

Lecture 5: MongoDB Basic Queries

Big Data System Design

In Last Lecture

❖ MongoDB overview

- Database, collection, document, field, embedded documents

❖ Database

- Commands
 - use DATABASE_NAME
 - show dbs
 - db.dropDatabase()
- Don't forget that you must insert at least a document for database to be displayed

In Last Lecture

❖ Collection

- Collection is a group of MongoDB documents similar to tables of RDBMS
- Commands
 - `db.createCollection(name, options)`
 - `show collections`
 - `db.COLLECTION_NAME.drop()`
- Capped collections
 - A fixed size collection that automatically overwrites its oldest entries when it reaches its maximum size

In Last Lecture

❖ Document

- A document is a set of key-value pairs
- MongoDB provides the following three methods for inserting documents into a database:
 - `db.COLLECTION_NAME.insertOne(document)`
 - `db.COLLECTION_NAME.insertMany([documents])`

Table of Contents

❖ Part 1.

- Query Document

❖ Part 2.

- Update Document

❖ Part 3.

- Delete Document

Part 1

QUERY DOCUMENT

Query Document

❖ Preparation


- Insert the following documents into the inventory collection
 - use DATABASE_NAME
 - `db.inventory.insertMany([`
 `{item: "journal", qty: 25, size: {h: 14, w: 21, uom: "cm"}, status: "A"},`
 `{item: "notebook", qty: 50, size: {h: 8.5, w: 11, uom: "in"}, status: "A"},`
 `{item: "paper", qty: 100, size: {h: 8.5, w: 11, uom: "in"}, status: "D"},`
 `{item: "planner", qty: 75, size: {h: 22.85, w: 30, uom: "cm"}, status: "A"}`
 `{item: "paper", qty: 45, size: {h: 10, w: 15.25, uom: "cm"}, status: "A"}`
 `]);`



Query Document





❖ Preparation







practice > inventory

Documents 5 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#) 

Explain Reset Find  Options 

 ADD DATA  EXPORT DATA  UPDATE  DELETE

1 - 5 of 5      

```
{
  "_id": ObjectId('6603cee6d34e6e5f7e140b72'),
  "item": "journal",
  "qty": 25,
  "size": Object,
  "status": "A"
}
```

```
{
  "_id": ObjectId('6603cee6d34e6e5f7e140b73'),
  "item": "notebook",
  "qty": 50,
  "size": Object,
  "status": "A"
}
```

```
{
  "_id": ObjectId('6603cee6d34e6e5f7e140b74'),
  "item": "paper",
  "qty": 100,
  "size": Object,
  "status": "D"
}
```

```
{
  "_id": ObjectId('6603cee6d34e6e5f7e140b75'),
  "item": "planner",
  "qty": 75,
  "size": Object,
  "status": "A"
}
```

```
{
  "_id": ObjectId('6603cee6d34e6e5f7e140b76'),
  "item": "paper",
  "qty": 45,
  "size": Object,
  "status": "A"
}
```


Query Document

❖ find() function

- To query documents from the target collection
- Syntax
 - `db.COLLECTION_NAME.find(query, projection, options)`

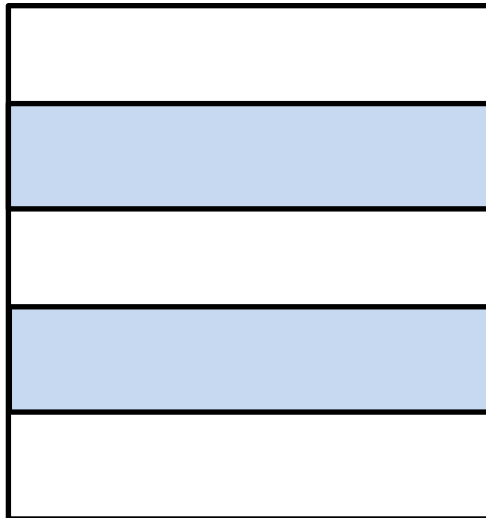
Parameter	Type	Description
query	document	Optional. Specifies selection filter using query operators . To return all documents in a collection, omit this parameter or pass an empty document (<code>{}</code>).
projection	document	Optional. Specifies the fields to return in the documents that match the query filter. To return all fields in the matching documents, omit this parameter. For details, see Projection .
options	document	Optional. Specifies additional options for the query. These options modify query behavior and how results are returned. To see available options, see FindOptions . ↗

<https://www.mongodb.com/docs/manual/reference/method/db.collection.find/>

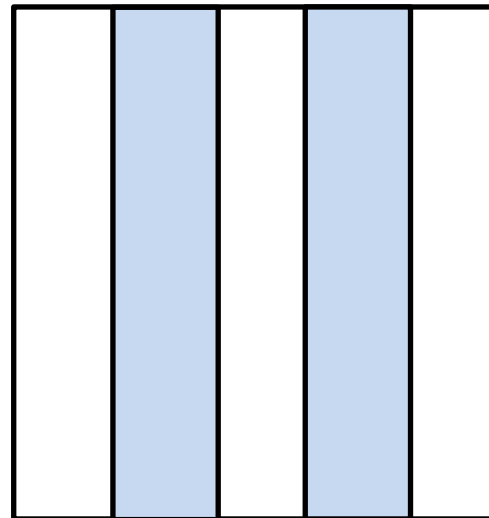
Query Document

❖ Find types

- Selection: documents
- Projection: fields



Selection



Projection

Query Document

- ❖ Select all documents in a collection
 - To query the inventory collection, simply define the following select criteria
 - `db.inventory.find()`
 - This operation corresponds to the following SQL statement
 - `SELECT * FROM inventory`

Query Document

❖ Query

- A document that consists of the key(field)-value conditions
- Example)
 - `SELECT * FROM inventory WHERE qty=25;`
 - `{qty: 25}`
- Operator
 - Comparison
 - Logical
 - Element
 - Evaluation
 - Geospatial
 - Array
 - Bitwise
 - Miscellaneous operators
 - Projection operators

Query Document

- ❖ Query operator
 - Comparison operators

Name	Description
<code>\$eq</code>	Matches values that are equal to a specified value.
<code>\$gt</code>	Matches values that are greater than a specified value.
<code>\$gte</code>	Matches values that are greater than or equal to a specified value.
<code>\$in</code>	Matches any of the values specified in an array.
<code>\$lt</code>	Matches values that are less than a specified value.
<code>\$lte</code>	Matches values that are less than or equal to a specified value.
<code>\$ne</code>	Matches all values that are not equal to a specified value.
<code>\$nin</code>	Matches none of the values specified in an array.

Query Document

❖ Query operator

- Comparison operator: \$eq
 - Matches documents where the value of a field equals the specific value
 - {<field>: {\$eq: <value>}}
 - Select all documents where the value of the qty field equals 100
 - db.inventory.find({qty: {\$eq: 100}})
 - db.inventory.find({qty: 100})

```
> db.inventory.find({qty: {$eq: 100}})
< {
  _id: ObjectId('6603cee6d34e6e5f7e140b74'),
  item: 'paper',
  qty: 100,
  size: {
    h: 8.5,
    w: 11,
    uom: 'in'
  },
  status: 'D'
}
practice>
```

Query Document

❖ Query operator

- Comparison operators: \$gt, \$gte, \$lt, \$lte
 - Select documents where the value of the field is greater, greater or equal, less, less or equal than the specific value
 - {<field>: {\$gt: <value>}}
 - The following example queries the inventory collection to select all documents where the value of the qty field greater than 25
 - db.inventory.find({qty: {\$gt: 25}})
 - db.inventory.find({qty: {\$gte: 25}})
 - These operations correspond to the following SQL statement:
 - SELECT * FROM inventory WHERE qty > 25 (qty >= 25)

Query Document

❖ Query operator

- Comparison operators: \$gt, \$gte, \$lt, \$lte

```
>_MONGOSH
> db.inventory.find({qty: {$gt: 25}})
< {
  _id: ObjectId('6603cee6d34e6e5f7e140b73'),
  item: 'notebook',
  qty: 50,
  size: {
    h: 8.5,
    w: 11,
    uom: 'in'
  },
  status: 'A'
}
{
  _id: ObjectId('6603cee6d34e6e5f7e140b74'),
  item: 'paper',
  qty: 100,
  size: {
    h: 8.5,
    w: 11,
    uom: 'in'
  },
  status: 'D'
}
{
  _id: ObjectId('6603cee6d34e6e5f7e140b75'),
  item: 'planner',
  qty: 75,
  size: {
    h: 22.85,
    w: 30,
    uom: 'cm'
  },
  status: 'A'
}
```


Query Document

❖ Query operator

- Comparison operator: `$ne`
 - Matches all values that are not equal to a specific value
 - `{<field>: {$ne: <value>}}`
 - This query will select all documents in the inventory collection where the qty field value does not equal 25
 - `db.inventory.find({qty: {$ne: 25}})`
 - This operation corresponds to the following SQL statement:
 - `SELECT * FROM inventory WHERE qty != 25`

Query Document

❖ Query operator

- Comparison operator: \$ne

```
>_MONGOSH
> db.inventory.find({qty: {$ne: 25}})
< {
  _id: ObjectId('6603cee6d34e6e5f7e140b73'),
  item: 'notebook',
  qty: 50,
  size: {
    h: 8.5,
    w: 11,
    uom: 'in'
  },
  status: 'A'
}
{
  _id: ObjectId('6603cee6d34e6e5f7e140b74'),
  item: 'paper',
  qty: 100,
  size: {
    h: 8.5,
    w: 11,
    uom: 'in'
  },
  status: 'D'
}
{
  _id: ObjectId('6603cee6d34e6e5f7e140b75'),
  item: 'planner',
  qty: 75,
  size: {
    h: 22.85,
    w: 30,
    uom: 'cm'
  },
  status: 'A'
}
```

Query Document

❖ Query operator

- Comparison operator: \$in

- Selects the documents where the value of a field equals any value in the specific array (shorthand for multiple OR conditions)
 - {<field>: {\$in: [<value 1>, <value 2>, ..., <value N>]}}
- The following example retrieves all documents from the inventory collection where status equals either "A" or "D"
 - db.inventory.find({status: {\$in: ["A", "D"]}})
- This operation corresponds to the following SQL statement:
 - SELECT * FROM inventory WHERE status in ("A", "D")

Query Document

❖ Query operator

- Comparison operator: \$in

```
>_MONGOSH
> db.inventory.find({status: {$in: ["A", "D"]}})
< {
  _id: ObjectId('6603cee6d34e6e5f7e140b72'),
  item: 'journal',
  qty: 25,
  size: {
    h: 14,
    w: 21,
    uom: 'cm'
  },
  status: 'A'
}
{
  _id: ObjectId('6603cee6d34e6e5f7e140b73'),
  item: 'notebook',
  qty: 50,
  size: {
    h: 8.5,
    w: 11,
    uom: 'in'
  },
  status: 'A'
}
{
  _id: ObjectId('6603cee6d34e6e5f7e140b74'),
  item: 'paper',
  qty: 100,
  size: {
    h: 8.5,
    w: 11,
    uom: 'in'
  },
  status: 'D'
}
```

Query Document

❖ Query operator

- Logical operators

Name	Description
<code>\$and</code>	Joins query clauses with a logical <code>AND</code> returns all documents that match the conditions of both clauses.
<code>\$not</code>	Inverts the effect of a query expression and returns documents that do <i>not</i> match the query expression.
<code>\$nor</code>	Joins query clauses with a logical <code>NOR</code> returns all documents that fail to match both clauses.
<code>\$or</code>	Joins query clauses with a logical <code>OR</code> returns all documents that match the conditions of either clause.

Query Document

❖ Query operator

- Logical operator: `$and`
 - Performs a logical AND operation on an array of two or more expressions
 - `{ $and: [{<expression 1>}, {<expression 2>}, ..., {<expression N>}] }`
 - Select the documents that satisfy all the expressions in the array
 - `db.inventory.find({ $and: [{ status: "D" }, { qty: { $lte: 75 } }] })`
 - Simplify above query using comma (,) operator
 - `db.inventory.find({ status: "D", qty: { $lte: 75 } })`
 - This operation corresponds to the following SQL statement:
 - `SELECT * FROM inventory WHERE status = "D" and qty =< 75`

Query Document

❖ Query operator

- Logical operator: \$and

```
>_MONGOSH
```

```
> db.inventory.find({$and: [{status: "D"}, {qty: {$lte: 75}}]})
```

```
<
```

```
practice>
```

Query Document

❖ Query operator

- Logical operator: \$or
 - Perform a logical OR operation on an array of two or more expressions
 - `{ $or: [{<expression 1>}, {<expression 2>}, ..., {<expression N>}] }`
 - Select the documents that satisfy at least one of the expressions in the array
 - `db.inventory.find({ $or: [{ status: "A" }, { qty: { $lt: 30 } }] })`
 - This operation corresponds to the following SQL statement:
 - `SELECT * FROM inventory WHERE status = "A" OR qty < 30`

Query Document

❖ Query operator

- Logical operator: \$or

```
>_MONGOSH
> db.inventory.find({$or: [{status: "A"}, {qty: {$lt: 30}}]})
< {
  _id: ObjectId('6603cee6d34e6e5f7e140b72'),
  item: 'journal',
  qty: 25,
  size: {
    h: 14,
    w: 21,
    uom: 'cm'
  },
  status: 'A'
}
{
  _id: ObjectId('6603cee6d34e6e5f7e140b73'),
  item: 'notebook',
  qty: 50,
  size: {
    h: 8.5,
    w: 11,
    uom: 'in'
  },
  status: 'A'
}
{
  _id: ObjectId('6603cee6d34e6e5f7e140b75'),
  item: 'planner',
  qty: 75,
  size: {
    h: 22.85,
    w: 30,
    uom: 'cm'
  },
  status: 'A'
}
```

Query Document

❖ Query operator

- Logical operator: `$not`
 - Perform a logical NOT operation on the specified `<operator-expression>`
 - `{<field: {$not: {<operator-expression>}}>}`
 - Select the documents that do not match the `<operator-expression>`
 - Select all documents where qty is not greater than 75
 - `db.inventory.find({qty: {$not: {$gt: 75}}})`

Query Document

❖ Query operator

- Logical operator: \$not

```
>_MONGOSH
> db.inventory.find({qty: {$not: {$gt: 75}}})
< {
  _id: ObjectId('6603cee6d34e6e5f7e140b72'),
  item: 'journal',
  qty: 25,
  size: {
    h: 14,
    w: 21,
    uom: 'cm'
  },
  status: 'A'
}
{
  _id: ObjectId('6603cee6d34e6e5f7e140b73'),
  item: 'notebook',
  qty: 50,
  size: {
    h: 8.5,
    w: 11,
    uom: 'in'
  },
  status: 'A'
}
{
  _id: ObjectId('6603cee6d34e6e5f7e140b75'),
  item: 'planner',
  qty: 75,
  size: {
    h: 22.85,
    w: 30,
    uom: 'cm'
  },
  status: 'A'
}
```

Query Document

❖ Query embedded document

- To specify an equality condition on a field that is an embedded document, use the following syntax:
 - {<field>: <value>}
 - <value> is the document to match
- The following query selects all documents where the field size equals the document
 - `db.inventory.find({size: {h: 14, w: 21, uom: "cm"}})`

Query Document

❖ Query embedded document

```
>_MONGOSH
> db.inventory.find({size: {h: 14, w: 21, uom: "cm"}})
< {
  _id: ObjectId('6603cee6d34e6e5f7e140b72'),
  item: 'journal',
  qty: 25,
  size: {
    h: 14,
    w: 21,
    uom: 'cm'
  },
  status: 'A'
}
```

Query Document

❖ Query embedded document

- Equality matches on the whole embedded document require an EXACT match of the specified <value> document including the field order
- The following queries does not match any documents in the inventory collection

- `db.inventory.find({size: {w: 21, h: 14, uom: "cm"}})`

```
>_MONGOSH
> db.inventory.find({size: {w: 21, h: 14, uom: "cm"}})
<
```

- `db.inventory.find({size: {h: 14, w: 21}})`

```
>_MONGOSH
> db.inventory.find({size: {h: 14, w: 21}})
<
```

Query Document

❖ Query embedded document

- To specify a query condition on fields, use dot(.) notation
 - ("field.nestedField")
 - Don't forget quotation

```
>_MONGOSH
> db.inventory.find({size.uom: "in"})
✖ SyntaxError: Unexpected token, expected ",", (1:23)

> 1 | db.inventory.find({size.uom: "in"})
    |                               ^
```

- All documents where the field uom nested in the size field equals "in"
 - db.inventory.find({"size.uom": "in"})

Query Document

❖ Query embedded document

- To specify a query condition on fields, use dot(.) notation
 - ("field.nestedField")

```
>_MONGOSH
> db.inventory.find({"size.uom": "in"})
< {
  _id: ObjectId('6603cee6d34e6e5f7e140b73'),
  item: 'notebook',
  qty: 50,
  size: {
    h: 8.5,
    w: 11,
    uom: 'in'
  },
  status: 'A'
}
{
  _id: ObjectId('6603cee6d34e6e5f7e140b74'),
  item: 'paper',
  qty: 100,
  size: {
    h: 8.5,
    w: 11,
    uom: 'in'
  },
  status: 'D'
}
```


Query Document

❖ Query embedded document

- Try to rewrite the following query using dot notation
 - `db.inventory.find({size: {h: 14, w: 21}})`

▪ Result

- `db.inventory.find({"size.h": 14, "size.w": 21})`

▪ Task

- Select all documents where the nested field `h` is less than 15, the nested field `uom` equals `"in"`, and the `status` field equals `"D"`
 - `db.inventory.find({"size.h": {$lt: 15}, "size.uom": "in", status: "D"})`

```
>_MONGOSH
> db.inventory.find({"size.h": 14, "size.w": 21})
< {
  _id: ObjectId('6603cee6d34e6e5f7e140b72'),
  item: 'journal',
  qty: 25,
  size: {
    h: 14,
    w: 21,
    uom: 'cm'
  },
  status: 'A'
}
```

Query Document

- ❖ Query embedded document
 - Task

```
>_MONGOSH
> db.inventory.find({"size.h": {$lt: 15}, "size.uom": "in", status: "D"})
< {
  _id: ObjectId('6603cee6d34e6e5f7e140b74'),
  item: 'paper',
  qty: 100,
  size: {
    h: 8.5,
    w: 11,
    uom: 'in'
  },
  status: 'D'
}
```

Query Document

❖ Projection

- Determine which fields are returned in the matching documents
 - {<field 1>: <value>, <field 2>: <value>, ...}
 - <value>
 - 1 or true to include the field in the return documents
 - 0 or false to exclude the field
- Example
 - `db.inventory.find({}, {qty: 1})`
 - `db.inventory.find({}, {qty: 1, _id: 0})`
 - `db.inventory.find({}, {qty: 0})`

Query Document

❖ Projection

- Determine which fields are returned in the matching documents
 - Example
 - `db.inventory.find({}, {qty: 1})`

```
>_MONGOSH
> db.inventory.find({}, {qty: 1})
< {
  _id: ObjectId('6603cee6d34e6e5f7e140b72'),
  qty: 25
}
{
  _id: ObjectId('6603cee6d34e6e5f7e140b73'),
  qty: 50
}
{
  _id: ObjectId('6603cee6d34e6e5f7e140b74'),
  qty: 100
}
{
  _id: ObjectId('6603cee6d34e6e5f7e140b75'),
  qty: 75
}
{
  _id: ObjectId('6603cee6d34e6e5f7e140b76'),
  qty: 45
}
```

Query Document

❖ Projection

- Determine which fields are returned in the matching documents
 - Example
 - `db.inventory.find({}, {qty: 1, _id: 0})`

```
>_MONGOSH
> db.inventory.find({}, {qty: 1, _id: 0})
< {
  qty: 25
}
{
  qty: 50
}
{
  qty: 100
}
{
  qty: 75
}
{
  qty: 45
}
```

Query Document

❖ Projection

- Determine which fields are returned in the matching documents
 - Example
 - `db.inventory.find({}, {qty: 0})`

```
>_MONGOSH
> db.inventory.find({}, {qty: 0})
< {
  _id: ObjectId('6603cee6d34e6e5f7e140b72'),
  item: 'journal',
  size: {
    h: 14,
    w: 21,
    uom: 'cm'
  },
  status: 'A'
},
{
  _id: ObjectId('6603cee6d34e6e5f7e140b73'),
  item: 'notebook',
  size: {
    h: 8.5,
    w: 11,
    uom: 'in'
  },
  status: 'A'
},
{
  _id: ObjectId('6603cee6d34e6e5f7e140b74'),
  item: 'paper',
  size: {
    h: 8.5,
    w: 11,
    uom: 'in'
  },
  status: 'D'
}
```

Query Document

❖ Projection

- You should fill up query part of find() if you want to combine query with projection
- Select all documents that has "paper" item and exclude "_id" and "size" fields from the result
 - `db.inventory.find({item: "paper"}, {size: 0, _id: 0})`
- This operation corresponds to the following SQL statement:
 - `SELECT item, qty, status FROM inventory WHERE item = "paper"`

Query Document

❖ Projection

```
>_MONGOSH
> db.inventory.find({item: "paper"}, {size: 0, _id: 0})
< {
  item: 'paper',
  qty: 100,
  status: 'D'
}
{
  item: 'paper',
  qty: 45,
  status: 'A'
}
```


Part 2

UPDATE DOCUMENT

Update Document

❖ update()

- The update() function updates the values in the selected document
 - `db.COLLECTION_NAME.update(query, updated_document)`
 - Here, query is a selection criteria
 - Both, query and updated_document are document types ({})
 - \$set query operator is used for updating documents
- Update a single document with “paper” items to “paperless” items
 - `db.inventory.update({item: “paper”}, {$set: {item: “paperless”}})`
- By default, MongoDB will update only a single document

Update Document

❖ update()

```
> db.inventory.update({item: "paper"}, {$set: {item: "paperless"}})
< DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Update Document

❖ update()

- To update multiple documents, you need to set a parameter 'multi' to true
- Update all documents with "paper" items to "paperless" items
 - `db.inventory.update({item: "paper"}, {$set: {item: "paperless"}}, {multi: true})`
- This operation corresponds to the following SQL statement:
 - `UPDATE inventory SET item = "paperless" WHERE item = "paper"`
- Update quantity of all documents with status "A" to 0

Update Document

❖ update()

```
>_MONGOSH
> db.inventory.update({item: "paper"}, {$set: {item: "paperless"}}, {multi: true})
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

Part 3

DELETE DOCUMENT

Delete Document

❖ remove()

- Use to remove a document from the collection
- remove() function accepts two parameters
 - Deletion criteria
 - (Optional) deletion criteria according to documents will be removed
 - justOne
 - (Optional) if set to true or 1, then remove only one document
- Basic syntax of remove() function is as follows
 - db.COLLECTION_NAME.remove(query)
 - Here, query is a deletion criteria

Delete Document

❖ remove()

- Following example will remove all documents whose item is "paper"
 - `db.inventory.remove({item: "journal"})`
 - `deleteOne()`, `deleteMany()`
- If there are multiple records and you want to delete only the first record, then use `justOne` parameter
 - `db.inventory.remove({item: "notebook"}, 1)`
- If you want to remove all documents,
 - `db.inventory.remove({})`

Delete Document

❖ remove()

```
>_MONGOSH
```

```
> db.inventory.remove({item: "journal"})
```

```
< DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkWrite.
```

```
< {
```

```
  acknowledged: true,
```

```
  deletedCount: 1
```

```
}
```

Summary

❖ Query document

- find() function
- Query operators
 - Comparison
 - Logical
- Projection

❖ Update document

- update() function

❖ Delete document

- remove() function
- deleteOne(), deleteMany()

Questions?

SEE YOU NEXT TIME!