

# MongoDB教育訓練-03

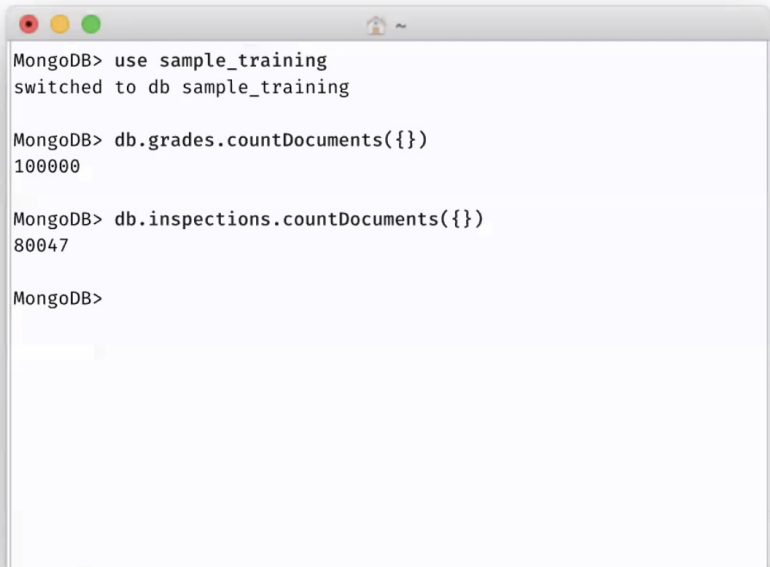
## Validate Loaded Data

Connect to the MongoDB  
Atlas Cluster using mongosh

Verify that we have loaded  
proper data.

Database to use:  
**sample\_training**

Collections to verify: **grades**  
and **inspections**



```
MongoDB> use sample_training
switched to db sample_training

MongoDB> db.grades.countDocuments({})
100000

MongoDB> db.inspections.countDocuments({})
80047

MongoDB>
```

```
Atlas atlas-vnfeuv-shard-0 [primary] sample_training> db.grades.countDocuments({})
100000
Atlas atlas-vnfeuv-shard-0 [primary] sample_training> db.inspections
db.inspections

Atlas atlas-vnfeuv-shard-0 [primary] sample_training> db.inspections.countDocuments()
80047
Atlas atlas-vnfeuv-shard-0 [primary] sample_training> db.inspections.count()
80047
Atlas atlas-vnfeuv-shard-0 [primary] sample_training> █
```

count 儘量不要用

db.inspections.countDocuments() #精確查詢(1)

db.inspections.count() #

# Basic Database CRUD Interactions

	Single Document	Multiple Documents
Create Documents	<code>insertOne(doc)</code>	<code>insertMany([doc,doc,doc])</code>
Read Documents	<code>findOne(query, projection)</code>	<code>find(query, projection)</code>
Update Documents	<code>updateOne(query, change)</code>	<code>updateMany(query, change)</code>
Delete Documents	<code>deleteOne(query)</code>	<code>deleteMany(query)</code>

## Creating New Documents with insertOne()

Documents can be added to a collection with `insertOne()`

Documents are Objects in the programming language being used.

`_id` will be added if not supplied, and must be unique if supplied.

```
MongoDB> db.customers.insertOne({ _id : "bob@gmail.com", name:
"Robert Smith", orders: [], spend: 0, lastpurchase: null })
{ "acknowledged": true, "insertedId" : "bob@gmail.com" }

MongoDB> db.customers.insertOne({ _id : "bob@gmail.com", name:
"Bobby Smith", orders: [], spend: 0, lastpurchase: null })
MongoServerError: E11000 duplicate key error collection:
sample_training.customers index:_id_ dup key:{_id:"bob@gmail.com"}

MongoDB> db.customers.insertOne({name: "Andi Smith", orders: [],
spend: 0, lastpurchase: null })
{"acknowledged": true,
"insertedId": ObjectId("609ab0c1aeb5ca24f9daa254") }
```

## insertOne 插入一筆資料

```
#插入一條資料
insertOne
```

```

Atlas atlas-vnfeuv-shard-0 [primary] sample_training> db.customers.insertOne({_id: "yaoxing.zhang@mongodb.com", department: "Professional Service"})
{ acknowledged: true, insertedId: 'yaoxing.zhang@mongodb.com' }
Atlas atlas-vnfeuv-shard-0 [primary] sample_training> db.customers
db.customers

Atlas atlas-vnfeuv-shard-0 [primary] sample_training> db.customers.find()
[
  {
    _id: 'yaoxing.zhang@mongodb.com',
    department: 'Professional Service'
  }
]
Atlas atlas-vnfeuv-shard-0 [primary] sample_training> db.customers.insertOne({name: "Yaoxing", department: "Professional Service"})
{ acknowledged: true,
  insertedId: ObjectId("61bad04a013c47ef96e04d4e")
}
Atlas atlas-vnfeuv-shard-0 [primary] sample_training> db.customers.find()
[
  {
    _id: 'yaoxing.zhang@mongodb.com',
    department: 'Professional Service'
  },
  {
    _id: ObjectId("61bad04a013c47ef96e04d4e"),
    name: 'Yaoxing',
    department: 'Professional Service'
  }
]
Atlas atlas-vnfeuv-shard-0 [primary] sample_training>

```

id 本身會包含時間

```
getTimestamp #取得時間
```

```

Atlas atlas-vnfeuv-shard-0 [primary] sample_training> ObjectId("61bad04a013c47ef96e04d4e").getTimestamp()
ISODate("2021-12-16T05:36:10.000Z")

```

## insertMany 插入多筆資料

### Adding Multiple Documents with insertMany()

Accepts an array of documents.

Single network call normally - reduces network time

Returns an object with information about each insert.

```

// 1000 Network Calls
MongoDB> let st = ISODate()
for(let d=0;d<1000;d++) {
  db.orders.insertOne({ product: "socks", quantity: d})
}
print(`${ISODate()-st} milliseconds`)

9106ms

// 1 Network call, same data
MongoDB> st = ISODate()
let docs = []
for(let d=0;d<1000;d++) {docs.push({ product: "socks", quantity: d})}

db.orders.insertMany(docs)
print(`${ISODate()-st} milliseconds`)

51ms

```

## 有序插入 & 無序插入

```
{ordered: true | false}
```

- 有序可保證順序，速度較慢，中間錯誤會全部返回
- 使用無序不影響，中間錯誤紀錄

## insertMany() – Ordering of operations

insertMany() can be ordered or unordered.

Default is ordered which stops on first error.

Unordered allows the operation to report errors but keep going.

Unordered can be reordered by the server to make the operation faster.

```
MongoDB> let friends = [ { _id: "joe" }, { _id: "bob" },  
  { _id: "joe" }, { _id: "jen" } ]  
  
MongoDB> db.collection1.insertMany(friends)  
{  
  "errmsg" : "E11000 duplicate key error collection:  
test.example index: _id_ dup key: { _id: \"joe\" }",  
  "nInserted" : 2 }  
  
MongoDB> db.collection2.insertMany(friends,{ordered:false})  
{  
  "errmsg" : "E11000 duplicate key error collection:  
test.example index: _id_ dup key: { _id: \"joe\" }",  
  "nInserted" : 3 }  
  
MongoDB> db.collection1.find()  
{ "_id" : "joe" }  
{ "_id" : "bob" }  
MongoDB> db.collection2.find()  
{ "_id" : "joe" }  
{ "_id" : "bob" }  
{ "_id" : "jen" }
```

## 搜尋

- find()
- findOne()

## Finding and Retrieving a document.

- findOne() retrieves a single document.
- We pass it a document to "query-by-example"

```
MongoDB> db.customers.insertOne({ _id : "tim@gmail.com", name:  
"Timothy", orders: [], spend: 0, lastpurchase: null })  
  
{ "acknowledged": true, "insertedId" : "tim@gmail.com" }  
  
MongoDB> db.customers.findOne({ _id : "tim@gmail.com" })  
  
{  
  "_id" : "tim@gmail.com",  
  "name" : "Timothy",  
  "orders" : [ ],  
  "spend" : 0,  
  "lastpurchase" : null  
}  
  
MongoDB> db.customers.findOne({ spend: 0 })  
MongoDB> db.customers.findOne({ spend: 0 , name: "Timothy" })  
MongoDB> db.customers.findOne({ name: "timothy" }) // No match  
MongoDB> db.customers.findOne({ spend: "0" }) // No Match  
MongoDB> db.customers.findOne({}) // Match everything
```

```

Atlas atlas-vnfeuv-shard-0 [primary] sample_training> db.customers.findOne()
{
  _id: 'yaoxing.zhang@mongodb.com',
  department: 'Professional Service'
}
Atlas atlas-vnfeuv-shard-0 [primary] sample_training> db.customers.find()
[
  {
    _id: 'yaoxing.zhang@mongodb.com',
    department: 'Professional Service'
  },
  {
    _id: ObjectId("61bad04a013c47ef96e04d4e"),
    name: 'Yaoxing',
    department: 'Professional Service'
  }
]

```

## 返回特定欄位

### Projection – choosing the fields to return

We can add a projection parameter to a find operation.

Documents can be large, so we may want a subset.

Simple projections are including or excluding a set of fields.

```

MongoDB> db.customers.insertOne({ _id : "ann@gmail.com", name:
"Ann", orders: [], spend: 0, lastpurchase: null })

MongoDB> db.customers.findOne({ name: "Ann" })
{ "_id" : "ann@gmail.com",
  "name" : "Ann",
  "orders" : [], spend: 0, lastpurchase: null }

MongoDB> db.customers.findOne({ name:"Ann" }, {name: 1, spend: 1})
{ "_id" : "ann@gmail.com", "name" : "Ann", "spend" : 0 }

MongoDB> db.customers.findOne({ name:"Ann" }, {name: 0, orders:0})
{ "_id" : "ann@gmail.com", "spend" : 0, "lastpurchase" : null }

MongoDB> db.customers.findOne({ name:"Ann" }, {name: 0, orders:1})
MongoServerError: Cannot do inclusion on field orders in exclusion
projection

MongoDB> db.customers.findOne({ name:"Ann" }, {_id: 0, name:1})
{ "name" : "Ann" }

```

```

Atlas atlas-vnfeuv-shard-0 [primary] sample_training> db.customers.findOne({}, {name: 1})
{ _id: 'yaoxing.zhang@mongodb.com' }
Atlas atlas-vnfeuv-shard-0 [primary] sample_training> db.customers.find({}, {name: 1})
[
  { _id: 'yaoxing.zhang@mongodb.com' },
  { _id: ObjectId("61bad04a013c47ef96e04d4e"), name: 'Yaoxing' }
]
Atlas atlas-vnfeuv-shard-0 [primary] sample_training> db.customers.find({}, {name: 1, _id: 0})
[ {}, { name: 'Yaoxing' } ]

```

Find 搜尋20筆資料

```
for(var i = 0; i < 200; i++) db.taxis.insertOne({plate: i})
```



```
# plate = 5
db.taxis.find({plate:5})

# plate >= 5
db.taxis.find({plate:{$gt: 5}})

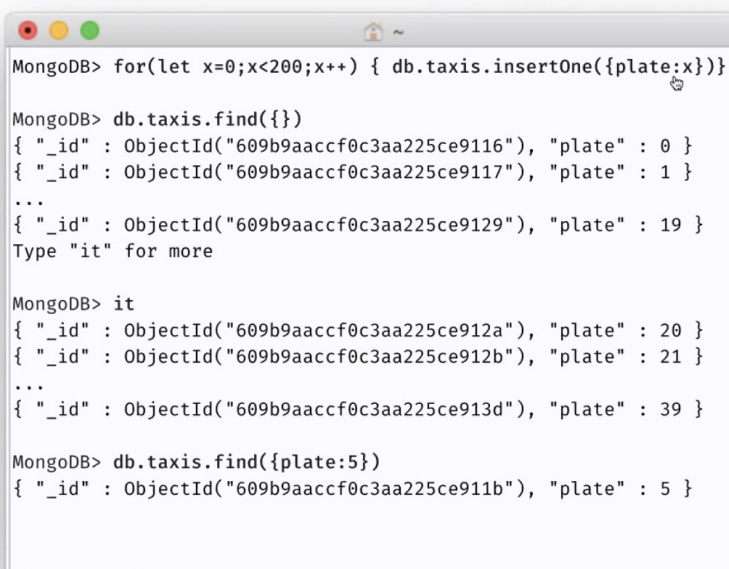
#下20筆
it
```

## Fetching multiple documents using find()

find() returns a cursor object rather than a single document.

We can keep fetching documents from the cursor to get all matches.

When mongosh displays a cursor object it fetches and shows 20 documents from the cursor.



```
MongoDB> for(let x=0;x<200;x++) { db.taxis.insertOne({plate:x})}

MongoDB> db.taxis.find({})
{ "_id" : ObjectId("609b9aaccf0c3aa225ce9116"), "plate" : 0 }
{ "_id" : ObjectId("609b9aaccf0c3aa225ce9117"), "plate" : 1 }
...
{ "_id" : ObjectId("609b9aaccf0c3aa225ce9129"), "plate" : 19 }
Type "it" for more

MongoDB> it
{ "_id" : ObjectId("609b9aaccf0c3aa225ce912a"), "plate" : 20 }
{ "_id" : ObjectId("609b9aaccf0c3aa225ce912b"), "plate" : 21 }
...
{ "_id" : ObjectId("609b9aaccf0c3aa225ce913d"), "plate" : 39 }

MongoDB> db.taxis.find({plate:5})
{ "_id" : ObjectId("609b9aaccf0c3aa225ce911b"), "plate" : 5 }
```

```
Type "it" for more
Atlas atlas-vnfeuv-shard-0 [primary] sample_training> var c = db.taxis.find();
[
  { _id: ObjectId("61bad403013c47ef96e04d4f"), plate: 0 },
  { _id: ObjectId("61bad403013c47ef96e04d50"), plate: 1 },
  { _id: ObjectId("61bad403013c47ef96e04d51"), plate: 2 },
  { _id: ObjectId("61bad404013c47ef96e04d52"), plate: 3 },
  { _id: ObjectId("61bad404013c47ef96e04d53"), plate: 4 }
```

```
Type "it" for more
Atlas atlas-vnfeuv-shard-0 [primary] sample_training> c.hasNext()
true
Atlas atlas-vnfeuv-shard-0 [primary] sample_training> c.next()
{ _id: ObjectId("61bad408013c47ef96e04d63"), plate: 20 }
Atlas atlas-vnfeuv-shard-0 [primary] sample_training> c.next()
{ _id: ObjectId("61bad408013c47ef96e04d64"), plate: 21 }
```

# Cursor modifiers Skip and Limit and Count

Cursors can include additional instructions.

- limit
- skip
- count

Count causes the query to return just the number of results found.

```
MongoDB > for(let x=0;x<200;x++) { db.taxi.insertOne({plate:x})}

MongoDB > db.taxi.find({}).limit(5)
{ "_id" : ObjectId("609b9aaccf0c3aa225ce9116"), "plate" : 0 }
{ "_id" : ObjectId("609b9aaccf0c3aa225ce9117"), "plate" : 1 }
{ "_id" : ObjectId("609b9aaccf0c3aa225ce9118"), "plate" : 2 }
{ "_id" : ObjectId("609b9aaccf0c3aa225ce9119"), "plate" : 3 }
{ "_id" : ObjectId("609b9aaccf0c3aa225ce911a"), "plate" : 4 }

MongoDB > db.taxi.find({}).skip(2)
{ "_id" : ObjectId("609b9aaccf0c3aa225ce9118"), "plate" : 2 }
... REMOVED for clarity ...
{ "_id" : ObjectId("609b9aaccf0c3aa225ce912b"), "plate" : 21 }
Type "it" for more
MongoDB > db.taxi.find({}).skip(8).limit(2)
{ "_id" : ObjectId("609b9aaccf0c3aa225ce911e"), "plate" : 8 }
{ "_id" : ObjectId("609b9aaccf0c3aa225ce911f"), "plate" : 9 }

MongoDB > db.taxi.find({}).count()
```

```
Atlas atlas-vnfeuv-shard-0 [primary] sample_training> db.taxi.find({}).skip(20).limit(10)
[
  { "_id" : ObjectId("61bad408013c47ef96e04d63"), "plate" : 20 },
  { "_id" : ObjectId("61bad408013c47ef96e04d64"), "plate" : 21 },
  { "_id" : ObjectId("61bad408013c47ef96e04d65"), "plate" : 22 },
  { "_id" : ObjectId("61bad409013c47ef96e04d66"), "plate" : 23 },
  { "_id" : ObjectId("61bad409013c47ef96e04d67"), "plate" : 24 },
  { "_id" : ObjectId("61bad409013c47ef96e04d68"), "plate" : 25 },
  { "_id" : ObjectId("61bad409013c47ef96e04d69"), "plate" : 26 },
  { "_id" : ObjectId("61bad40a013c47ef96e04d6a"), "plate" : 27 },
  { "_id" : ObjectId("61bad40a013c47ef96e04d6b"), "plate" : 28 },
  { "_id" : ObjectId("61bad40a013c47ef96e04d6c"), "plate" : 29 }
]
```

cursor 10分鐘會被丟棄

## Cursors work in batches

- Cursors fetch results from the server in batches.
  - Fetching one by one would be slow.
  - Fetching all results at once would use too much client RAM.
- The default batch size in the shell is 101 documents during the initial call to `find()` with a limit of 16MB.
- If we fetch more than the first 100 document from a cursor it fetches in 16MB batches in the shell or up to 48MB in some drivers.

# Exercise

Add four documents to a collection called diaries using the following commands - drop removes the collection before we start. This is to make sure it's empty.

```
db.diaries.drop()
```

```
db.diaries.insertMany([
  { name: "dug", day: ISODate("2014-11-04"), txt: "went for a walk"},
  { name: "dug", day: ISODate("2014-11-06"), txt: "saw a squirrel"},
  { name: "ray", day: ISODate("2014-11-06"), txt: "met dug in the park"},
  { name: "dug", day: ISODate("2014-11-09"), txt: "got a treat"}
])
```

- Write a find() operation to output only diary entries from "dug".
- Modify it to output the line below using skip, limit and a projection

```
{ name: "dug", txt: "saw a squirrel" }
```

```
db.diaries.drop()

db.diaries.insertMany([
  { name: "dug", day: ISODate("2014-11-04"), txt: "went for a walk"},
  { name: "dug", day: ISODate("2014-11-06"), txt: "saw a squirrel"},
  { name: "ray", day: ISODate("2014-11-06"), txt: "met dug in the park"},
  { name: "dug", day: ISODate("2014-11-09"), txt: "got a treat"}
])

#找出 name = dug

#輸出 name, text
```

## 子查詢

```
# city = "New York" 即可
address.city : "New York"
#{ city: "New York" } 完整匹配
address : { city: "New York" }
```



# Querying values in nested documents

Fields in MongoDB can themselves contain documents.

To specify a field in a nested document we use dot notation.

**"address.city"**

In the shell we must put this type of field names in quotes.

```
MongoDB> db.people.insertOne( {"name": "John Doe",
  "email": "john.doe@mongodb.com",
  "address": {
    "country": "USA",
    "city": "New York",
    "zipcode": "10005"}
})

//These find our document
MongoDB> db.people.findOne({})

MongoDB> db.people.findOne({"address.city" : "New York"})

MongoDB> db.people.findOne({"name": "John Doe",
  "address.city" : "New York"})

//Shell Error - Shell think address is a variable
MongoDB> db.people.findOne({address.city : "New York"}) ✗

// This will only work if there are no other fields in the address
MongoDB> db.people.findOne({address : { city: "New York"}}) ✗
```

## 範圍查詢

### Querying by ranges of values.

MongoDB can do more than just matching by exact value.

There are operators to compare relative values like greater or less than.

We can also check an explicit set of values using `$in` - true if the value is in the list.

```
MongoDB> for(x=0;x<200;x++) { db.taxi.insertOne({plate:x})}

MongoDB> db.taxi.find({plate : { $gt : 25 }}) // >25
{ "_id" : ObjectId("609b9aaccf0c3aa225ce9130"), "plate" : 26 }
{ "_id" : ObjectId("609b9aaccf0c3aa225ce9131"), "plate" : 27 }
... REMOVED for clarity ...
{ "_id" : ObjectId("609b9aaccf0c3aa225ce9143"), "plate" : 45 }
Type "it" for more

MongoDB> db.taxi.find({plate: { $gte: 25 }}) // >= 25

MongoDB> db.taxi.find({plate: { $lt: 25 }}) // < 25

MongoDB> db.taxi.find({plate: { $gt: 25 , $lt:30 }}) // between 25 and 30

MongoDB> db.taxi.find({plate: { $ne: 3 }}) // Not 3

MongoDB> db.taxi.find({plate: { $in: [1,3,6] }}) // 1,3 or 6

MongoDB> db.taxi.find({plate: { $nin: [2,4,7] }})// Not 2,4 or 7

MongoDB> db.taxi.find({plate: { $eq: 6 }})// Same as { plate:6 }
```

## Boolean 查詢

```
$or : [ {species:"cat", color:"black"},
  {species:"dog", color:"brown"}]
#等於
```

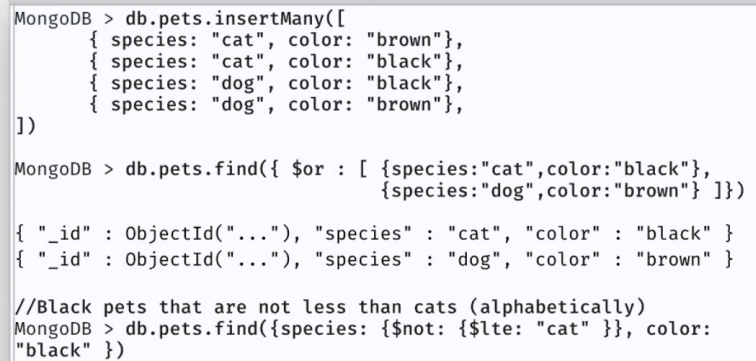
where (species = "cat" and color = "black") or (species = "dog" and color = "brown")

# Boolean Logic Operators

When required MongoDB is able to use logic like AND, OR, NOR and NOT with queries.

They can take an array as a value and can have more than two clauses.

These are normally used with complex clauses.



```
MongoDB > db.pets.insertMany([
  { species: "cat", color: "brown"},
  { species: "cat", color: "black"},
  { species: "dog", color: "black"},
  { species: "dog", color: "brown"},
])

MongoDB > db.pets.find({ $or : [ {species:"cat",color:"black"},
                                   {species:"dog",color:"brown"} ]})

{ "_id" : ObjectId("..."), "species" : "cat", "color" : "black" }
{ "_id" : ObjectId("..."), "species" : "dog", "color" : "brown" }

//Black pets that are not less than cats (alphabetically)
MongoDB > db.pets.find({species: {$not: {$lte: "cat" }}, color:
"black" })
```

## Exercise - Range and Logic

In the MongoDB shell change to using the database **sample\_training**

How many documents in the **grades** collection have a student\_id less than or equal to 65

How many documents in the **inspections** collection have **result** "Pass" or "Fail"  
(Write this in two different ways.)

```
db.grades.find({ student_id : {$lt: 65}})
```

```
db.inspections.find({$or :[ {result:"Pass"},{result:"Fail"}]})
```

🕒 修訂版本 #10

★ 由 treeman 建立於 16 🕒 2021 21:24:26

🔧 由 treeman 更新於 5 🕒 2023 10:14:59