### **Windows**

cd c:/"Program Files"/MongoDB/Server/3.6/bin/

1. **Start MongoDB Server:** Navigate to your Mongo folder and execute the **mongod.exe** file.

**.**/mongod // in gitbash

**Ctrl + c // turn off (make sure to exit terminal when turning off terminal)**

1. **Open Mongo Terminal:** Navigate to your Mongo folder and execute the **mongo.exe** file.

|  |  |  |
| --- | --- | --- |
| Database Type: | **SQL** | **Mongo** |
| **Database** | Schema | Database (db) |
| **Collection of related records** | Tables | Collections |
| **Each one record in the collection of records** | Row / Record | Document |

**MySQL Database Schema == MongoDB Database (db)**

|  |  |
| --- | --- |
| **Show all databases** available on our current MongoDB server | Example: **show dbs** |
| **Show current database** selected | Example: **db** |
| **Change to another database** Note: If the database you're trying to switch to does not exist,  Mongo shell will create a new database and switch to it. | Pattern: **use DB\_NAME**  Example: **use message\_board\_db** |
| **Delete database** Note: db.dropDatabase() will delete the current database in use. | Example: **use message\_board\_db** **db.dropDatabase()** |

**SQL: Tables == MongoDB: Collections**

|  |  |
| --- | --- |
| **View all** collections in a MongoDB | Example: **show collections** |
| **Create** a new collection in the current database | Pattern: **db.createCollection("COLLECTION\_NAME")**  Example: **db.createCollection("ninjas")** |
| **Destroy** a collection | Pattern: **db.COLLECTION\_NAME.drop()**  Example: **db.ninjas.drop()** |

### **CRUD operations**

### CREATE - Inserting a document into a collection:

// Pattern:

db.COLLECTION\_NAME.insert({YOUR\_JSON\_DOCUMENT})

// Example:

db.ninjas.insert({name: "Trey", belt: "black", status: "awesome"})

### READ - Retrieving documents from a collection:

To retrieve documents from your collections, you will use the following method.

// Pattern:

db.COLLECTION\_NAME.find({YOUR\_QUERY\_DOCUMENT})

### DESTROY - Removing documents from a collection:

To remove an item from the database, we would run the following syntax:

// Pattern:

db.COLLECTION\_NAME.remove({YOUR\_QUERY\_DOCUMENT}, BOOLEAN)

// Example

db.ninjas.remove({belt: "yellow"})

db.ninjas.remove({belt: "yellow"}, false) // this query would have the same effect as the one above.

### UPDATE - Updating documents in a collection:

The update syntax can be tricky if you do not see it done correctly first. Let's say I wanted to add a location to the ninja whose name is Trey. I want to make his location equal to Mountain View. I would try to the following:

// Example:

db.ninjas.update({name: "Trey"}, {location: "Mountain View"})

# **Operators**

If I wanted to get all the Dojos whose **number of students is greater than 15**, I would run the following:

db.dojos.find({number\_of\_students: {$gt: 15}})

**Here is a chart of the most frequently-used operators (take some time to play around with them):**

| **name** | **description** |
| --- | --- |
| $gt (greater than) | Use to query selectively on numerical-valued fields |
| $gte (greater than or equal to) | Use to query selectively on numerical-valued fields |
| $lt (less than) | Use to query selectively on numerical-valued fields |
| $lte (less than or equal to) | Use to query selectively on numerical-valued fields |
| $in (in array) | Use to find documents who have a particular value within an array. |

To add the interest 'snowboarding' to the student document, we would run...

db.students.update({\_id: ObjectId("5463d871a6a96d5ed6252f4d")}, {$push: {interests: 'snowboarding'}})

Here are a couple common **array update operators** to get familiar with:

| **Name** | **Description** |
| --- | --- |
| **$push** | Push to an array contained within a document. |
| **$pop** | Removes either the first or last element from an array. EX:  db.COLLECTION.update({QUERY}, {$pop: {array\_key: (1 or -1)}})    Use 1 for the last item in the array, -1 for the first item. |
| **$addToSet** | It functions just like **$push**.  However, **$addToSet** only adds to the specified array if the value doesn't already exist (thereby preventing duplicate entries). |
| **$pull** | Removes a specified value from an array, unlike **$pop**, which removes by location. Ex:  db.COLLECTION.update({QUERY}, {$pull: {array\_key: VALUE}})    This will remove all instances of VALUE from the documents with the array specified by the array\_key that match QUERY. |