# MongoDB – Query & Data Modeling

# Query Selectors - Comparison

```
{field: {operator: value} }
```

- Can be applied on numeric and string field values
- **$eq** equal to

```
{ field: { $eq: value } }
{ field: value }
```

- **$gt** greater than
- **$gte** greater than or equal to
- **$lt** less than
- **$lte** less than or equal to
- **$ne** not equal to
- **$in** matches any of the values specified in an array
- **$nin** matches none of the values specified in an array.

https://docs.mongodb.com/manual/reference/operator/query/

# Examples - Comparison Query Operators

```
// return all documents that the score property is greater than 85
db.col.find({score: {$gt: 85}})
// return all documents that the name property starts with A, B and C only
db.col.find({name: {$lt: "D"}})
// return all documents that the name property which the first letter starts between B
and D(implicit AND)
db.col.find({name: {$gt: "B", $lt: "D"}})
// return all documents where the qty field value is either 5 or 15
   { _id: 1, qty : 3 }
   { _id: 2, qty : 5 }
db.col.find({ qty: { $in: [ 5, 15 ] } } ) // returns _id: 2
// return all documents where courses field value is either CS472 or CS572
   { _id: 1, courses: [ "CS472", "CS572", "CS477" ] }  (implicit OR)
db.col.find({ courses: { $in: ["CS572", "CS472"] } } )
```

- **Notes:** Because different values types for the same field is possible, MongoDB will do strongly/dynamically typed comparison operations.

# Examples - $regex

- Provides regular expression capabilities for pattern matching strings in queries.

```
{ field: { $regex: 'pattern', $options: '<options>' } }
```

```
// return all documents where the name field has the letter "a"
   anywhere in the value
db.col.find( { name: { $regex: "a" } } )

// return all documents where the name field values end with letter "e"
   upper and lower cases (i for case insensitivity)
db.col.find( { name: { $regex: "e$", $options: "i" } } )
```

# Query Selectors - Logical

- Joins query clauses with a logical operation:
- $or returns all documents that match the conditions of either clause.
- $and returns all documents that match the conditions of both clauses.
- $not returns documents that do not match the query expression.
- $nor returns all documents that fail to match both clauses.

```
{ $or:  [ { <expression1> }, { <expression2> }, ... , { <expressionN> } ] }
{ $and: [ { <expression1> }, { <expression2> }, ... , { <expressionN> } ] }

// return all documents where either the quantity field value is
   less than 20 or the price field value equals 10:
db.col.find( { $or: [ { quantity: { $lt: 20 } }, { price: 10 } ] } )
```

# Searching an Array

```
// { _id: 1, courses: [ "CS472", "CS572", "CS477" ] }

// find all documents where courses value contains "CS472" or "CS572"
db.col.find({ courses: { $in: ["CS572", "CS472"] } })
db.col.find({ $or: [ { courses: "CS572" }, { courses: "CS472" } ] })

// find all documents where courses value contains "CS472" and "CS572"
db.col.find({ courses: { $all: [ "CS572" , "CS472" ] } })
db.col.find({ $and: [ { courses: "CS572" }, { courses: "CS472" } ] })
```

- The **`$in`** operator selects the documents where the value of a field is an array that contains at any order any (at least one) of the specified elements
- The **`$all`** operator selects the documents where the value of a field is an array that contains at any order all the specified elements

# Searching an Object

```
{_id: 1, email: { work: "work@mum.edu",
                  personal: "personal@gmail.com"} }

// nothing will be returned
db.col.find({ email: { work: "work@mum.edu" } })
db.col.find({ email: { personal: "personal@gmail.com" } })
db.col.find({ email: { personal: "personal@gmail.com",
                       work: "work@mum.edu"} })

// will work
db.col.find({ email: { work: "work@mum.edu",
                       personal: "personal@gmail.com"} })

// how to search for one key only?
db.col.find({ "email.work": "work@mum.edu" })
db.col.find({ "email.personal": "personal@gmail.com" })
```

# Field Update Operators

- By default `update()` will: update a **single document** and **replace** everything but the `_id`

```
// Original document
{ "_id" : "MUM", "students" : 250, "courses" : ["CS572", "CS477"] }

db.col.update({_id: "MUM"}, {"students":500})

// Results
{ "_id" : "MUM", "students" : 500 }

// To target only one field use { $set: { field1: value1, ... } }
// If the field does not exist, $set will add a new field with the
   specified value
{ "_id" : "MUM", "students" : 250, "courses" : ["CS572", "CS477"] }

db.col.update({_id:"MUM"}, {$set: {"students":500, "entry":"Oct"} })

// Results
{ "_id" : "MUM", "students" : 500, "courses" : ["CS572", "CS477"] , "entry":"Oct" }
```

# Field Update Operators

```
// Original document
//If set to true, creates a new document when no document matches the query criteria. The default value is false.
{ "_id" : "MUM", "students" : 250, "courses" : ["CS572", "CS477"] }
db.col.update({_id:"MUM University"}, {"students":500}, {upsert: true})
// Results
{ "_id" : "MUM", "students" : 250, "courses" : ["CS572", "CS477"] }
{ "_id" : "MUM University", "students" : 500 }

// update all docs to have one more field (city: Fairfield)
{ "_id" : "1", "Dept" : "CompPro" }
{ "_id" : "2", "Dept" : "SustainableLiving" }
db.col.update({}, {$set: {"city":"Fairfield"}} , {multi: true})
// Results
{ "_id" : "1", "Dept" : "CompPro", "city":"Fairfield" }
{ "_id" : "2", "Dept" : "SustainableLiving", "city":"Fairfield" }
```

# Array Update Operators

```
//Original Document { "_id" : 1, "a" : [1, 2, 3, 4] }
db.testCol.update({_id:1}, { $set { "a.2":5 } })
// output: { "_id" : 1, "a" : [1, 2, 5, 4] }
db.col.update({_id:1}, { $push { "a": 6 } })
// output: { "_id" : 1, "a" : [1, 2, 5, 4, 6] }
db.col.update({_id:1}, { $pop { "a": 1 } })
// output: { "_id" : 1, "a" : [1, 2, 5, 4] }
db.col.update({_id:1}, { $pop { "a": -1 } })
// output: { "_id" : 1, "a" : [2, 5, 4] }
db.col.update({_id:1}, { $push:{$each: { "a": [7, 8, 9] } }})
// output: { "_id" : 1, "a" : [2, 5, 4, 7, 8, 9] }
db.col.update({_id:1}, { $pull: { "a": [5] } })
// output: { "_id" : 1, "a" : [2, 4, 7, 8, 9] }
db.col.update({_id:1}, { $pullAll: { "a": [2, 4, 8] } })
// output: { "_id" : 1, "a" : [7, 9] }
db.col.update({_id:1}, { $addToSet: { "a": 5 } })
// output: { "_id" : 1, "a" : [7, 9, 5] }
db.col.update({_id:1}, { $addToSet: { "a": 5 } })
// output: { "_id" : 1, "a" : [7, 9, 5] }
```

# Delete documents db.collection.remove()

```
// delete all documents - One by One
db.col.remove({})

// delete all students whose names start with N-Z
db.col.remove({"student": {$gt: "M"}})

// drop the collection - Faster than remove()
db.col.drop()
```

**Notes**

- When we want to delete large number of documents, it's faster to use drop() but we will need to create the collection again and create all indexes as drop() will take the indexes away (while remove() will keep them)
- Multi-docs remove are not atomic isolated transactions to other R/Ws and it will yield in between.
- Each single document is atomic, no other R/Ws will see a half removed document.

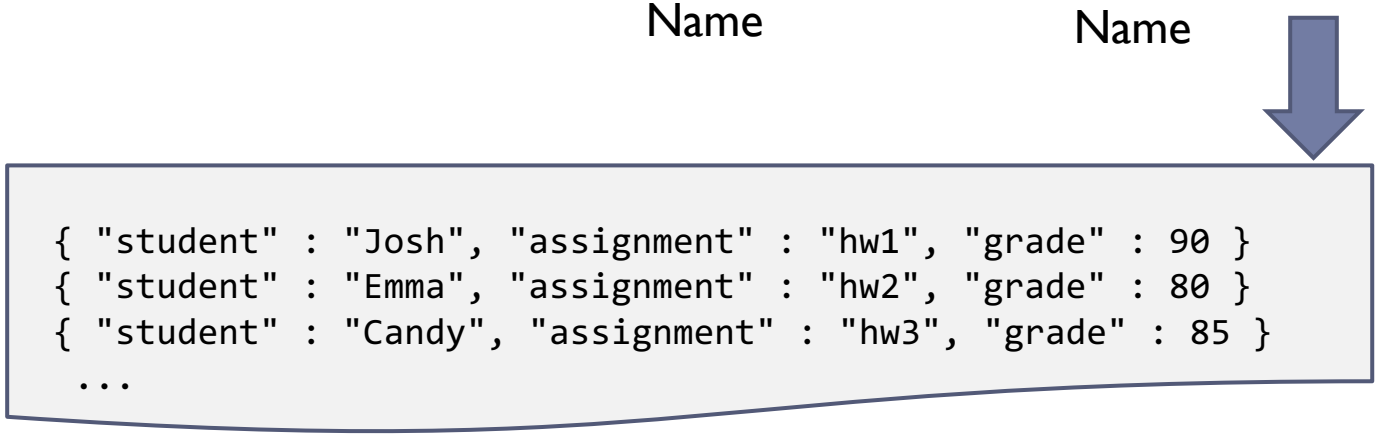
# Import/Export JSON in MongoDB

```
mongoexport --collection=events --db=reporting --out=events.json
```

Collection Name

Database Name

JSON File



```
{ "student" : "Josh", "assignment" : "hw1", "grade" : 90 }
{ "student" : "Emma", "assignment" : "hw2", "grade" : 80 }
{ "student" : "Candy", "assignment" : "hw3", "grade" : 85 }
 ...
```

**Note**: `mongoimport` and `mongoexport` might not work very well with rich documents.
Instead you may use `mongodump –db dbName` and `mongorestore` for full support of rich documents.

```
mongoexport --collection=events --db=reporting --o=file.json
```

Collection Name

Database Name

JSON Output File

# Resources

- Data Model
  - https://docs.mongodb.com/manual/core/data-modeling-introduction/
  - https://www.studytonight.com/mongodb/data-modelling-in-mongodb
  - https://severalnines.com/database-blog/how-use-mongodb-data-modeling-improve-throughput-operations
- CRUD Operations
  - https://docs.mongodb.com/manual/crud/
- Other Resources
  - Multiple collections vs Embedded documents