    - Horizontally = add more machines
    - Commodity hardware = relatively inexpensive servers

MongoDB vs Relational DBs

- Keep the functionality that works well in RDBMSs
  - Ad-hoc queries
  - Fully featured indexes
  - Secondary indexes
- Do not offer RDBMS functionalities that don't scale up
  - Long running multi-row transactions
  - ACID consistency
  - Joins

MongoDB Tutorial

Tutorial is at GitHub The instructions are here:

```
> cd $GIT\_REPO/tutorials/tutorial\_mongodb

> vi tutorial\_mongo.md
```