MongoDB-Query

Reference: https://docs.mongodb.com/manual/

Ascending/Descending Sort

- Specify in the sort parameter the field or fields to sort by and a value of 1 or -1 to specify an ascending or descending sort respectively

- The following sample document specifies a descending sort by the age field and then an ascending sort by the posts field:

```
{ age : -1, posts: 1 }
```

Query on Embedded/Nested Documents

- Consider the following inventory collection

```
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: "D" },
   { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
]);
```

Match an Embedded/Nested Document

- To specify an equality condition on a field that is an embedded/nested document, use the query filter document { <field>: <value> } where <value> is the document to match. For example:

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

- Equality matches on the whole embedded document require an exact match of the specified <value> document, including the field order. For example, the following query does not match any documents in the inventory collection:

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

Query on Nested Field

- To specify a query condition on fields in an embedded/nested document, use **dot notation** ("field.nestedField"), and the field and nested field must be inside quotation marks
- Specify Equality Match on a Nested Field
  - *db.inventory.find({ "size.uom": "in" })*
- Specify Match using Query Operator using

{ <field1>: { <operator1>: <value1> }, ... }

E.g., *db.inventory.find({ "size.h": { $lt: 15 } })*

- Specify AND Condition

db.inventory.find({ "size.h": { $lt: 15 }, "size.uom": "in", status: "D" })

- Specify OR Condition

db.inventory.find({ $or: [{"size.h": { $lt: 15 }}, {"size.uom": "in"}, {status: "D"}]})

Query an Array

- Consider the following inventory collection

```
db.inventory.insertMany([
   { item: "journal", qty: 25, tags: ["blank", "red"], dim_cm: [ 14, 21 ] },
   { item: "notebook", qty: 50, tags: ["red", "blank"], dim_cm: [ 14, 21 ] },
   { item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim_cm: [ 14, 21 ] },
   { item: "planner", qty: 75, tags: ["blank", "red"], dim_cm: [ 22.85, 30 ] },
   { item: "postcard", qty: 45, tags: ["blue"], dim_cm: [ 10, 15.25 ] }
]);
```

Match an Array

- To specify equality condition on an array, use the query document { <field>: <value> } where <value> is the exact array to match, including the order of the elements

  db.inventory.find({ tags: ["red", "blank"] })

- If, instead, you wish to find an array that contains both the elements "red" and "blank", without regard to order or other elements in the array, use the $all operator:

  db.inventory.find({ tags: { $all: ["red", "blank"] } })

Query an Array for an Element

- To query if the array field contains at least one element with the specified value, use the filter { <field>: <value> } where <value> is the element value

  db.inventory.find({ tags: "red" })

- To specify conditions on the elements in the array field, use query operators in the query filter document:

  db.inventory.find({ dim_cm: { $gt: 25 } })

  dim_cm is an array!

Specify Multiple Conditions for Array Elements

- Query an Array with Compound Filter Conditions on the Array Elements
  - The following example queries for documents where the dim_cm array contains elements that in some combination satisfy the query conditions; e.g., one element can satisfy the greater than 15 condition and another element can satisfy the less than 20 condition, or a single element can satisfy both

db.inventory.find({ dim_cm: { $gt: 15, $lt: 20 } })

Too many possibilities, how to specify?

Specify Multiple Conditions for Array Elements

- Query for an Array Element that Meets Multiple Criteria
  - Use $elemMatch operator to specify multiple criteria on the elements of an array such that at least one array element satisfies all the specified criteria

    db.inventory.find({ dim_cm: { $elemMatch: { $gt: 15, $lt: 20 } } })

Query for an Element by the Array Index Position

- The following example queries for all documents where the second element in the array dim_cm is greater than 25:

  db.inventory.find({ "dim_cm.1": { $gt: 25 } })

  Again, the index starts from zero

Query an Array by Array Length

- Use the $size operator to query for arrays by number of elements. For example, the following selects documents where the array tags has 3 elements

  db.inventory.find({ tags: { $size: 3 } })

Query for a Document Nested in an Array

- Consider the following collection

```
db.inventory.insertMany( [
   { item: "journal", instock: [ { warehouse: "A", qty: 5 }, { warehouse: "C", qty: 15 } ] },
   { item: "notebook", instock: [ { warehouse: "C", qty: 5 } ] },
   { item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 15 } ] },
   { item: "planner", instock: [ { warehouse: "A", qty: 40 }, { warehouse: "B", qty: 5 } ] },
   { item: "postcard", instock: [ { warehouse: "B", qty: 15 }, { warehouse: "C", qty: 35 } ] }
]);
```

Query for a Document Nested in an Array

- The following example selects all documents where an element in the instock array matches the specified document:

  db.inventory.find({ instock: { warehouse: "A", qty: 5 } })

- Order matters. The following not working

  db.inventory.find({ instock: { qty: 5, warehouse: "A" } })

Specify a Query Condition on a Field in an Array of Documents

- If you do not know the index position of the document nested in the array, concatenate the name of the array field, with a dot (.) and the name of the field in the nested document
  - The following example selects all documents where the instock array has at least one embedded document that contains the field qty whose value is less than or equal to 20:

    db.inventory.find({"instock.qty": { $lte: 20 } })
- Use the Array Index to Query for a Field in the Embedded Document
  - The following example selects all documents where the instock array has as its first element a document that contains the field qty whose value is less than or equal to 20:

    db.inventory.find({"instock.0.qty": { $lte: 20 } })

Specify Multiple Conditions for Array of Documents

- A Single Nested Document Meets Multiple Query Conditions on Nested Fields
  - Use $elemMatch operator to specify multiple criteria on an array of embedded documents such that at least one embedded document satisfies all the specified criteria

  - The following example queries for documents where the instock array has at least one embedded document that contains both the field qty equal to 5 and the field warehouse equal to A (order does not matter):

    db.inventory.find({instock: { $elemMatch: { qty: 5, warehouse: "A" } } })

  - The following example queries for documents where the instock array has at least one embedded document that contains the field qty that is greater than 10 and less than or equal to 20:

    db.inventory.find({instock: { $elemMatch: { qty: { $gt: 10, $lte: 20 } } } })

Specify Multiple Conditions for Array of Documents

- Combination of Elements Satisfies the Criteria
  - If the compound query conditions on an array field do not use the $elemMatch operator, the query selects those documents whose array contains any combination of elements that satisfies the conditions

  - The following query matches documents where any document nested in the instock array has the qty field greater than 10 and any document (but not necessarily the same embedded document) in the array has the qty field less than or equal to 20:

    db.inventory.find({ "instock.qty": { $gt: 10, $lte: 20 } })

  - The following example queries for documents where the instock array has at least one embedded document that contains the field qty equal to 5 and at least one embedded document (but not necessarily the same embedded document) that contains the field warehouse equal to A:

    db.inventory.find({ "instock.qty": 5, "instock.warehouse": "A" })

Query for Null or Missing Fields

- Consider the following collection

```
db.inventory.insertMany([
  { _id: 1, item: null },
  { _id: 2 }
])
```

- The { item : null } query matches documents that either contain the item field whose value is null or that do not contain the item field

  db.inventory.find({ item : null })

  What will be returned?

Query for Null or Missing Fields

- The { item : { $type: 10 } } query matches only documents that contain the item field whose value is null; i.e., the value of the item field is of BSON Type Null (type number 10) :

  db.inventory.find({ item : { $type: 10 } })

- The { item : { $exists: false } } query matches documents that do not contain the item field:

  db.inventory.find({ item : { $exists: false } })

Project Fields to Return from Query

- By default, queries in MongoDB return all fields in matching documents.
- How to specify fields as using SELECT in RDBMS?
- Consider the following collection

```
db.inventory.insertMany( [
  { item: "journal", status: "A", size: { h: 14, w: 21, uom: "cm" }, instock: [ { warehouse: "A", qty: 5 } ] },
  { item: "notebook", status: "A",  size: { h: 8.5, w: 11, uom: "in" }, instock: [ { warehouse: "C", qty: 5 } ] },
  { item: "paper", status: "D", size: { h: 8.5, w: 11, uom: "in" }, instock: [ { warehouse: "A", qty: 60 } ] },
  { item: "planner", status: "D", size: { h: 22.85, w: 30, uom: "cm" }, instock: [ { warehouse: "A", qty: 40 } ] },
  { item: "postcard", status: "A", size: { h: 10, w: 15.25, uom: "cm" }, instock: [ { warehouse: "B", qty: 15 }, { warehouse: "C", qty: 35 } ] }
]);
```

Return the Specified Fields and the _id Field Only

- A projection can explicitly include several fields by setting the <field> to 1 in the projection document. The following operation returns all documents that match the query
  - General syntax: db.collection.find(query, projection)

  db.inventory.find({ status: "A" }, { item : 1, status : 1 })

- The operation corresponds to the following SQL statement:

  SELECT _id, item, status from inventory WHERE status = "A"

- You can remove the _id field from the results by setting its exclusion <field> to 0 in the projection, as in the following example:

  db.inventory.find({ status: "A" }, { item : 1, status : 1, _id: 0 }) // only _id can mix

  db.inventory.find({ status: "A" }, { item : 0, status : 1 }) // not working

Return the Specified Fields and the _id Field Only

- Similarly, you can use a projection to exclude specific fields. The following example which returns all fields except for the status and the instock fields in the matching documents:

  db.inventory.find({ status: "A" }, { status : 0, instock : 0 })

Project Fields to Return from Query

- Return Specific Fields in Embedded Documents

  db.inventory.find({ status: "A" }, { item : 1, status : 1, "size.uom": 1 })

- Suppress

  db.inventory.find({ status: "A" }, { "size.uom": 0 })

  db.inventory.find({ status: "A" }, { "instock.warehouse": 0 })

Project Fields to Return from Query

- Projection on Embedded Documents in an Array

  db.inventory.find({ status: "A" }, { item: 1, status: 1, instock : 1 })

  db.inventory.find({ status: "A" }, { item: 1, status: 1, "instock.warehouse": 1 })

  db.inventory.find({ status: "A" }, { item: 1, status: 1, "instock.qty": 1 })