**MONGODB**

[**https://www.mikedane.com/databases/mongodb/**](https://www.mikedane.com/databases/mongodb/)

1. **WINDOWS INSTALLATION**

**Instructions**

1. Download the Windows installer
   * <https://www.mongodb.com/download-center?jmp=nav#community>
2. Step through installer (be sure to download compass)
3. Add mongod.exe to path
4. Create ‘/data/db’ directory
   * Create this at root directory of hard-drive
     + Ensure that current user has read/write permissions
   * Or customize location using:
     + mongod –dbpath
     + I put this at the root directory of my home folder
5. Start MongoDB server
   * mongod
6. Connect to MongoDB server from client – (in new terminal window) mongo –host localhost:27017 – (in compass) use the default settings
7. **CREATING/ DROPING COLLECTIONS**

// if you insert into a non-existant collection it will be created automatically

db.createCollection("students")

// https://docs.mongodb.com/manual/reference/method/db.createCollection/index.html

db.createCollection("students", {options})

db.dropCollection("students")

show collections

1. **INSERTING DOCUMENTS**

**Data Types**

{

string: "String of text",

int: 405,

double: 3.565,

boolean: true, // false

array: [1, 2, 3],

object: {attr1: "attr1", attr2: "attr2"},

date: new Date("<YYYY-mm-dd>"),

object\_id: <ObjectId>,

no\_value: null

}

**Additional Data Types**

Timestamp

Binary data

Regular expressions

JS Code

**// Inserting Documents**

db.students.insertOne({name: "Jack", major: "Biology", gpa: 3.5})

db.students.insertOne({name: "Claire", major: "Marketing", gpa: 3.7, awards: ["Valedictorian", "Summa Cum Laude"]} )

db.students.insertOne({name: "Evan", major: "Astronomy", gpa: 3.7, grades: [90, 88, 95, 78] } )

db.students.insertOne({name: "Kate", major: "Sociology", gpa: 3.2, contact: {phone: "333-3333", email: "student@school.edu"}})

db.students.insertOne({name: "Phil", major: "Chemistry", gpa: 2.5, startdate: new Date("2012-08-23")})

db.students.insertOne({\_id: 4, name: "John", major: "Biology", gpa: 3.2})

db.students.insertMany([

{name: "Mike", major: "Computer Science", gpa: 2.7},

{name: "Andrea", major: "Math", gpa: 4.0, awards: ["Summa Cum Laude"]}

])

1. **FINDING DOCUMENTS**

**// Find all students**

db.students.find( {} )

db.students.find( {} ,{\_id:0})

**// Find the first 3 students**

db.stuents.find( {} ).limit(3)

**// Find all students and sort by name in ascending order**

db.students.find( {} ).sort( {name: 1} )

db.students.find( {} ,{\_id:0}).sort( {name: 1} )

**// Find all students and sort by multiple fields (gpa, name) in ascending order**

**db.students.find( {} ,{\_id:0}).sort( {gpa: -1, name: 1} )**

1 => sort in ascending order, -1 => sort in descending order

**Filtering**

**// Find all biology majors**

**db.students.find( {major: "Biology"} )**

**db.students.find( {major: "Biology"},{\_id:0} )**

**db.students.find( {major: "Biology", name: “Jack”},{\_id:0} *)*** *this give us the AND logic. Name = Jack and Major = Biology.*

**// Find all student's with a phone number 333-3333**

**db.students.find( {contact: {phone: "333-3333", email: "student@school.edu"} } )**

**// Find all biology majors named Jack**

**db.students.find( {name: "Jack", major: "Biology"} )**

**// Final all students who are chemistry majors or named Jack** *--the OR logic*

**db.students.find( { $or: [ {name: "Jack"}, {major: "Chemistry"} ] } ,{\_id:0})**

**// Final all students with a gpa above 3.5** *– comparison operators*

**db.students.find( {gpa: {$gt: 3.5} } ,{\_id:0})**

**// Find all students with a gpa less than or equal to 3.2**

**db.students.find( {gpa: {$lte: 3.2} } ,{\_id:0}).sort({gpa: -1})**

**$eq 🡺 equal to**

**$ne 🡺 not equal to**

**$lt 🡺 less than**

**$lte 🡺 less than or equal to**

**$gt 🡺 greater than**

**$gte 🡺 greater than or equal to**

**// Find all students with names in the array**

**db.students.find( {name: {$in: ["Kate", "Claire"]} },{\_id:0} )**

**// $in, $nin =>not in**

**// Find all students who have awards –** *check if records exist for a certain entry/document*

**db.students.find( {awards: {$exists: true} } ,{\_id:0})**

**// false => checks if missing**

**// Find all db entries where the name is a string –** *handling data types*

**// Type list - https://docs.mongodb.com/manual/reference/bson-types/**

**db.students.find({name: {$type: 2} })**

*Tackling arrays*

**// Find all students who's first grade is a 90**

**db.students.find( {"grades.0": 90 } )**

**// Find all students who have a grade greater than 80**

**db.students.find( {grades: {$elemMatch: { $gte: 80} } } )**

**// Find all students who have 4 grades recorded**

**db.students.find( {grades: {$size: 4 } } )**

1. **UPDATE/ DELETE DOCUMENTS**

**// same filters as inserting**

**db.stuents.updateOne(<filter>, <update>, <options>)**

**// Do this twice so we can change it back with updateMany**

**db.students.updateOne(**

**{major: "Biology"},**

**{**

**$set:**

**{major: "Bio"}**

**}**

**)**

**db.students.updateMany(**

**{major: "Bio"},**

**{**

**$set:**

**{major: "Biology"}**

**}**

**)**

**// replaceMany()**

**db.students.replaceOne(**

**{major: "Bio"},**

**{name: "new name", major: "new major", gpa: 4.0}**

**)**

**// Delete all documents**

**db.students.deleteMany({})**

**db.students.deleteOne({major: "Biology"})**

**db.students.deleteMany({gpa: {$gte: 3.5}})**

1. **BULK ACTIONS**

**db.students.bulkWrite(**

**[**

**{ insertOne :**

**{**

**"document" :**

**{**

**name: "Andrew", major: "Architecture", gpa: 3.2**

**}**

**}**

**},**

**{ insertOne :**

**{**

**"document" :**

**{**

**name: "Terry", major: "Math", gpa: 3.8**

**}**

**}**

**},**

**{ updateOne :**

**{**

**filter : { name : "Terry" },**

**update : { $set : { gpa : 4.0 } }**

**}**

**},**

**{ deleteOne :**

**{ filter : { name : "Kate"} }**

**},**

**{ replaceOne :**

**{**

**filter : { name : "Claire" },**

**replacement : { name: "Genny", major: "Counsling", gpa: 2.4 }**

**}**

**}**

**],**

**{ordered: false}**

**);**

1. **TEXT INDEXING**

**db.stores.insertMany(**

**[**

**{ \_id: 1, name: "Java Hut", description: "Coffee and cakes" },**

**{ \_id: 2, name: "Burger Buns", description: "Gourmet hamburgers" },**

**{ \_id: 3, name: "Coffee Shop", description: "Just coffee" },**

**{ \_id: 4, name: "Clothes Clothes Clothes", description: "Discount clothing" },**

**{ \_id: 5, name: "Java Shopping", description: "Indonesian goods" }**

**]**

**)**

**db.stores.createIndex( { name: "text", description: "text" } )**

**db.stores.find({ $text: {$search: "Coffee" } })**

**db.stores.find({ $text: {$search: "Java Hut Coffee" } })**

**db.stores.find(**

**{ $text: { $search: "java hut coffee" } },**

**{ score: { $meta: "textScore" } }**

**).sort( { score: { $meta: "textScore" } } ) --** *score ranks your results based on how closely they are to the search term—it sorts based on relevance*

1. **AGGREGATION**

Aggregation allows us to process data records and return computed results.

**Create a new database collection and populate it:**

**db.purchase\_orders.insertMany(**

**[**

**{product: "toothbrush", total: 4.75, customer: "Mike"},**

**{product: "guitar", total: 199.99, customer: "Tom"},**

**{product: "milk", total: 11.33, customer: "Mike"},**

**{product: "pizza", total: 8.50, customer: "Karen"},**

**{product: "toothbrush", total: 4.75, customer: "Karen"},**

**{product: "pizza", total: 4.75, customer: "Dave"}**

**{product: "toothbrush", total: 4.75, customer: "Mike"},**

**]**

**)**

**// find out how many toothbrushes were sold**

**db.purchase\_orders.count({product: "toothbrush"})**

**// Find list of all products sold**

**db.purchase\_orders.distinct("product")**

**// Find the total amount of money spent by each customer**

**db.purchase\_orders.aggregate(**

**[**

**{$match: {} },** *-- filters different entries in the collection*

**{$group: {\_id: "$customer", total: { $sum: "$total"} } } –** *define the structure of the info you want to get back*

**]**

**)**

**// Find how much has been spent on each product and sort it by price**

**db.purchase\_orders.aggregate(**

**[**

**{$match: {} },**

**{$group: {\_id: "$product", total: { $sum: "$total"} } },**

**{$sort: {total: -1}}**

**]**

**)**

**// Find how much money each customer has spent on toothbrushes and pizza**

**db.purchase\_orders.aggregate(**

**[**

**{$match: {product: {$in: ["toothbrush", "pizza"]} } },**

**{$group: {\_id: "$product", total: { $sum: "$total"} } },**

**]**

**)**

**// https://docs.mongodb.com/manual/reference/operator/aggregation/**

**// Show the list of all pipeline operators**