Querying Complex Data with MQL

Using MQL to Query Complex Data

- Embedded/Nested Documents
- Arrays
- Array of Nested Documents
- Project fields to return

Embedded / Nested Documents

Embedding/Nesting Documents

Embedded data exists in a single document

Can be viewed as “denormalized” model

Use dot notation to access data within the nested/embedded document

Provides better read performance, allows for all related data to be request and retrieved in one database operation.

Provides for the updates on related data to be done in a single *atomic* write.

Example: Embedded Data

```
{
  _id: <ObjectId1>,
  username: “123xyz”,
  contact: {
              phone: “123-456-7890”,
              email: xyz@example.com
            },
  access: {
              level: 5
              group: “dev”
            }
}
```

Embedded sub- document

Embedded sub- document

Embedding/Nesting Documents

Embedded data exists in a single document

Can be viewed as a “denormalized” model

Use dot notation to access data within the nested/embedded document

Provides better read performance, allows for all related data to be request and retrieved in one database operation.

Provides for the updates on related data to be done in a single *atomic* write.

Embedding/Nesting Documents

Embedded data exists in a single document

Can be viewed as “denormalized” model

Uses dot notation to access data within the nested/embedded document

Provides better read performance, allows for all related data to be request and retrieved in one database operation.

Provides for the updates on related data to be done in a single *atomic* write.

What is dot notation? How do I use it?

Dot notation, or using the "." character, is used to access the fields of an embedded document or the elements of an array.

"<array>.<index>" is the syntax for array element access

"<embedded document>.<field>" is the syntax for accessing a field within an embedded document

Embedding/Nesting Documents

Embedded data exists in a single document

Can be viewed as “denormalized” model

Use dot notation to access data within the nested/embedded document

Provides better read performance, allows for all related data to be request and retrieved in one database operation.

Provides for the updates on related data to be done in a single *atomic* write.

```
{
  "_id" : ObjectId("5ad88534e3632e1a35a58d0b"),
  "sport": "tennis",
  "athlete_first_name": "Martina",
  "athlete_surname": "Navratilova",
  "athlete_full_name": "Martina Navratilova",
  "competition_earnings": {value:
NumberDecimal("216226089"), currency:"USD"},
  "number_of_tournaments": 390,
  "number_of_titles": 177,
  "event": [
  +   {...}
  ],
...
}
```

Embedding Example

In this document we highlight an example using a tennis player who competed in both singles and doubles competitions.

We are able to use document embedding to keep all the data in a single document.

These details aren't exactly the same, so we will introduce the polymorphic pattern which helps us structure the document to allow similar but not exactly the same information to be embedded together in a single document.

Polymorphic Pattern

- Allows single query for all data on a sports person
- Adds additional code paths

Let’s look at the embedded document

```
{
  "_id" :
ObjectId("5ad88534e3632e1a35a58d0b"),
  "sport": "tennis",
  "athlete_first_name": "Martina",
  "athlete_surname": "Navratilova",
  "athlete_full_name": "Martina
Navratilova",
  "competition_earnings": {value:
NumberDecimal("216226089"),
currency:"USD"},
  "number_of_tournaments": 390,
  "number_of_titles": 177,
  "event": [
  +   {...}
  ]
}
```

```
{
  "_id" : ObjectId("5ad88534e3632e1a35a58d0b"),
  "sport": "tennis",
  "athlete_first_name": "Martina",
  "athlete_surname": "Navratilova",
  "athlete_full_name": "Martina Navratilova",
  "competition_earnings": {value: NumberDecimal("216226089"),
currency:"USD"},
  "number_of_tournaments": 390,
  "number_of_titles": 177,
  "event": [ {
       "type": "singles",
       "number_of_tournaments": 390,
       "number_of_titles": 167
  },
  {
       "type": "doubles",
       "number_of_tournaments": 233,
       "number_of_titles": 177,
       "partner_full_name": ["Renáta Tomanová",
     "Beatriz Fernández", "Olga Morozova", "Chris
     Evert"]
  } ]
}
```

Exercise: Embedding Documents

Let’s insert a document with embedded data

```
>>> sportsCol = db.getCollection("sports")
>>> db.sportsCol.insertOne({"sport" : "tennis", "athlete_first_name" : "Martina",
"athlete_surname" : "Navratilova", "athlete_full_name" : "Martina Navratilova",
"competition_earnings" : { "value" : NumberDecimal("216226089"), "currency" : "USD" },
"number_of_tournaments" : 390, "number_of_titles" : 177, "event" : [ { "type" : "singles",
"number_of_tournaments" : 390, "number_of_titles" : 167 }, { "type" : "doubles",
"number_of_tournaments" : 233, "number_of_titles" : 177, "partner_full_name" : [ "Renáta
Tomanová", "Beatriz Fernández", "Olga Morozova", "Chris Evert" ] } ] })
…
{
      acknowledged : true,
      insertedId : ObjectId(5f3a594ae90262aae835efba)
```

Exercise: Embedding Documents

Let’s use find() and project on embedded data

```
>>> sportsCol.find({"athlete_full_name": "Martina
Navratilova"},{"event.type": 1})

{ "_id": ObjectId(5f3a594ae90262aae835efba), "event" : [ { "type" :
"singles" }, { "type" : "doubles" } ] }
```

Exercise: Embedding Documents

Using the same window, change **<a>** to the embedded field for event types.

Change **<b>** to the value necessary to find the singles titles.

To shorten the response, we use project to limit it to the event document.

```
>>> sportsCol.find({"<a>": <b>}, {event: 1})

{ "_id" : ObjectId("5f3a594ae90262aae835efba"), "event" : [ { "type" :
"singles", "number_of_tournaments" : 390, "number_of_titles" : 167 }, {
"type" : "doubles", "number_of_tournaments" : 233, "number_of_titles" :
177, "partner_full_name" : [ "Renáta Tomanová", "Beatriz Fernández", "Olga
Morozova", "Chris Evert" ] } ]
```

Embedding/Nesting Documents

Embedded data exists in a single document

Can be viewed as “denormalized” model

Use dot notation to access data within the nested/embedded document

Provides better read performance, allows for all related data to be request and retrieved in one database operation.

Provides for the updates on related data to be done in a single *atomic* write.

Arrays

Arrays

Query to match the entire array

Query to match for a specific element in the array

Query such that either a single array element meets these condition or any combination of array elements meets the conditions

Query for an element by the array index position

Query to match arrays of a given size

Querying Arrays: Exercise

Let’s insert some data with arrays!

```
>>> db.inventory.drop()
>>> 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 ] }
]);
…
{
      acknowledged : true,
      insertedId : ObjectId(5f3b939f63a92a8719c01239)
}
```

Querying Arrays: Exercise

Find a complete exact array for the “tags” field.

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

{ "_id" : ObjectId("5f3b939f63a92a8719c01236"), "item" : "notebook",
"qty" : 50, "tags" : [ "red", "blank" ], "dim_cm" : [ 14, 21 ] }
```

Arrays

Query to match the entire array

Query to match for a specific element in the array

Query such that either a single array element meets these condition or any combination of array elements meets the conditions

Query for an element by the array index position

Query to match arrays of a given size

Querying Arrays to Match: Exercise

Let’s match a specific element in the array

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

{ "_id" : ObjectId("5f3b939f63a92a8719c01235"), "item" : "journal", "qty" : 25,
"tags" : [ "blank", "red" ], "dim_cm" : [ 14, 21 ] }

{ "_id" : ObjectId("5f3b939f63a92a8719c01236"), "item" : "notebook", "qty" : 50,
"tags" : [ "red", "blank" ], "dim_cm" : [ 14, 21 ] }

{ "_id" : ObjectId("5f3b939f63a92a8719c01237"), "item" : "paper", "qty" : 100, "tags"
: [ "red", "blank", "plain" ], "dim_cm" : [ 14, 21 ] }

{ "_id" : ObjectId("5f3b939f63a92a8719c01238"), "item" : "planner", "qty" : 75,
"tags" : [ "blank", "red" ], "dim_cm" : [ 22.85, 30 ] }
```

Arrays

Query to match the entire array

Query to match for a specific element in the array

Query such that either a single array element meets these conditions or any combination of array elements meets the conditions

Query for an element by the array index position

Query to match arrays of a given size

Querying Arrays with Conditions: Exercise

Let’s match on conditions for the array

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

{ "_id" : ObjectId("5f3b939f63a92a8719c01235"), "item" : "journal", "qty" : 25,
"tags" : [ "blank", "red" ], "dim_cm" : [ 14, 21 ] }

{ "_id" : ObjectId("5f3b939f63a92a8719c01236"), "item" : "notebook", "qty" :
50, "tags" : [ "red", "blank" ], "dim_cm" : [ 14, 21 ] }

{ "_id" : ObjectId("5f3b939f63a92a8719c01237"), "item" : "paper", "qty" : 100,
"tags" : [ "red", "blank", "plain" ], "dim_cm" : [ 14, 21 ] }

{ "_id" : ObjectId("5f3b939f63a92a8719c01239"), "item" : "postcard", "qty" :
45, "tags" : [ "blue" ], "dim_cm" : [ 10, 15.25 ] }
```

Querying Arrays with Conditions: Exercise

Find results with a qty field where <50 and >20

```
>>> db.inventory.find( { qty: { $gt: 20, $lt : 50} } )

{ "_id" : ObjectId("5f3b939f63a92a8719c01235"), "item" :
"journal", "qty" : 25, "tags" : [ "blank", "red" ], "dim_cm" :
[ 14, 21 ] }

{ "_id" : ObjectId("5f3b939f63a92a8719c01239"), "item" :
"postcard", "qty" : 45, "tags" : [ "blue" ], "dim_cm" : [ 10,
15.25 ] }
```

Arrays

Query to match the entire array

Query to match for a specific element in the array

Query such that either a single array element meets these conditions or any combination of array elements meets the conditions

Query for an element by the array index position

Query to match arrays of a given size

Querying Arrays by Index Position: Exercise

Let’s match by array index position

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

{ "_id" : ObjectId("5f3b939f63a92a8719c01238"), "item" : "planner", "qty" :
75, "tags" : [ "blank", "red" ], "dim_cm" : [ 22.85, 30 ] }

>>> db.inventory.find( { dim_cm.0: { $lt: 14 } } )

{ "_id" : ObjectId("5f3b939f63a92a8719c01239"), "item" : "postcard", "qty"
: 45, "tags" : [ "blue" ], "dim_cm" : [ 10, 15.25 ] }
```

Arrays

Query to match the entire array

Query to match for a specific element in the array

Query such that either a single array element meets these conditions or any combination of array elements meets the conditions

Query for an element by the array index position

Query to match arrays of a given size

Querying Arrays by Size: Exercise

Let’s match by the size/length of the array

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

{ "_id" : ObjectId("5f3b939f63a92a8719c01237"), "item" :
"paper", "qty" : 100, "tags" : [ "red", "blank", "plain" ],
"dim_cm" : [ 14, 21 ] }
```

Querying Arrays with One Element: Exercise

Find docs where the tags array has 1 element

Using the same window, change **<A>** to 1 in the query below to search the field tags for an array with one element. The results should be the same as shown below.

```
>>> db.inventory.find( { "tags": { $size: <A> } } )

{ "_id" : ObjectId("5f3b939f63a92a8719c01239"), "item" :
"postcard", "qty" : 45, "tags" : [ "blue" ], "dim_cm" : [
10, 15.25 ] }
```

Array of Nested Documents

Array of Nested Documents

In MongoDB we can have arrays which consist of elements that are in of themselves documents.

Field ordering within query criteria can be important:

- Equality matching with field A: value X and field B: value Y will only return that exact sequence of field-value pairs.
- It will not find an array with field B: value Y and then field A: value X.

Querying an Array of Nested Documents: Exercise

Let’s insert some nested documents in an array!

```
>>> db.inventory.drop()
>>> 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 } ] }
]);
…
{
     acknowledged : true,
     insertedId : ObjectId(5f3b939f63a92a8719c01239)
}
```

Querying an Array of Nested Documents: Exercise

Let’s query the nested documents

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

{ "_id" : ObjectId("5f3bc03563a92a8719c0123a"), "item" : "journal", "instock" : [ {
"warehouse" : "A", "qty" : 5 }, { "warehouse" : "C", "qty" : 15 } ] }

>>> db.inventory.find( { 'instock.qty': { $lte: 5 } } )

{ "_id" : ObjectId("5f3bc03563a92a8719c0123a"), "item" : "journal", "instock" : [ {
"warehouse" : "A", "qty" : 5 }, { "warehouse" : "C", "qty" : 15 } ] }

{ "_id" : ObjectId("5f3bc03563a92a8719c0123b"), "item" : "notebook", "instock" : [ {
"warehouse" : "C", "qty" : 5 } ] }

{ "_id" : ObjectId("5f3bc03563a92a8719c0123d"), "item" : "planner", "instock" : [ {
"warehouse" : "A", "qty" : 40 }, { "warehouse" : "B", "qty" : 5 } ] }
```

Querying an Array of Nested Documents: Exercise

Let’s query with several conditionals

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

{ "_id" : ObjectId("5f3bc03563a92a8719c0123a"), "item" : "journal", "instock" : [ {
"warehouse" : "A", "qty" : 5 }, { "warehouse" : "C", "qty" : 15 } ] }

{ "_id" : ObjectId("5f3bc03563a92a8719c0123c"), "item" : "paper", "instock" : [ {
"warehouse" : "A", "qty" : 60 }, { "warehouse" : "B", "qty" : 15 } ] }

{ "_id" : ObjectId("5f3bc03563a92a8719c0123e"), "item" : "postcard", "instock" : [ {
"warehouse" : "B", "qty" : 15 }, { "warehouse" : "C", "qty" : 35 } ] }
```

Querying an Array of Nested Documents: Exercise

Find docs with a complex conditional

Using the same window, change <A> to 5, <B> to the less than or equal operator, and <C> to the field which represents which warehouse the inventory is stored in. The results should be the same as shown below.

```
>>> db.inventory.find( { "instock": { $elemMatch: { qty: { $gte:
<A>, <B>: 20 }, "<C>": "A" } } } )

{ "_id" : ObjectId("5f50c9962d4b45b7f11b6d89"), "item" :
"journal", "instock" : [ { "warehouse" : "A", "qty" : 5 }, {
"warehouse" : "C", "qty" : 15 } ] }
```

Project Fields to Return

Project Fields to Return

- A projection can help use restrict or explicitly specify which fields should be returned from the query.
- Without a projection in a query, all documents matching the query are returned.
- A projection can explicitly include several fields by setting the field to 1 in the projection document or explicitly exclude fields by setting the field to 0.
- Dot notation can be used with projection on nested documents or arrays.

Projection: Exercise

Let’s insert some nested documents in an array!

```
>>> db.inventory.drop()
>>> 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 } ] }
]);
…
```

Projection: Exercise

```
>>> db.inventory.find( { status: "A" }, { _id: 0, item: 1, "instock.qty": 1 } )

{ "item" : "journal", "instock" : [ { "qty" : 5 } ] }

{ "item" : "notebook", "instock" : [ { "qty" : 5 } ] }

{ "item" : "postcard", "instock" : [ { "qty" : 15 }, { "qty" : 35 } ] }

>>> db.inventory.find( { status: "A" }, { _id:0, item: 1, "size.uom": 1 } )

{ "item" : "journal", "size" : { "uom" : "cm" } }

{ "item" : "notebook", "size" : { "uom" : "in" } }

{ "item" : "postcard", "size" : { "uom" : "cm" } }
```