MongoDB queries: arrays

Adrian Velonis

Querying arrays

- Query for one element in an array:
  - ```
    db.collection.find({
        "field": "value"
    })
    ```
  - Returns all objects that contain `value` in the `field` array
- Query for multiple elements in an array:
  - ```
    db.collection.find({
        field: {$all: [
            "value1",
            "value2"]}
    })
    ```
  - Returns all objects that contain `value1` and `value2` in the `field` array
  - Order does not matter with the `$all` operation

Querying arrays (example)

Database:

```
[
     {
          "fruit": [
               "apple",
               "banana",
               "peach"
          ]
     },
     {
          "fruit": [
               "orange",
               "grape"
          ]
     }
]
```

Query: find arrays containing “banana”:

```
db.collection.find({
     "fruit": "banana"
})
```

Query: find arrays containing “apple” and “banana”:

```
db.collection.find({
  fruit: {
    $all: [
      "apple",
      "banana"
    ]
  }
})
```

Output:

```
[
  {
    "_id":
ObjectId("5a934e000102030
405000000"),
    "fruit": [
      "apple",
      "banana",
      "peach"
    ]
  }
]
```

Querying arrays (cont.)

- You can also search for an entire array without `$all`:
  - ```
    db.collection.find({
        "field": ["value1", "value2", "value3"]
    })
    ```
  - Returns all objects with the exact array field and values you specify
- Query for a value at a specific index of an array:
  - ```
    db.collection.find({
        "field.index": "value"
    })
    ```
  - Returns all objects that contain `value` at the `index` you specify for `field`
  - Arrays are 0-indexed
- Query for arrays of a given size:
  - ```
    db.collection.find({
        "field": {"$size": 3}
    })
    ```
  - Returns the `field` array object if value is in the field array at the index you specify
    ```

Querying arrays (`$slice`)

- Query for a subset (slice) of elements in an array:
  - ```
    db.collection.findOne(
        <query>,
        { <arrayField>: { $slice: <n> } }
    );
    ```
  - Returns the first `n` values in `arrayField`
  - To find values from the end of `arrayField`, use a negative value of `n`
  - To find a range of values starting in the middle, use this syntax:
    - `{ <arrayField>: { $slice: [ <i> <n> ] } }`
    - This returns `n` values starting at index `i`
  - You can also use `find` instead of `findOne`
    ```

Querying arrays (range)

- Query for a range of values of a field:
  - ```
    db.collection.find({
        "x": {"$gt": number1, "$lt": number2}
    }).min({
        "x": number1
    }).max({
        "x": number2
    })
    ```
    - Where `$gt` is “greater than” and `$lt` is “less than”
    - Returns documents in which `x` is greater than `number1` and less than `number2`
    - This is kind of inefficient, but it works
    ```

Querying arrays (embedded documents)

Database:

```
[{
    "name" : {
        "first": "John",
        "last": "Doe"
    },
    "age": 21
}]
```

Query: search for specific key in an embedded document:

```
db.collection.find({
     "name.first": "John",
     "name.last": "Doe"
})
```

Better than searching for a entire subdocument:

```
db.collection.find({
     "name" {
          "first": "John",
          "last": "Doe"
     }
})
```

(If more keys are added, this query will not work)

Querying arrays (`$where` queries)

- `$where` clause allows you to execute JavaScript in a query
-
```
db.collection.find{(
    $where: function() {
        for (var x in this) {
            if (x == y) {
                return true;
            }
        }
    return false;
    }
})
```
- Very slow compared to regular queries

Sources

- https://docs.mongodb.com/manual/tutorial/query-documents/
- https://docs.mongodb.com/manual/reference/operator/projection/slice/
- https://docs.mongodb.com/manual/reference/operator/query/where/