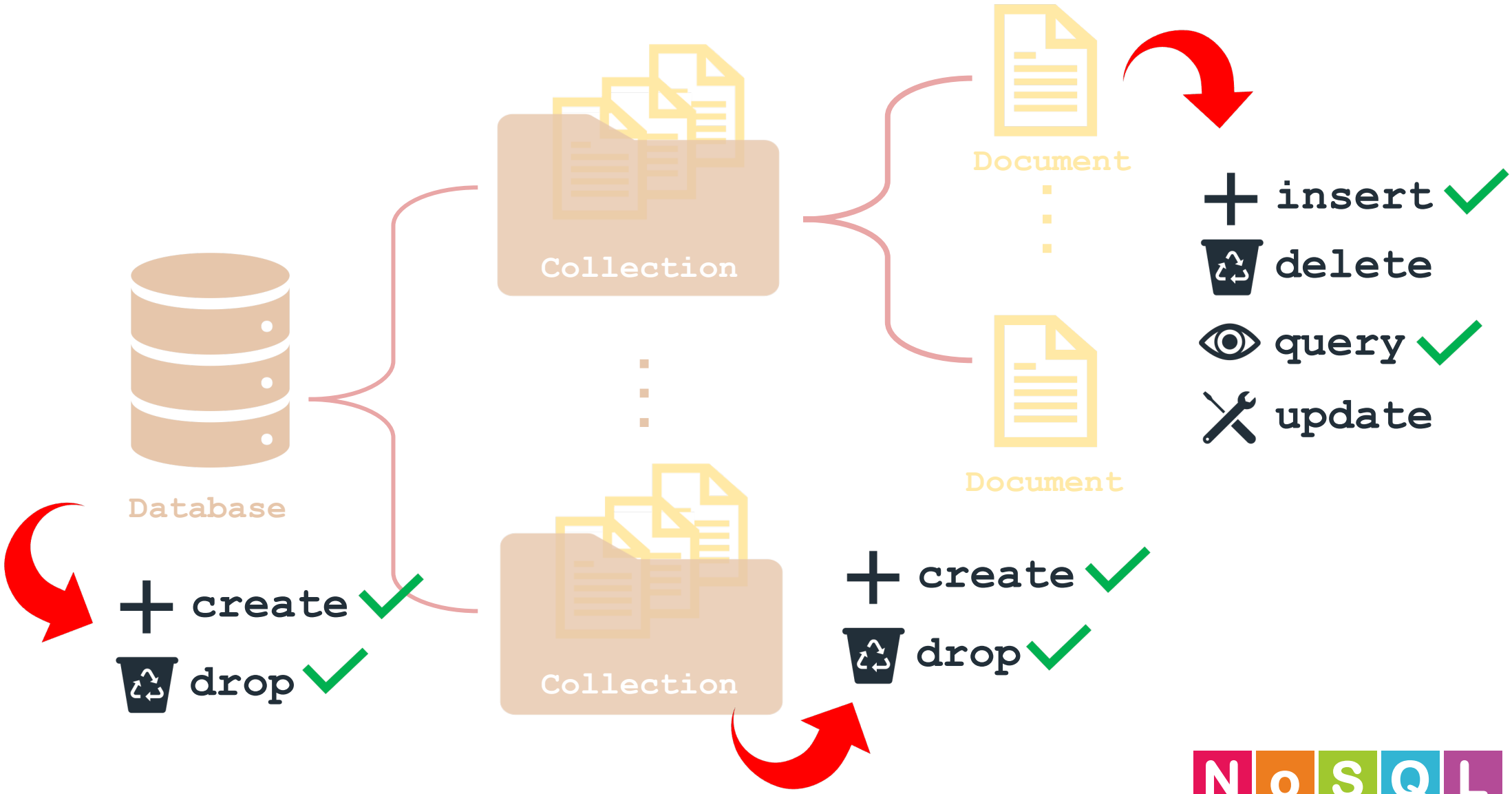
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang

Chapter 15: MongoDB Implementation (Part 3) and Its Indexes

By Asst. Prof. Dr. Taravichet Titijaroonroj

คำสั่งและตัวดำเนินการของ MongoDB



การแก้ไข Document ในฐานข้อมูล

การแก้ไข Document ในฐานข้อมูล

การแก้ไขข้อมูลลงในฐานข้อมูล MongoDB สามารถทำได้ 4 วิธีการ โดยอาศัยคำสั่งดังต่อไปนี้

1
```
db.COLLECTION_NAME.update({condition}, UPDATED_DATA)

db.COLLECTION_NAME.update({condition}, UPDATED_DATA, {multi:true})
```

2
```
db.COLLECTION_NAME.updateOne({condition},  UPDATED_DATA)
```

3
```
db.COLLECTION_NAME.updateMany({condition}, UPDATED_DATA)
```

4
```
db.COLLECTION_NAME.replaceOne({condition}, UPDATED_DATA)
```

การแก้ไข Document ในฐานข้อมูล

การแก้ไข Document ในฐานข้อมูล MongoDB จะอาศัยคำสั่ง “**update()**” เพื่อแก้ไขค่าจาก document ที่มีอยู่

```
>>> db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA)
```

กรณีที่การแก้ไขค่ามีมากกว่าหนึ่ง document จะต้องเพิ่มส่วนของ “**{multi:true}**” ต่อท้าย มิเช่นนั้น MongoDB จะแก้ไขค่าเพียง document แรกที่สอดคล้องเท่านั้น

```
>>> db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA, {multi:true})
```

ตัวอย่างที่ 1

```
>>> db.Wizard.find()
{ "_id" : ObjectId("5d199a28c745b527b7abdb02"), "sex" : "m", "name" : "Severus Snape", "school" :
"Hogwarts", "house" : "Slytherin", "pets" : [ ], "money" : 35000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a28c745b527b7abdb03"), "sex" : "f", "name" : "Minerva McGonagall", "school" :
"Hogwarts", "house" : "Gryffindor", "pets" : [ ], "money" : 50000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb05"), "sex" : "m", "name" : "Quirinus Quirrell", "school" :
"Hogwarts", "house" : "Ravenclaw", "pets" : [ ], "money" : 5000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb06"), "sex" : "m", "name" : "Albus Dumbledore", "school" :
"Hogwarts", "house" : "Gryffindor", "pets" : [ "Phoenix" ], "money" : 95000, "position" : "teacher" }

>>> db.Wizard.update({"name":"Severus Snape"},{$set:{"money":42000}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

>>> db.Wizard.find()
{ "_id" : ObjectId("5d199a28c745b527b7abdb02"), "sex" : "m", "name" : "Severus Snape", "school" :
"Hogwarts", "house" : "Slytherin", "pets" : [ ], "money" : 42000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a28c745b527b7abdb03"), "sex" : "f", "name" : "Minerva McGonagall", "school" :
"Hogwarts", "house" : "Gryffindor", "pets" : [ ], "money" : 50000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb05"), "sex" : "m", "name" : "Quirinus Quirrell", "school" :
"Hogwarts", "house" : "Ravenclaw", "pets" : [ ], "money" : 5000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb06"), "sex" : "m", "name" : "Albus Dumbledore", "school" :
"Hogwarts", "house" : "Gryffindor", "pets" : [ "Phoenix" ], "money" : 95000, "position" : "teacher" }
```

ตัวอย่างที่ 2

```
>>> db.Wizard.update({"sex":"m"},{$set:{"sex":"male"}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

>>> db.Wizard.find()
{ "_id" : ObjectId("5d199a28c745b527b7abdb02"), "sex" : "male", "name" : "Severus Snape", "school" : "Hogwarts",
"house" : "Slytherin", "pets" : [ ], "money" : 42000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb04"), "sex" : "f", "name" : "Sybill Trelawney", "school" : "Hogwarts",
"house" : "Ravenclaw", "pets" : [ ], "money" : 15000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb05"), "sex" : "m", "name" : "Quirinus Quirrell", "school" : "Hogwarts",
"house" : "Ravenclaw", "pets" : [ ], "money" : 5000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb06"), "sex" : "m", "name" : "Albus Dumbledore", "school" : "Hogwarts",
"house" : "Gryffindor", "pets" : [ "Phoenix" ], "money" : 95000, "position" : "teacher" }

>>> db.Wizard.update({"sex":"m"},{$set:{"sex":"male"}},{multi:true})
WriteResult({ "nMatched" : 9, "nUpserted" : 0, "nModified" : 9 })

>>> db.Wizard.find()
{ "_id" : ObjectId("5d199a28c745b527b7abdb02"), "sex" : "male", "name" : "Severus Snape", "school" : "Hogwarts",
"house" : "Slytherin", "pets" : [ ], "money" : 42000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a28c745b527b7abdb03"), "sex" : "f", "name" : "Minerva McGonagall", "school" :
"Hogwarts", "house" : "Gryffindor", "pets" : [ ], "money" : 50000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb04"), "sex" : "f", "name" : "Sybill Trelawney", "school" : "Hogwarts",
"house" : "Ravenclaw", "pets" : [ ], "money" : 15000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb05"), "sex" : "male", "name" : "Quirinus Quirrell", "school" :
"Hogwarts", "house" : "Ravenclaw", "pets" : [ ], "money" : 5000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb06"), "sex" : "male", "name" : "Albus Dumbledore", "school" :
"Hogwarts", "house" : "Gryffindor", "pets" : [ "Phoenix" ], "money" : 95000, "position" : "teacher" }
```

ตัวอย่างที่ 3

```
>>> db.Wizard.find()
{ "_id" : ObjectId("5d199a28c745b527b7abdb02"), "sex" : "male", "name" : "Severus Snape", "school" : "Hogwarts",
"house" : "Slytherin", "pets" : [ ], "money" : 42000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb04"), "sex" : "f", "name" : "Sybill Trelawney", "school" : "Hogwarts",
"house" : "Ravenclaw", "pets" : [ ], "money" : 15000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb05"), "sex" : "m", "name" : "Quirinus Quirrell", "school" : "Hogwarts",
"house" : "Ravenclaw", "pets" : [ ], "money" : 5000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb06"), "sex" : "m", "name" : "Albus Dumbledore", "school" : "Hogwarts",
"house" : "Gryffindor", "pets" : [ "Phoenix" ], "money" : 95000, "position" : "teacher" }

>>> db.Wizard.update({"name":"Severus Snape"},{$set:{"sex":"m","money":43500}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

>>> db.Wizard.find()
{ "_id" : ObjectId("5d199a28c745b527b7abdb02"), "sex" : "m", "name" : "Severus Snape", "school" : "Hogwarts",
"house" : "Slytherin", "pets" : [ ], "money" : 43500, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb04"), "sex" : "f", "name" : "Sybill Trelawney", "school" : "Hogwarts",
"house" : "Ravenclaw", "pets" : [ ], "money" : 15000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb05"), "sex" : "male", "name" : "Quirinus Quirrell", "school" :
"Hogwarts", "house" : "Ravenclaw", "pets" : [ ], "money" : 5000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb06"), "sex" : "male", "name" : "Albus Dumbledore", "school" :
"Hogwarts", "house" : "Gryffindor", "pets" : [ "Phoenix" ], "money" : 95000, "position" : "teacher" }
```

การแก้ไข Document ในฐานข้อมูล

นอกจากนี้ คำสั่ง **updateOne()** จะเทียบเคียงได้กับคำสั่ง **update()** แบบ**ไม่มี**การกำหนด **{multi:true}** หรือกล่าวคือการกำหนดเป็น **{multi:false}**

```
>>> db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA,{multi:false})
```

```
>>> db.COLLECTION_NAME.updateOne(SELECTION_CRITERIA, UPDATED_DATA)
```

ขณะที่ คำสั่ง **updateMany()** จะเทียบเคียงได้กับคำสั่ง **update()** แบบ**มี**การกำหนด **{multi:true}**

```
>>> db.COLLECTION_NAME.update(SELECTION_CRITERIA, UPDATED_DATA, {multi:true})
```

```
>>> db.COLLECTION_NAME.updateMany(SELECTION_CRITERIA, UPDATED_DATA)
```

ตัวดำเนินการที่ใช้กับการแก้ไขข้อมูล

ตัวดำเนินการ	คำอธิบาย
$set	กำหนดค่าให้กับแอททริบิวท์ (หรือคีย์) ที่ระบุ หรือ สร้างใหม่กรณีที่ไม่มี
$inc	การเพิ่มค่าด้วยตัวเลขที่กำหนด
$min	การแก้ไขค่าก็ต่อเมื่อค่าที่ระบุมีค่าน้อยกว่าค่าที่มีอยู่
$max	การแก้ไขค่าก็ต่อเมื่อค่าที่ระบุมีค่ามากกว่าค่าที่มีอยู่
$rename	เปลี่ยนชื่อแอททริบิวท์ (หรือคีย์)
$unset	ลบแอททริบิวท์ (หรือคีย์) ที่ระบุออกจาก Document นั้น

ตัวดำเนินการที่ใช้กับการแก้ไขข้อมูล

การใช้งานตัวดำเนินการ **$inc** ร่วมกับคำสั่ง **update(), updateOne(),** หรือ **updateMany()** สามารถเขียนในรูปแบบดังต่อไปนี้ โดยที่ **value** ต้องเป็นตัวเลขจำนวนบวกหรือลบก็ได้

```
>>> db.COLLECTION_NAME.update(SELECTION_CRITERIA, {$inc:{key 1 :value 1, …, ken N:value N}})
```

ตัวอย่างการใช้งาน **$inc**

```
>>> db.Product.find()
{ _id: 1, quantity: 10, metrics: { orders: 2, ratings: 3.5 } }

>>> db.Products.update( { "_id": 1 }, { $inc: { "quantity": -2, "metrics.orders": 1 } })

>>> db.Product.find()
{ _id: 1, quantity: 8, metrics: { orders: 3, ratings: 3.5 } }
```

ตัวดำเนินการที่ใช้กับการแก้ไขข้อมูล

การใช้งานตัวดำเนินการ **$min** ร่วมกับคำสั่ง **update(), updateOne(),** หรือ **updateMany()** สามารถเขียนในรูปแบบดังต่อไปนี้

```
>>> db.COLLECTION_NAME.update(SELECTION_CRITERIA, {$min:{"key 1" :value 1, …, "key N":value N}})
```

ตัวอย่างการใช้งาน **$min**

```
>>> db.Score.find()
{ _id: 1, highScore: 800, lowScore: 200 }
>>> db.scores.update( { _id: 1 }, { $min: { lowScore: 150 } })
>>> db.Product.find()
{ _id: 1, highScore: 800, lowScore: 150 }

>>> db.scores.update( { _id: 1 }, { $min: { lowScore: 250 } })
>>> db.Product.find()
{ _id: 1, highScore: 800, lowScore: 150 }
```

ตัวดำเนินการที่ใช้กับการแก้ไขข้อมูล

การใช้งานตัวดำเนินการ **$max** ร่วมกับคำสั่ง **update(), updateOne(),** หรือ **updateMany()** สามารถเขียนในรูปแบบดังต่อไปนี้

```
>>> db.COLLECTION_NAME.update(SELECTION_CRITERIA, {$max:{"key 1" :value 1, …, "key N":value N}})
```

ตัวอย่างการใช้งาน **$max**

```
>>> db.Score.find()
{ _id: 1, highScore: 800, lowScore: 200 }
>>> db.scores.update( { _id: 1 }, { $max: { highScore: 950 } } )
>>> db.Product.find()
{ _id: 1, highScore: 950, lowScore: 220 }

>>> db.scores.update( { _id: 1 }, { $max: { highScore: 870 } } )
>>> db.Product.find()
{ _id: 1, highScore: 950, lowScore: 220 }
```

ตัวดำเนินการที่ใช้กับการแก้ไขข้อมูล

การใช้งานตัวดำเนินการ **$rename** ร่วมกับคำสั่ง **update(), updateOne(),** หรือ **updateMany()** สามารถเขียนในรูปแบบดังต่อไปนี้

```
>>> db.COLLECTION_NAME.update(SELECTION_CRITERIA, {$rename:{"key 1" :value 1, …, "key N":value N}})
```

ตัวอย่างการใช้งาน **$rename**

```
>>> db.Score.find()
{ _id: 1, highScore: 800, lowScore: 200 }

>>> db.scores.update( { "_id": 1 }, { $rename: { "highScore": "hScore" } } )

>>> db.Product.find()
{ _id: 1, hScore: 950, lowScore: 220 }
```

ตัวดำเนินการที่ใช้กับการแก้ไขข้อมูล

การใช้งานตัวดำเนินการ **$unset** ร่วมกับคำสั่ง **update(), updateOne(),** หรือ **updateMany()** สามารถเขียนในรูปแบบดังต่อไปนี้

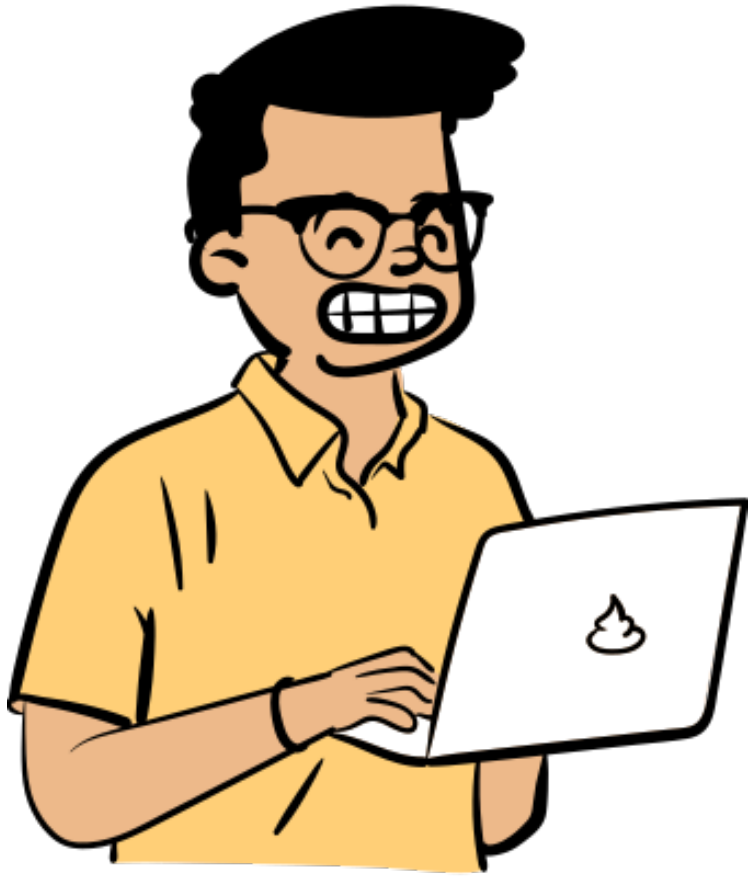
```
>>> db.COLLECTION_NAME.update(SELECTION_CRITERIA, {$unset:{"key 1" : "", …, "key N": ""}})
```

ตัวอย่างการใช้งาน **$unset**

```
>>> db.items.find()
{ "_id" : 1, "description" : "item1", "op_stock" : 100, "purqty" : 100 }

>>> db.items.update( { "_id": 1 },{ $unset: {"purqty": ""}})

>>> db.items.find()
{ "_id" : 1, "description" : "item1", "op_stock" : 100 }
```



คำสั่ง update(), updateOne() และ updateMany() จะเป็นการค้นหา Document ที่สอดคล้องกับเงื่อนไขแล้ว*ทำการแก้ไขข้อมูลใน Document* ดังกล่าวตามที่กำหนด

ขณะที่ replaceOne() คือ คำสั่งที่จะ*แทนที่ Document ที่สอดคล้องกับเงื่อนไขตัวแรกด้วย Document ใหม่* กล่าวคือการลบ Document เก่าและทำการเพิ่ม Document อันใหม่

การแก้ไข Document ในฐานข้อมูล

นอกจากนี้ คำสั่ง **replaceOne()** จะแทนที่ Document ที่สอดคล้องกับเงื่อนไข**ตัวแรกด้วย Document ใหม่**

```
>>> db.COLLECTION_NAME.replaceOne(SELECTION_CRITERIA, UPDATED_DATA)
```

ตัวอย่างเช่น

```
>>> db.product.find()
{"_id":1, "name":"Car",  "amount": 10,"price":450, "color":["red", "blue"]}
{"_id":2, "name":"Doll", "amount": 5, "price":120, "size":["S", "M", "L"]}
{"_id":3, "name":"Robot","amount": 7, "price":1050,"version":["G", "I"]}

>>> db.product.replaceOne({name: "Doll"},{"name": "iDoll", "amount":2,"price":220})

>>> db.product.find()
{"_id":1, "name":"Car",  "amount": 10,"price":450, "color":["red", "blue"]}
{"_id":2, "name": "iDoll", "amount":2,"price":220}
{"_id":3, "name":"Robot","amount": 7, "price":1050,"version":["G", "I"]}
```

ข้อสังเกตุ

- คำสั่ง update(), updateOne() และ updateMany() จะเป็นการค้นหา Document ที่สอดคล้องกับเงื่อนไขแล้วทำการแก้ไขข้อมูลใน Document ดังกล่าวตามที่กำหนด *แต่ถ้า Document นั้นไม่มีหรือไม่พบ* Attribute หรือ Key ที่ต้องการให้แก้ไขค่า จะถือว่าเป็นการ*แทรก key และ value ใหม่*เพิ่มเข้าไปใน Document นั้นเลย
  - ถ้าส่วน `{condition}` ของคำสั่ง `update()`, `updateOne()`, และ `updateMany()` เป็นค่า `{}` จะมีความหมายว่า
    - `update({},document, {multi:false})` เลือก Document แรก
    - `update({},documents, {multi:true})` เลือกทุก Document
    - `updateOne({},document)` เลือก Document แรก
    - `updateMany({},documents)` เลือกทุก Document
    - `replaceOne({},document)` เลือก Document แรก

การลบ Document ในฐานข้อมูล

การลบ Document ในฐานข้อมูล

การลบ Document ในฐานข้อมูล MongoDB จะอาศัยคำสั่ง “remove()” ที่ประกอบด้วยพารามิเตอร์หลัก 2 ตัว ได้แก่ “DELLETION_CRITTERIA” และ “JUSTONE”

```
>>> db.COLLECTION_NAME.remove(DELLETION_CRITTERIA, JUSTONE)
```

โดยที่ DELLETION_CRITTERIA คือ ส่วนกำหนดเงื่อนไขเพื่อระบุว่าจะลบ document ใดบ้าง ขณะที่ JUSTONE ใช้สำหรับกำหนดว่าจะลบเพียง document เดียวสำหรับกรณีมีการลบมากกว่าหนึ่ง document

ตัวอย่างที่ 1

```
>>> db.Wizard.find()
{ "_id" : ObjectId("5d199a28c745b527b7abdb02"), "sex" : "m", "name" : "Severus Snape", "school" : "Hogwarts",
"house" : "Slytherin", "pets" : [ ], "money" : 43500, "position" : "teacher" }
...
{ "_id" : ObjectId("5d199a29c745b527b7abdb12"), "sex" : "f", "name" : "Fleur Delacour", "school" : "Beauxbatons",
"house" : "", "pets" : [ ], "money" : 9000, "position" : "student" }
{ "_id" : ObjectId("5d1c190a8d149be840886137"), "sex" : "f" }

>>> db.Wizard.remove({"_id" : ObjectId("5d1c190a8d149be840886137")})
WriteResult({ "nRemoved" : 1 })

>>> db.Wizard.find()
{ "_id" : ObjectId("5d199a28c745b527b7abdb02"), "sex" : "m", "name" : "Severus Snape", "school" : "Hogwarts",
"house" : "Slytherin", "pets" : [ ], "money" : 43500, "position" : "teacher" }
...
{ "_id" : ObjectId("5d199a29c745b527b7abdb12"), "sex" : "f", "name" : "Fleur Delacour", "school" : "Beauxbatons",
"house" : "", "pets" : [ ], "money" : 9000, "position" : "student" }
```

ตัวอย่างที่ 2

```
>>> db.Wizard.find()
{ "_id" : ObjectId("5d199a28c745b527b7abdb03"), "sex" : "f", "name" : "Minerva McGonagall", "school" :
"Hogwarts", "house" : "Gryffindor", "pets" : [ ], "money" : 50000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb04"), "sex" : "f", "name" : "Sybill Trelawney", "school" : "Hogwarts",
"house" : "Ravenclaw", "pets" : [ ], "money" : 15000, "position" : "teacher" }
{ "_id" : ObjectId("5d1c2e978d149be840886139"), "sex" : "m", "name" : "Severus Snape", "school" : "Hogwarts",
"house" : "Slytherin", "pets" : [ ], "money" : 35000, "position" : "teacher" }
{ "_id" : ObjectId("5d1c2e9a8d149be84088613a"), "sex" : "m", "name" : "Severus Snape", "school" : "Hogwarts",
"house" : "Slytherin", "pets" : [ ], "money" : 35000, "position" : "student" }

>>> db.Wizard.remove({"name":"Severus Snape"})
WriteResult({ "nRemoved" : 2 })

>>> db.Wizard.find()
{ "_id" : ObjectId("5d199a28c745b527b7abdb03"), "sex" : "f", "name" : "Minerva McGonagall", "school" :
"Hogwarts", "house" : "Gryffindor", "pets" : [ ], "money" : 50000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb04"), "sex" : "f", "name" : "Sybill Trelawney", "school" : "Hogwarts",
"house" : "Ravenclaw", "pets" : [ ], "money" : 15000, "position" : "teacher" }
```

ตัวอย่างที่ 2 (ต่อ)

```
>>> db.Wizard.find()
{ "_id" : ObjectId("5d199a28c745b527b7abdb03"), "sex" : "f", "name" : "Minerva McGonagall", "school" :
"Hogwarts", "house" : "Gryffindor", "pets" : [ ], "money" : 50000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb04"), "sex" : "f", "name" : "Sybill Trelawney", "school" : "Hogwarts",
"house" : "Ravenclaw", "pets" : [ ], "money" : 15000, "position" : "teacher" }
{ "_id" : ObjectId("5d1c2e978d149be840886139"), "sex" : "m", "name" : "Severus Snape", "school" : "Hogwarts",
"house" : "Slytherin", "pets" : [ ], "money" : 35000, "position" : "teacher" }
{ "_id" : ObjectId("5d1c2e9a8d149be84088613a"), "sex" : "m", "name" : "Severus Snape", "school" : "Hogwarts",
"house" : "Slytherin", "pets" : [ ], "money" : 35000, "position" : "student" }

>>> db.Wizard.remove({"name":"Severus Snape"},1) // or >>> db.Wizard.remove({"name":"Severus Snape"},true)
WriteResult({ "nRemoved" : 1 })

>>> db.Wizard.find()
{ "_id" : ObjectId("5d199a28c745b527b7abdb03"), "sex" : "f", "name" : "Minerva McGonagall", "school" :
"Hogwarts", "house" : "Gryffindor", "pets" : [ ], "money" : 50000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb04"), "sex" : "f", "name" : "Sybill Trelawney", "school" : "Hogwarts",
"house" : "Ravenclaw", "pets" : [ ], "money" : 15000, "position" : "teacher" }
{ "_id" : ObjectId("5d1c2e9a8d149be84088613a"), "sex" : "m", "name" : "Severus Snape", "school" : "Hogwarts",
"house" : "Slytherin", "pets" : [ ], "money" : 35000, "position" : "student" }
```


ข้อควรระวัง ใน MongoDB ค่า 0 ส่วนมากมีความหมายว่า false และเลขอื่น ๆ มีความหมายว่า true

ตัวอย่างที่ 3

```
>>> db.Wizard.find()
{ "_id" : ObjectId("5d199a28c745b527b7abdb03"), "sex" : "f", "name" : "Minerva McGonagall", "school" :
"Hogwarts", "house" : "Gryffindor", "pets" : [ ], "money" : 50000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb04"), "sex" : "f", "name" : "Sybill Trelawney", "school" : "Hogwarts",
"house" : "Ravenclaw", "pets" : [ ], "money" : 15000, "position" : "teacher" }
{ "_id" : ObjectId("5d1c2e978d149be840886139"), "sex" : "m", "name" : "Severus Snape", "school" : "Hogwarts",
"house" : "Slytherin", "pets" : [ ], "money" : 35000, "position" : "teacher" }
{ "_id" : ObjectId("5d1c2e9a8d149be84088613a"), "sex" : "m", "name" : "Severus Snape", "school" : "Hogwarts",
"house" : "Slytherin", "pets" : [ ], "money" : 35000, "position" : "student" }

>>> db.Wizard.remove({$and:[{"name":"Severus Snape"},{"position":"student"}]},true)
WriteResult({ "nRemoved" : 1 })

>>> db.Wizard.find()
{ "_id" : ObjectId("5d199a28c745b527b7abdb03"), "sex" : "f", "name" : "Minerva McGonagall", "school" :
"Hogwarts", "house" : "Gryffindor", "pets" : [ ], "money" : 50000, "position" : "teacher" }
{ "_id" : ObjectId("5d199a29c745b527b7abdb04"), "sex" : "f", "name" : "Sybill Trelawney", "school" : "Hogwarts",
"house" : "Ravenclaw", "pets" : [ ], "money" : 15000, "position" : "teacher" }
{ "_id" : ObjectId("5d1c2f858d149be84088613d"), "sex" : "m", "name" : "Severus Snape", "school" : "Hogwarts",
"house" : "Slytherin", "pets" : [ ], "money" : 35000, "position" : "teacher" }
```

การลบ Document ในฐานข้อมูล

นอกจากนี้ การลบ Document ในฐานข้อมูล MongoDB ยังสามารถใช้คำสั่ง “**deleteOne()**” และ “**deleteMany()**” ที่มีเพียง “**DELLETION_CRITTERIA**” เป็นพารามิเตอร์หลักตัวเดียว

```
>>> db.COLLECTION_NAME.deleteOne(DELLETION_CRITTERIA)

>>> db.COLLECTION_NAME.deleteMany(DELLETION_CRITTERIA)
```

โดยที่

- คำสั่ง **deleteOne(DELLETION_CRITTERIA)** เทียบเคียงได้กับ **remove(DELLETION_CRITTERIA,true)**
- คำสั่ง **deleteMany(DELLETION_CRITTERIA)** เทียบเคียงได้กับ **remove(DELLETION_CRITTERIA)**

อย่างไรก็ตาม คำสั่ง **remove()** จะคืนค่ามาเป็นวัตถุ WriteResult ที่จัดเก็บสถานะของการดำเนินการ ขณะที่คำสั่ง **deleteOne()** และ **deleteMany()** จะคืนค่ามาเป็น boolean เพื่อบ่งบอกว่าดำเนินการสำเร็จหรือไม่ และค่า deletedCount ที่บ่งบอกว่าลบไปกี่ document

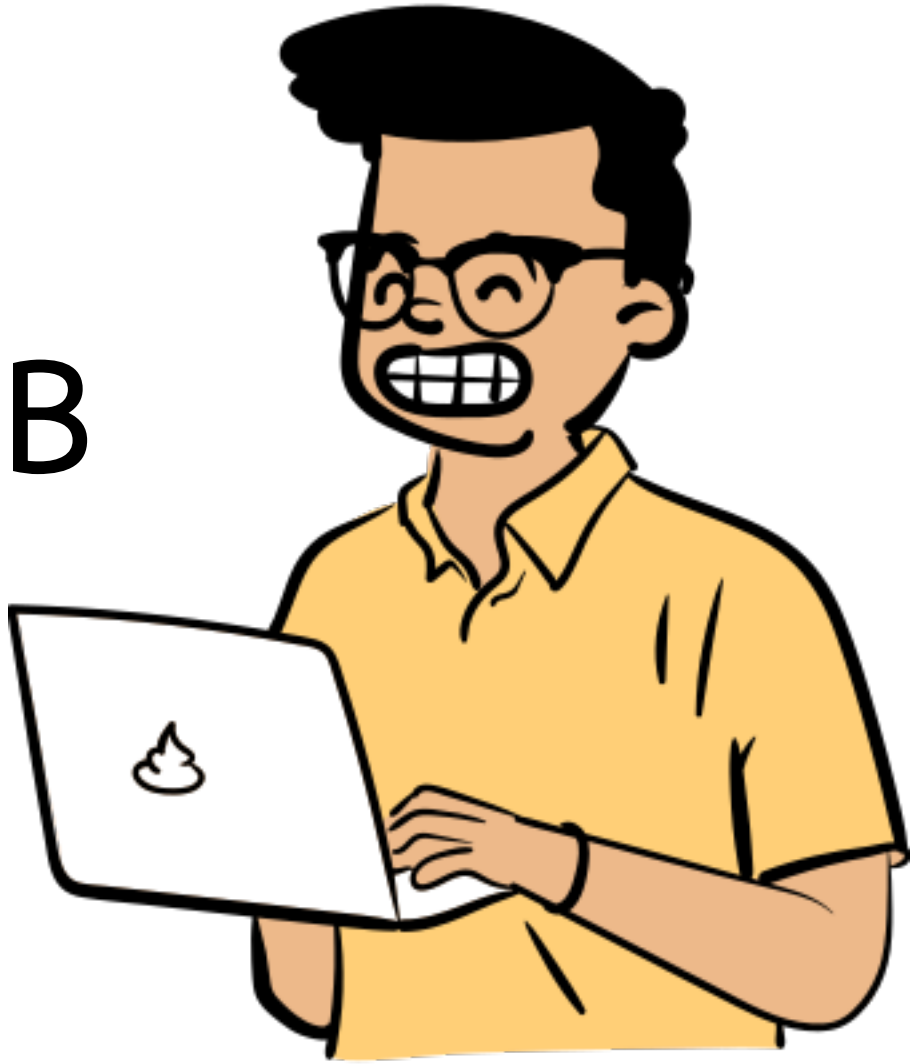
ถ้าส่วน `{condition}` ของคำสั่ง `remove()`, `deleteOne()` และ `deleteMany()` เป็นค่า `{}` จะมีความหมายว่า

- `deleteOne({})` ลบ Document แรก
- `deleteMany({})` ลบทุก Document

เช่นเดียวกับคำสั่ง `remove()`

- `remove({},true)` ลบ Document แรก
- `remove({})` ลบทุก Document

การใช้งาน Indexs ในฐานข้อมูล MongoDB



ตัวอย่างข้อมูลชุด Book

```
{"_id":1, "name":"A1", "category": "art", "price":1050}
{"_id":2, "name":"A2", "category": "cook","price":450}
{"_id":3, "name":"A3", "category": "cook","price":560}
{"_id":4, "name":"A4", "category": "biz", "price":230}
{"_id":5, "name":"A5", "category": "art", "price":850}
{"_id":6, "name":"A6", "category": "biz", "price":420}
{"_id":7, "name":"A7", "category": "art", "price":980}
{"_id":8, "name":"A8", "category": "cook","price":500}
{"_id":9, "name":"A9", "category": "biz", "price":350}
{"_id":10, "name":"A10", "category": "biz","price":250}
```

การสร้าง Index ในฐานข้อมูล

การสร้าง Index ใน MongoDB จะช่วยเพิ่มประสิทธิภาพในการค้นหา Collection ที่สอดคล้องได้เร็วยิ่งขึ้น โดย*ค้นหาแบบเรียงจากน้อยไปมาก (ค่าเริ่มต้นกรณีไม่กำหนด) หรือมากไปน้อย*

https://docs.mongodb.com/manual/reference/method/db.collection.createIndex/#ensureindex-options

```
>>> db.COLLECTION_NAME.createIndex({KEY:1 or -1}, {unique:true})
```

ตัวอย่างเช่น

```
>>> db.Book.createIndex({category: 1})                    // 1 แทน ASC และ -1 แทน DESC
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,                           // จำนวน Index เดิม
        "numIndexsAfter" : 2,                             // จำนวน Index ปัจจุบัน
        "ok" : 1
}
```

การแสดงรายชื่อ Index ในฐานข้อมูล

การแสดงรายชื่ออาร์เรย์ของ Index ใน Collection นั้นของ MongoDB สามารถทำได้ โดยอาศัยคำสั่งต่อไปนี้

```
>>> db.COLLECTION_NAME.getIndexes()
```

ตัวอย่างเช่น

```
>>> db.Book.getIndexes()
[{
                "v" : 2,
                "key" : {"_id" : 1},          // ชื่อ index ตาม Collection
                "name" : "_id_"               // สิ่งที่ต้องใช้ตอนลบ
        },{
                "v" : 2,
                "key" : {"category" : 1},
                "name" : "category_1"
}]
```

การลบ Index ในฐานข้อมูล

การลบ Index ใน Collection ของ MongoDB สามารถทำได้จากคำสั่ง dropIndex(name) โดยที่ name คือ **ค่าของคีย์ name ที่ได้จากคำสั่ง getIndexes()**

```
>>> db.COLLECTION_NAME.dropIndex(name)
```

ตัวอย่างเช่น

```
>>>  db.Book.dropIndex("category_1")
{ "nIndexesWas" : 2, "ok" : 1 }
```

การเปรียบเทียบประสิทธิภาพด้านเวลาระหว่างไม่มี Index กับมี

การเปรียบเทียบประสิทธิภาพด้านเวลาระหว่าง find() แบบไม่มีการสร้าง Index ใน Collection กับมีการสร้าง Index ว่ามีความแตกต่างกันอย่างไร โดยอาศัยคำสั่ง explain(“executionStats”) ต่อท้ายจากคำสั่งประเภท find()

```
>>> db.COLLECTION_NAME.find().explain(“executionStats”)
```

ตัวอย่างเช่น

```
>>> db.Book.find({category: “cook”}).explain(“executionStats”)
```

นอกจากนี้ ยังสามารถใช้วิเคราะห์ประสิทธิภาพกับคำสั่ง count() หรือ aggregate() ได้ด้วย

(1)
ข้อมูล
ก่อน
การ
สร้าง
index

Command Prompt - mongo

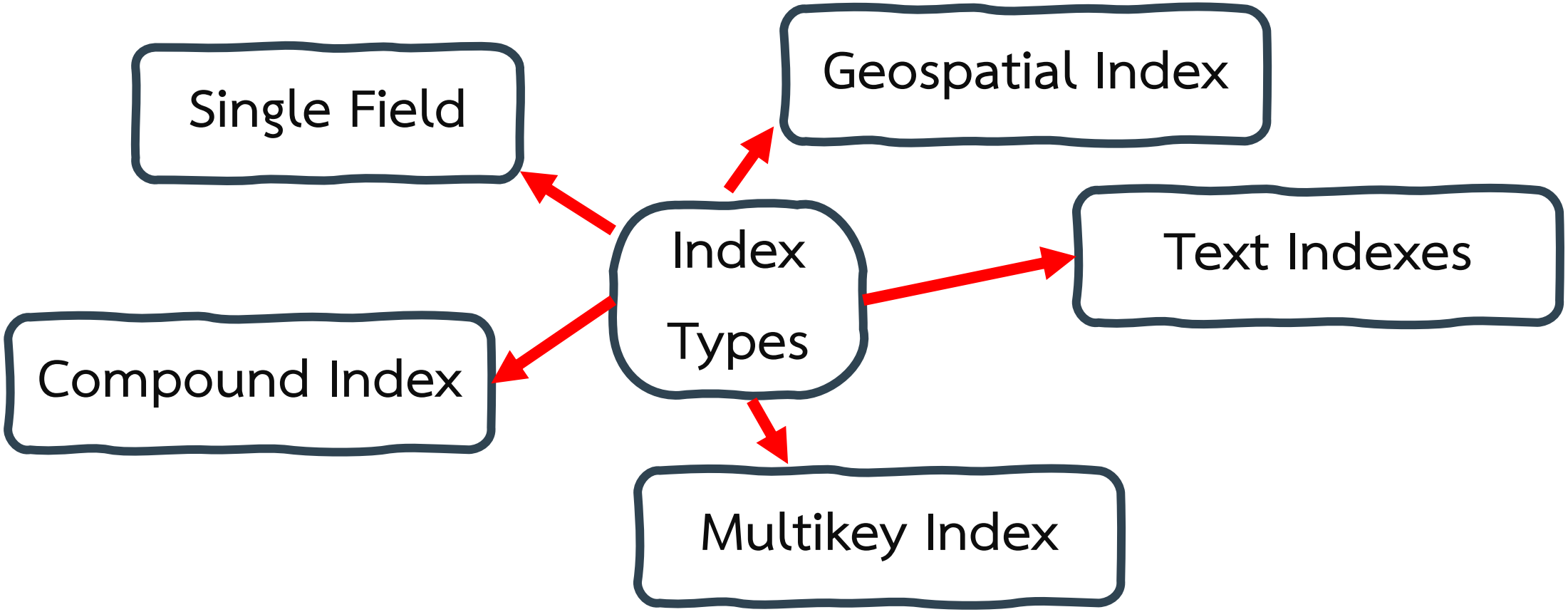
```
{ "nIndexesWas" : 2, "ok" : 1 }
> db.Book.find({category: "cook"}).explain("executionStats")
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "Lib.Book",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "category" : {
                                "$eq" : "cook"
                        }
                },
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "category" : {
                                        "$eq" : "cook"
                                }
                        },
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 3,
                "executionTimeMillis" : 0,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 10,
                "executionStages" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "category" : {
                                        "$eq" : "cook"
                                }
                        },
                        "nReturned" : 3,
                        "executionTimeMillisEstimate" : 0,
                        "works" : 12,
                        "advanced" : 3,
                        "needTime" : 8,
                        "needYield" : 0,
                        "saveState" : 0,
                        "restoreState" : 0,
                        "isEOF" : 1,
                        "direction" : "forward",
                        "docsExamined" : 10
                }
        },
        "serverInfo" : {
                "host" : "DESKTOP-UDLH4VM",
                "port" : 27017,
                "version" : "4.4.1",
                "gitVersion" : "ad91a93a5a31e175f5cbf8c69561e78
```

(2)
ข้อมูล
หลัง
การ
สร้าง
index

Command Prompt - mongo

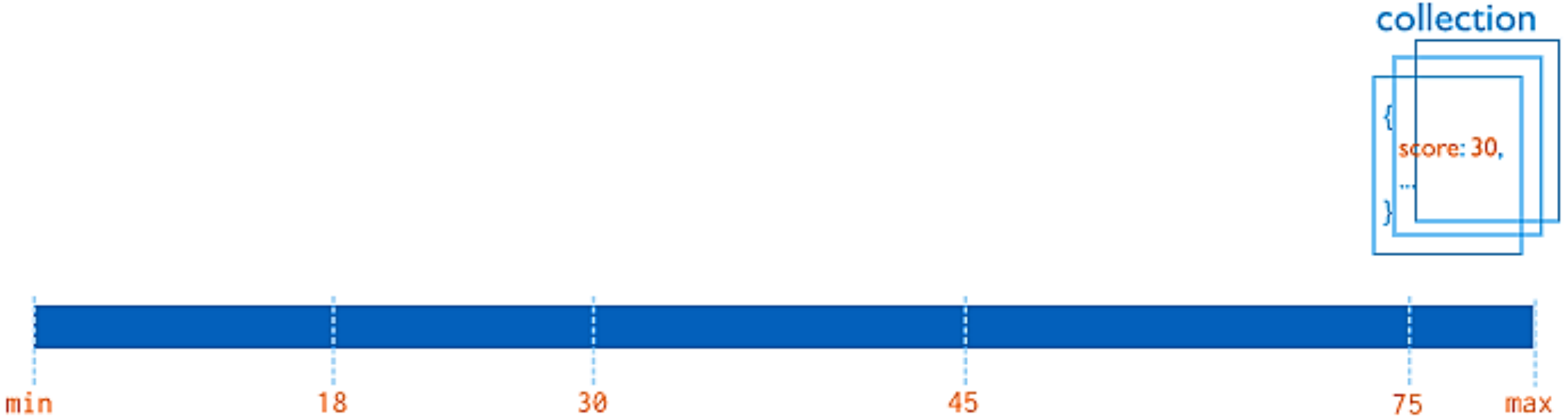
```
> db.Book.find({category: "cook"}).explain("executionStats")
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "Lib.Book",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "category" : {
                                "$eq" : "cook"
                        }
                },
                "winningPlan" : {
                        "stage" : "FETCH",
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "keyPattern" : {
                                        "category" : 1
                                },
                                "indexName" : "category_1",
                                "isMultiKey" : false,
                                "multiKeyPaths" : {
                                        "category" : [ ]
                                },
                                "isUnique" : false,
                                "isSparse" : false,
                                "isPartial" : false,
                                "indexVersion" : 2,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "category" : [
                                                "[\"cook\", \"cook\"]"
                                        ]
                                }
                        }
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 3,
                "executionTimeMillis" : 7,
                "totalKeysExamined" : 3,
                "totalDocsExamined" : 3,
                "executionStages" : {
                        "stage" : "FETCH",
                        "nReturned" : 3,
                        "executionTimeMillisEstimate" : 0,
                        "works" : 4,
                        "advanced" : 3,
                        "needTime" : 0,
                        "needYield" : 0,
                        "saveState" : 0,
                        "restoreState" : 0,
                        "isEOF" : 1,
                        "docsExamined" : 3,
                        "alreadyHasObj" : 0,
```

Indexes



MongoDB มี index หลากหลายรูปแบบ เพื่อที่จะได้รองรับกับชนิดข้อมูลหรือ Query ที่หลากหลาย

Single Field



MongoDB รองรับการกำหนดคีย์จากฟิลด์ของ Document ซึ่งกำหนดโดยผู้ใช้งานได้

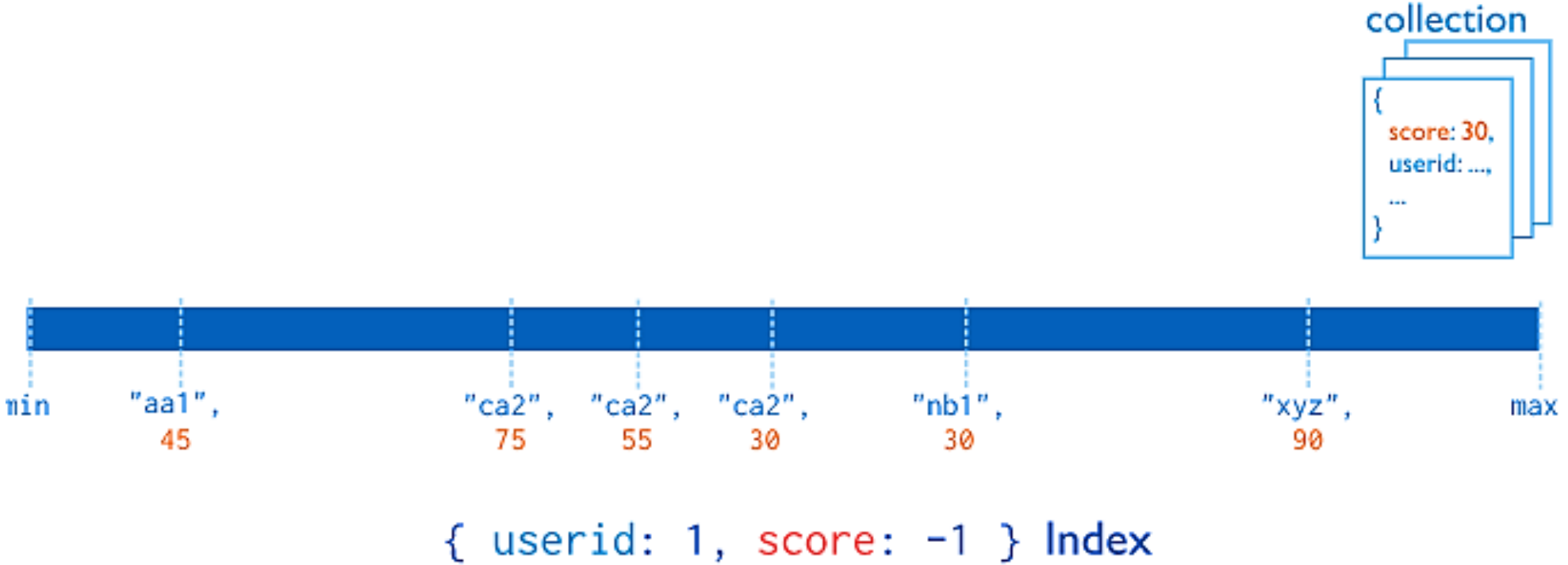
การสร้าง Single Index

```
db.COLLECTION_NAME.createIndex({KEY1: 1 หรือ -1})
```

ตัวอย่างเช่น

```
>>> db.Book.createIndex({category: 1})                    // 1 แทน ASC และ -1 แทน DESC
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,                           // จำนวน Index เดิม
        "numIndexsAfter" : 2,                             // จำนวน Index ปัจจุบัน
        "ok" : 1
}
```

Compound Index



MongoDB รองรับการสร้างคีย์จากข้อมูลหลากหลายฟิลด์ หรือสามารถเรียกว่า Compound Index สำหรับการเรียงข้อมูลที่เกิดจากหลากหลายฟิลด์ ผู้ใช้งานสามารถกำหนดลำดับและรูปแบบการจัดเรียงได้

การสร้าง Compound Index

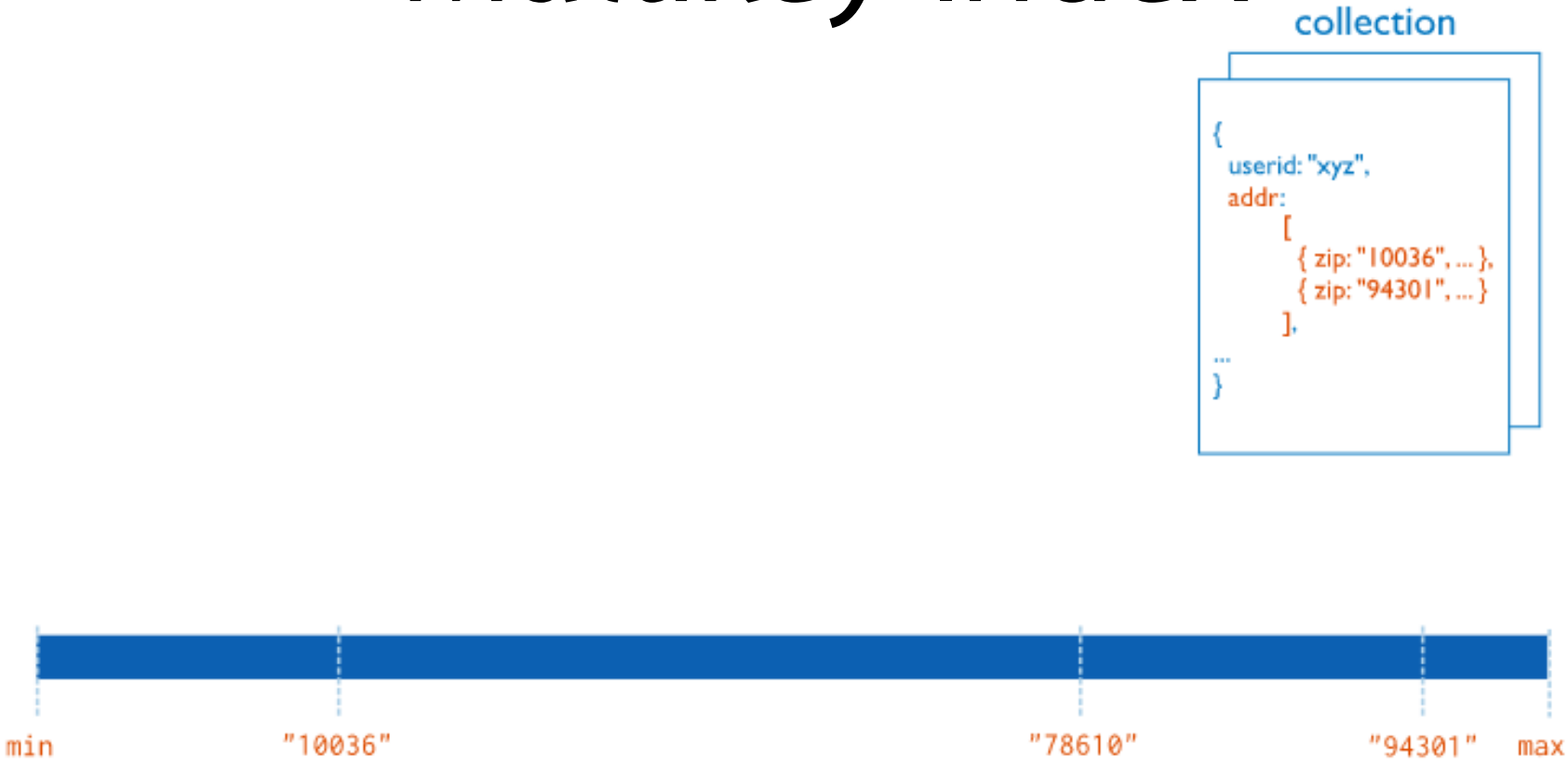
https://docs.mongodb.com/manual/reference/method/db.collection.createIndex/#ensureindex-options

```
db.COLLECTION_NAME.createIndex({KEY1: 1/-1, KEY2: 1/-1, …, KEYN: 1/-1})
```

ตัวอย่างเช่น

```
>>> db.Book.createIndex({category: 1, name: 1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
>>> db.Book.getIndexes()
[{
        "v" : 2,
        "key" : {"_id" : 1},
        "name" : "_id_"
},{
        "v" : 2,
        "key" : {"category" : 1, "name" : 1},
        "name" : "category_1_name_1"
}]
```

Multikey Index



การใช้ Multikey Index ก็เพื่อระบุ Document ที่ต้องการจากคีย์ที่เป็นรูปแบบ arrays

การสร้าง Multikey Index

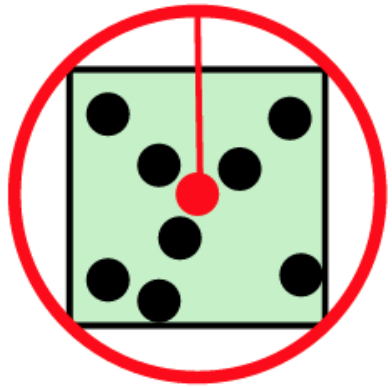
```
db.COLLECTION_NAME.createIndex({KEY1: 1 หรือ -1})
```

ตัวอย่างเช่น

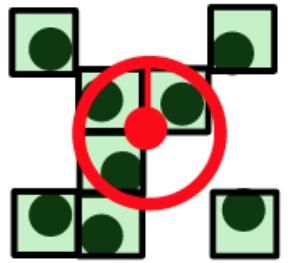
```
{"name": "Albus Dumbledore", "school": "Hogwarts","house": "Gryffindor","pets": ["Phoenix"], "position": "teacher"}
{"name": "Remus Lupin", "school": "Hogwarts","house": "Gryffindor","pets": [], "position": "teacher"}
{"name": "Igor Karkaroff", "school": "Durmstrang","house": "","pets": [], "position": "teacher"}
{"name": "Pomona Sprout", "school": "Hogwarts","house": "Hufflepuff","pets": [], "position": "teacher"}
{"name": "Madame Maxime", "school": "Beauxbatons","house": "","pets": [], "position": "teacher"}
{"name": "Harry Potter", "school": "Hogwarts", "house": "Gryffindor", "pets": ["cat", "bird"], "position": "student"}
{"name": "Cho Chang", "school": "Hogwarts", "house": "Ravenclaw", "pets": ["cat","bird","toad"], "position": "student"}

>>> db.Wizard.createIndex({pets: 1})
```

Geospatial Index



Indexed at 500x500m



Indexed at finest

MongoDB รองรับการ query ในรูปแบบเชิงพิกัด (queries of geospatial coordinate data) เช่น คีย์ที่สร้างจากตำแหน่ง latitude และ longitude เป็นต้น ซึ่งใน MongoDB มี Geospatial Index อยู่ 2 ชนิด ได้แก่

- **2d Index** จะแสดงผลลัพธ์ในรูปแบบเรขาคณิต**เชิงระนาบ** อาทิเช่น คู่ลำดับ (x, y) เป็นต้น
- **2dsphere Index** จะแสดงผลลัพธ์ในรูปแบบเรขาคณิต**ทรงกลม** ซึ่งสอดคล้องกับ**ตำแหน่งที่อยู่บนพื้นโลก**

การสร้าง Geospatial Index

การสร้าง Index แบบระนาบ 2 มิติ โดยต้องกำหนดค่าเป็น “2d”

```
db.COLLECTION_NAME.createIndex({KEY: “2d”})
```

การสร้าง Index แบบเรขาคณิตทรงกลม โดยต้องกำหนดค่าเป็น “2dsphere”

```
db.COLLECTION_NAME.createIndex({KEY: “2dsphere”})
```

ข้อมูลเชิงเรขาคณิตทางด้านภูมิศาสตร์

ข้อมูลเชิงเรขาคณิตทางด้านภูมิศาสตร์ หรือเรียกว่า GeoJSON คือ รูปแบบข้อมูลเชิงเรขาคณิตในระบบพิกัดแบบ (latitude , longitude) โดยที่ latitude จะมีค่าอยู่ในช่วง -90 ถึง 90 องศา ขณะที่ longitude จะอยู่ในช่วง -180 ถึง 180 องศา ซึ่งใน MongoDB มีการจัดการข้อมูลเชิงพิกัด 3 รูปแบบ

- Point คือ คู่ลำดับที่เก็บข้อมูลพิกัดตำแหน่งเดียว หรือ จุดเดียว ตัวอย่างเช่น

```
{location: {type: "Point" , coordinates:[39, 15] }}
```

- LineString คือ คู่ลำดับที่เก็บข้อมูลพิกัดมากกว่า 1 ตำแหน่งในรูปแบบอาร์เรย์

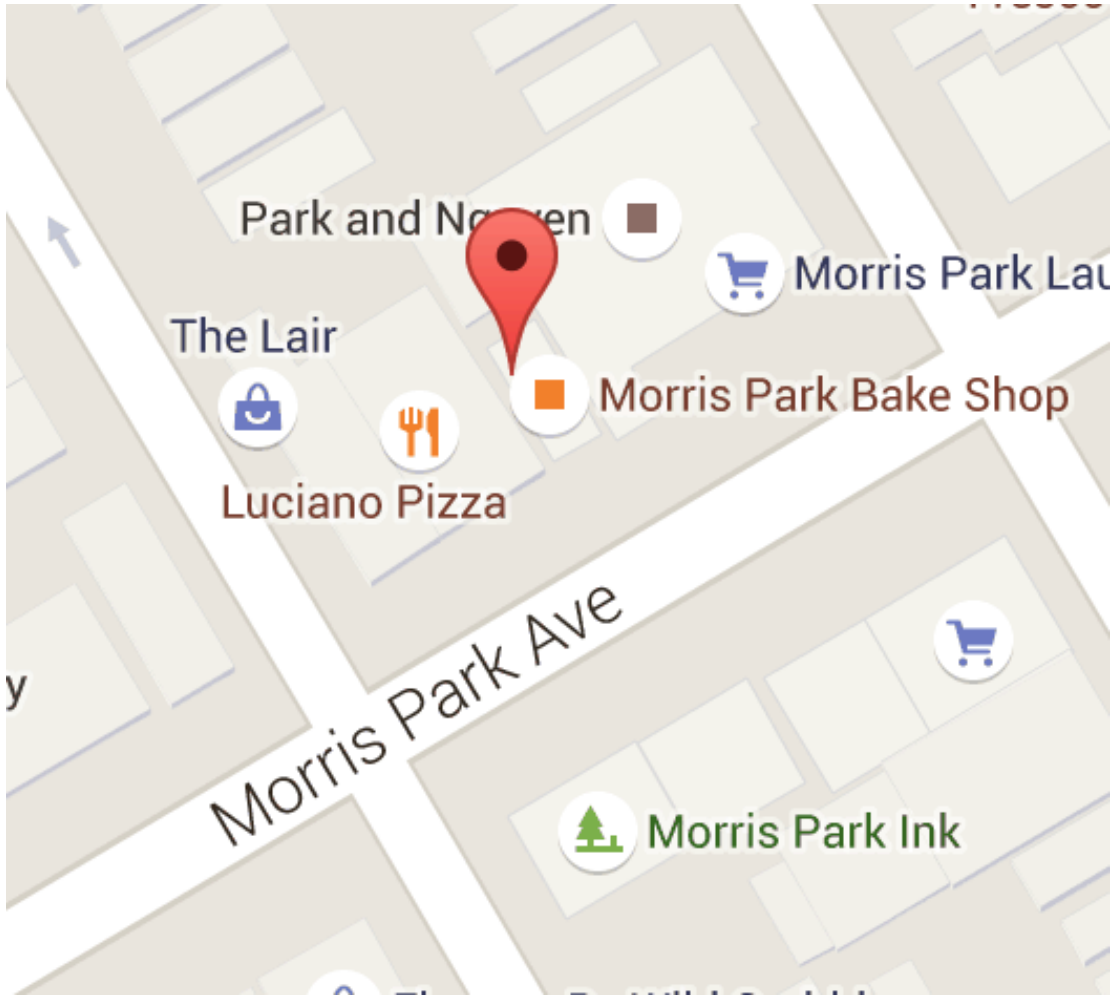
```
{location: {type: "LineString", coordinates:[[38, 4], [40, 6]] }}
```

- Polygon คือ คู่ลำดับที่เก็บข้อมูลพิกัดมากกว่า 1 ตำแหน่งในรูปแบบอาร์เรย์ โดยอย่างน้อยต้องมี 4 ตำแหน่ง โดยตำแหน่งแรกและสุดท้ายต้องเป็นตำแหน่งเดียวกัน

```
{location: { type: "Polygon", coordinates:[[[121.01, 14.51],
[121.00, 14.51], [121.00, 14.52],[121.01,14.51]]]}}
```

ข้อมูลเชิงเรขาคณิตทางด้านภูมิศาสตร์

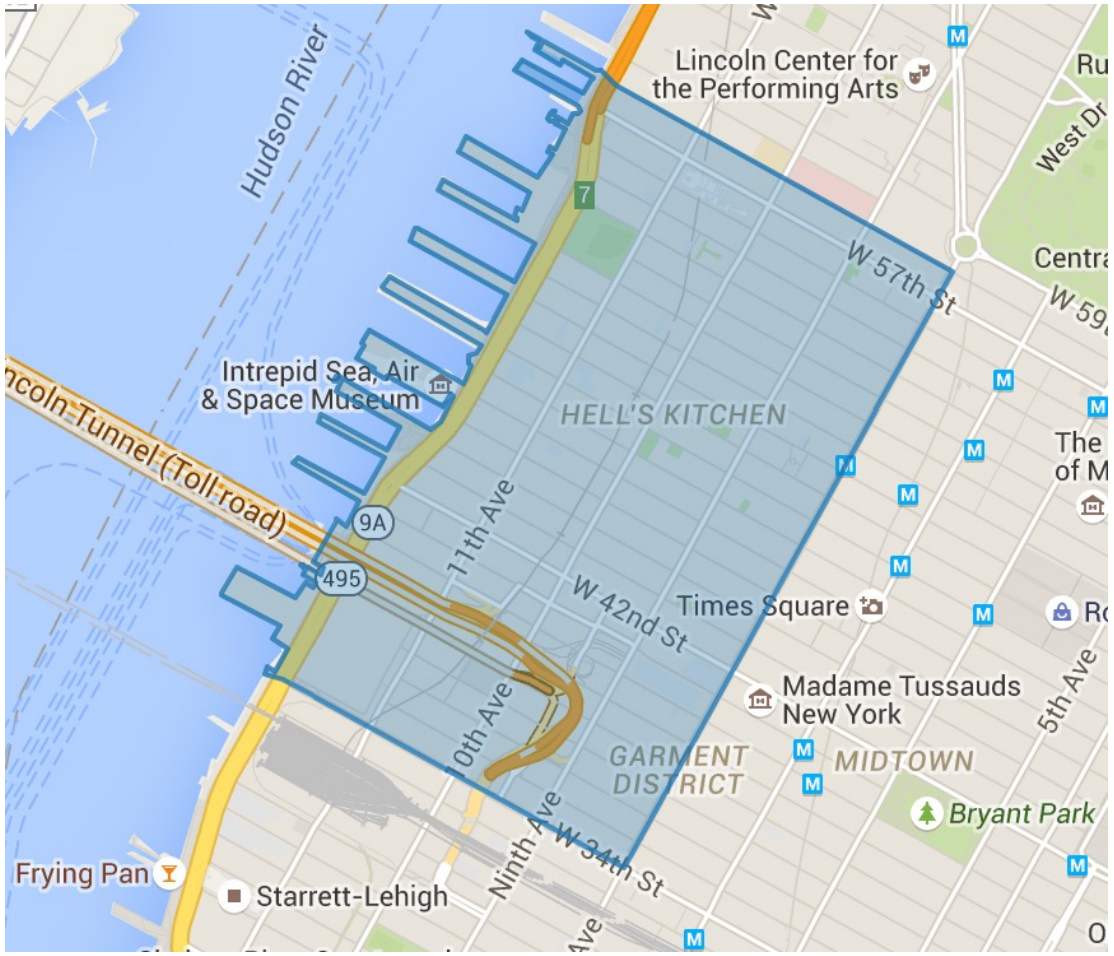
ตัวอย่างชนิดข้อมูล GeoJSON แบบ Point




```
{
    location: {
        type: "Point",
        coordinates: [-73.856077, 40.848447]
    },
    name: "Morris Park Bake Shop"
}
```

ข้อมูลเชิงเรขาคณิตทางด้านภูมิศาสตร์

ตัวอย่างชนิดข้อมูล GeoJSON แบบ Polygon




```
{
    geometry: {
        type: "Polygon",
        coordinates: [[
            [ -73.99, 40.75 ],
            ...
            [ -73.98, 40.76 ],
            [ -73.99, 40.75 ]
        ]]
     },
     name: "Hell's Kitchen"
}
```

ตัวดำเนินการทาง Geospatial Query

ตัวดำเนินการทาง Geospatial Query สำหรับใช้จัดการข้อมูลเชิงเรขาคณิตทางด้านภูมิศาสตร์ (GeoJSON)

ตัวดำเนินการ	ความหมาย
$geometry	ใช้กำหนดตำแหน่งทางด้านภูมิศาสตร์
$maxDistance	ใช้เพื่อระบุระยะทางที่มากสุดจากจุดที่กำหนดถึงจุดศูนย์กลาง (เมตร)
$minDistance	ใช้เพื่อระบุระยะทางที่น้อยสุดจากจุดที่กำหนดถึงจุดศูนย์กลาง (เมตร)
$near	ใช้ค้นหา Document ที่มีค่าพิกัดใกล้เคียงกับค่าที่กำหนด

โดยปกติแล้ว ตัวดำเนินการข้างต้นจะใช้งานร่วมกัน ดังตัวอย่างในหน้าถัดไป

ตัวอย่างการค้นหาข้อมูลพิกัดโดยอาศัย Geospatial Index ร่วมกับตัวดำเนินการทาง Geospatial Query

```
db.places.find(
    {
      location:
        { $near :
            {
              $geometry: { type: "Point",  coordinates: [ -73.9667, 40.78 ] },
              $minDistance: 1000,
              $maxDistance: 5000
            }
        }
    }
)
```

ตัวอย่างข้อมูลภูมิศาสตร์บางส่วนของ Food Collection


```
{"_id":{"$oid":"55cba2476c522cafdb053add"},"location":{"coordinates":[-73.856077,40.848447],"type":"Point"},"name":"Morris Park Bake Shop"}
{"_id":{"$oid":"55cba2476c522cafdb053ade"},"location":{"coordinates":[-73.961704,40.662942],"type":"Point"},"name":"Wendy'S"}
{"_id":{"$oid":"55cba2476c522cafdb053adf"},"location":{"coordinates":[-73.98241999999999,40.579505],"type":"Point"},"name":"Riviera Caterer"}
{"_id":{"$oid":"55cba2476c522cafdb053ae0"},"location":{"coordinates":[-73.8601152,40.7311739],"type":"Point"},"name":"Tov Kosher Kitchen"}
{"_id":{"$oid":"55cba2476c522cafdb053ae1"},"location":{"coordinates":[-73.8803827,40.7643124],"type":"Point"},"name":"Brunos On The Boulevard"}
{"_id":{"$oid":"55cba2476c522cafdb053ae2"},"location":{"coordinates":[-73.98513559999999,40.7676919],"type":"Point"},"name":"Dj Reynolds Pub And Restaurant"}
{"_id":{"$oid":"55cba2476c522cafdb053ae3"},"location":{"coordinates":[-73.9068506,40.6199034],"type":"Point"},"name":"Wilken'S Fine Food"}
{"_id":{"$oid":"55cba2476c522cafdb053ae4"},"location":{"coordinates":[-74.00528899999999,40.628886],"type":"Point"},"name":"Regina Caterers"}
{"_id":{"$oid":"55cba2476c522cafdb053ae5"},"location":{"coordinates":[-73.9482609,40.6408271],"type":"Point"},"name":"Taste The Tropics Ice Cream"}
{"_id":{"$oid":"55cba2476c522cafdb053ae6"},"location":{"coordinates":[-74.1377286,40.6119572],"type":"Point"},"name":"Kosher Island"}
{"_id":{"$oid":"55cba2476c522cafdb053ae7"},"location":{"coordinates":[-73.8786113,40.8502883],"type":"Point"},"name":"Wild Asia"}
{"_id":{"$oid":"55cba2476c522cafdb053ae8"},"location":{"coordinates":[-73.9973325,40.61174889999999],"type":"Point"},"name":"C \u0026 C Catering Service"}
{"_id":{"$oid":"55cba2476c522cafdb053ae9"},"location":{"coordinates":[-73.96926909999999,40.7685235],"type":"Point"},"name":"1 East 66Th Street Kitchen"}
{"_id":{"$oid":"55cba2476c522cafdb053aea"},"location":{"coordinates":[-73.871194,40.6730975],"type":"Point"},"name":"May May Kitchen"}
{"_id":{"$oid":"55cba2476c522cafdb053aeb"},"location":{"coordinates":[-73.9653967,40.6064339],"type":"Point"},"name":"Seuda Foods"}
{"_id":{"$oid":"55cba2476c522cafdb053aec"},"location":{"coordinates":[-73.97822040000001,40.6435254],"type":"Point"},"name":"Carvel Ice Cream"}
{"_id":{"$oid":"55cba2476c522cafdb053aed"},"location":{"coordinates":[-73.7032601,40.7386417],"type":"Point"},"name":"Carvel Ice Cream"}
{"_id":{"$oid":"55cba2476c522cafdb053aee"},"location":{"coordinates":[-74.0259567,40.6353674],"type":"Point"},"name":"Nordic Delicacies"}
{"_id":{"$oid":"55cba2476c522cafdb053aef"},"location":{"coordinates":[-73.9829239,40.6580753],"type":"Point"},"name":"The Movable Feast"}
{"_id":{"$oid":"55cba2476c522cafdb053af0"},"location":{"coordinates":[-73.839297,40.78147],"type":"Point"},"name":"Sal'S Deli"}
{"_id":{"$oid":"55cba2476c522cafdb053af1"},"location":{"coordinates":[-73.95171,40.767461],"type":"Point"},"name":"Glorious Food"}
{"_id":{"$oid":"55cba2476c522cafdb053af2"},"location":{"coordinates":[-73.9925306,40.7309346],"type":"Point"},"name":"Bully'S Deli"}
{"_id":{"$oid":"55cba2476c522cafdb053af3"},"location":{"coordinates":[-73.976112,40.786714],"type":"Point"},"name":"Harriet'S Kitchen"}
{"_id":{"$oid":"55cba2476c522cafdb053af4"},"location":{"coordinates":[-73.94024739999999,40.7623288],"type":"Point"},"name":"Steve Chu'S Deli \u0026 Grocery"}
{"_id":{"$oid":"55cba2476c522cafdb053af5"},"location":{"coordinates":[-73.96805719999999,40.7925587],"type":"Point"},"name":"P \u0026 S Deli Grocery"}
{"_id":{"$oid":"55cba2476c522cafdb053af6"},"location":{"coordinates":[-73.996984,40.72589],"type":"Point"},"name":"Angelika Film Center"}
{"_id":{"$oid":"55cba2476c522cafdb053af7"},"location":{"coordinates":[-73.9634876,40.6940001],"type":"Point"},"name":"White Castle"}
{"_id":{"$oid":"55cba2476c522cafdb053af8"},"location":{"coordinates":[-73.8642349,40.75356],"type":"Point"},"name":"Ho Mei Restaurant"}
{"_id":{"$oid":"55cba2476c522cafdb053af9"},"location":{"coordinates":[-74.0085357,40.70620539999999],"type":"Point"},"name":"The Country Cafe"}
{"_id":{"$oid":"55cba2476c522cafdb053afa"},"location":{"coordinates":[-73.9246028,40.6522396],"type":"Point"},"name":"Shashemene Int'L Restaura"}
{"_id":{"$oid":"55cba2476c522cafdb053afb"},"location":{"coordinates":[-74.00920839999999,40.7132925],"type":"Point"},"name":"Downtown Deli"}
{"_id":{"$oid":"55cba2476c522cafdb053afc"},"location":{"coordinates":[-73.84856870000002,40.8903781],"type":"Point"},"name":"Carvel Ice Cream"}
{"_id":{"$oid":"55cba2476c522cafdb053afd"},"location":{"coordinates":[-73.991495,40.692273],"type":"Point"},"name":"Dunkin' Donuts"}
{"_id":{"$oid":"55cba2476c522cafdb053afe"},"location":{"coordinates":[-73.9998042,40.7251256],"type":"Point"},"name":"Olive'S"}
```

ตัวอย่างการค้นหาข้อมูลพิกัดโดยอาศัย Geospatial Index ร่วมกับตัวดำเนินการทาง Geospatial Query

```
>>> db.Food.find({location:{$near :{$geometry: { type: "Point",
coordinates: [ -73.856077,40.848447 ] }, $minDistance: 100, $maxDistance:
300}}},{"_id":0,"location":0})

//ผลลัพธ์
{ "name" : "New Fresco Toetillas Tommy'S Kitchen Inc" }
{ "name" : "Nana'S Kitchen" }
{ "name" : "15 Flavors " }
{ "name" : "Antivari Pizza" }
{ "name" : "Luke'S Lounge" }
{ "name" : "Subway" }
{ "name" : "Emilio'S Pizza & Pasta" }
{ "name" : "Enrico'S Pastry Shop & Caffe" }
{ "name" : "Captain'S Pizzeria And Restaurant" }
{ "name" : "Soups,Salads & Beyond" }
{ "name" : "Scaglione Brothers Bakery" }
{ "name" : "Burger Time" }
{ "name" : "Burger Time" }
{ "name" : "Patricia'S" }
{ "name" : "Dunkin Donuts" }
```

Text Index

MongoDB สามารถกำหนดคีย์เป็นชนิดข้อมูลแบบข้อความได้ เพื่อให้ง่ายต่อการค้นหาในรูปแบบตัวอักษร ซึ่งถือว่าเป็นชนิดคีย์ที่มีประโยชน์มากและมีประสิทธิภาพคล้ายกับเวลาเราค้นหาข้อมูลด้วย Google search **ข้อควรระวัง Text Indexes สามารถมีได้เพียงอันเดียวเท่านั้น** ซึ่งสามารถสร้างคีย์ข้อความได้จากคำสั่งต่อไปนี้

```
db.COLLECTION_NAME.createIndex({KEY: "text"})
```

ตัวอย่าง

```
>>> db.Food.createIndex({name: "text"})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 2,
        "numIndexesAfter" : 3,
        "ok" : 1
}
```

Text Index

ประโยชน์ของการกำหนดคีย์ประเภท Text ได้แก่ นักศึกษาสามารถค้นหาข้อความที่ต้องการผ่านคีย์ประเภท Text ได้ โดยอาศัยตัวดำเนินการข้อความ คือ $text และ $search ร่วมกัน อาทิเช่น

```
db.COLLECTION_NAME.find({$text:{$search:"text value"}})
```

ตัวอย่าง

```
db.Food.find({$text:{$search:"Pizza"}},{"_id":0,"location":0})
{ "name" : "Mangias Pizza And Pizza" }
{ "name" : "Pizza" }
{ "name" : "My Pizza" }
{ "name" : "My Pizza" }
{ "name" : "Neighborhood Pizza" }
{ "name" : "Best Pizza" }
{ "name" : "Spicy Pizza" }
Type "it" for more
```

ตัวดำเนินการ $text และ $search

$text และ $search เป็น query operator ที่ใช้สำหรับ Full-Text Search (การค้นหาข้อความเต็มรูปแบบ) ใน MongoDB ซึ่งช่วยให้เราสามารถค้นหาข้อความภายใน string fields ได้อย่างรวดเร็วและมีประสิทธิภาพ โดยพื้นฐานของ $text และ $search

Operator	รายละเอียด
$text	• ใช้เพื่อค้นหา document ที่ตรงกับ index ข้อความ (text index) • ต้องสร้าง text index ก่อนถึงจะใช้ได้ • ใช้ร่วมกับ $search เพื่อค้นหาคำที่ตรงกับ index
$search	• ใช้ค้นหาข้อความภายใน fields ที่ถูก index เป็น text index • สามารถค้นหาคำที่อยู่ใน document ได้โดยไม่ต้องรู้โครงสร้างของข้อความทั้งหมด • รองรับ multiple words, phrase search และ stop words (คำที่ถูกละเว้น เช่น "the", "is") • ไม่สนใจตัวพิมพ์ใหญ่-พิมพ์เล็ก (case-insensitive) • รองรับการค้นหา บางส่วนของคำ ได้โดยใช้ "\" (double quotes) หรือ "-" (negative search)

กรณีที่ 1 การค้นหาข้อความง่าย ๆ

ข้อมูล

```
{ "_id": 1, "title": "MongoDB Full-Text Search", "content": "MongoDB supports
text search using the $text operator." }
{ "_id": 2, "title": "Introduction to Databases", "content": "Databases store
structured information efficiently." }
{ "_id": 3, "title": "Using MongoDB for Big Data", "content": "Big data
solutions often require NoSQL databases like MongoDB." }
```

คำสั่ง

```
db.articles.find({ $text: { $search: "MongoDB" } })
```

ผลลัพธ์ -> Document _id: 1 และ _id: 3 ถูกเลือกเพราะมี "MongoDB"

```
{ "_id": 1, "title": "MongoDB Full-Text Search", "content": "MongoDB supports
text search using the $text operator." }
{ "_id": 3, "title": "Using MongoDB for Big Data", "content": "Big data
solutions often require NoSQL databases like MongoDB." }
```

กรณีที่ 2 ค้นหาด้วยวลี (Phrase Search)

ข้อมูล

```
{ "_id": 1, "title": "MongoDB Full-Text Search", "content": "MongoDB supports
text search using the $text operator." }
{ "_id": 2, "title": "Introduction to Databases", "content": "Databases store
structured information efficiently." }
{ "_id": 3, "title": "Using MongoDB for Big Data", "content": "Big data
solutions often require NoSQL databases like MongoDB." }
```

คำสั่ง

จะหาคำว่า Full-Text Search เลยต้องมีแท็กเปิดเป็นๆ \" แท็กปิดเป็น \"

```
db.articles.find({ $text: { $search: "\"Full-Text Search\"" } })
```

ผลลัพธ์

```
{ "_id": 1, "title": "MongoDB Full-Text Search", "content": "MongoDB supports
text search using the $text operator." }
```

- MongoDB จะค้นหา "Full-Text Search" เป็นวลีที่ต้องอยู่ติดกัน

กรณีที่ 3 ค้นหาคำที่ต้องมีและคำที่ต้องไม่มี

ข้อมูล

```
{ "_id": 1, "title": "MongoDB Full-Text Search", "content": "MongoDB supports
text search using the $text operator." }
{ "_id": 2, "title": "Introduction to Databases", "content": "Databases store
structured information efficiently." }
{ "_id": 3, "title": "Using MongoDB for Big Data", "content": "Big data
solutions often require NoSQL databases like MongoDB." }
```

คำสั่ง

```
db.articles.find({ $text: { $search: "MongoDB -\"Big Data\"" } })
```

ผลลัพธ์

```
{ "_id": 1, "title": "MongoDB Full-Text Search", "content": "MongoDB supports
text search using the $text operator." }
```

- MongoDB จะตัด _id: 3 ออกไปเพราะมี "Big Data"

กรณีที่ 4 ค้นหาคำที่เป็นตัวเลือก (OR)

ข้อมูล

```
{ "_id": 1, "title": "MongoDB Full-Text Search", "content": "MongoDB supports
text search using the $text operator." }
{ "_id": 2, "title": "Introduction to Databases", "content": "Databases store
structured information efficiently." }
{ "_id": 3, "title": "Using MongoDB for Big Data", "content": "Big data
solutions often require NoSQL databases like MongoDB." }
```

คำสั่ง

มีคือการ search ว่า MongoDB หรือ Databases

```
db.articles.find({ $text: { $search: "MongoDB Databases" } })
```

ผลลัพธ์

```
{ "_id": 1, "title": "MongoDB Full-Text Search", "content": "MongoDB supports
text search using the $text operator." }
{ "_id": 2, "title": "Introduction to Databases", "content": "Databases store
structured information efficiently." }
{ "_id": 3, "title": "Using MongoDB for Big Data", "content": "Big data
solutions often require NoSQL databases like MongoDB." }
```


- MongoDB ค้นหา "MongoDB" หรือ "Databases" และคืนค่า document ที่มีคำใดคำหนึ่ง

ข้อควรระวังของการค้นหาจาก Text

จากตัวอย่างต่อไปนี้ เป็นการค้นหาทุก document ที่มีคำว่า coffee หรือ shop

```
db.stores.find( { $text: { $search: "coffee shop" } } )
```

จากตัวอย่างต่อไปนี้ เป็นการค้นหาทุก document ที่มีคำว่า coffee และ shop

```
db.stores.find( { $text: { $search: "\"coffee shop\"" } } )
```

ข้อจำกัดของ $text และ $search

❌ ต้องสร้าง text index ก่อนใช้

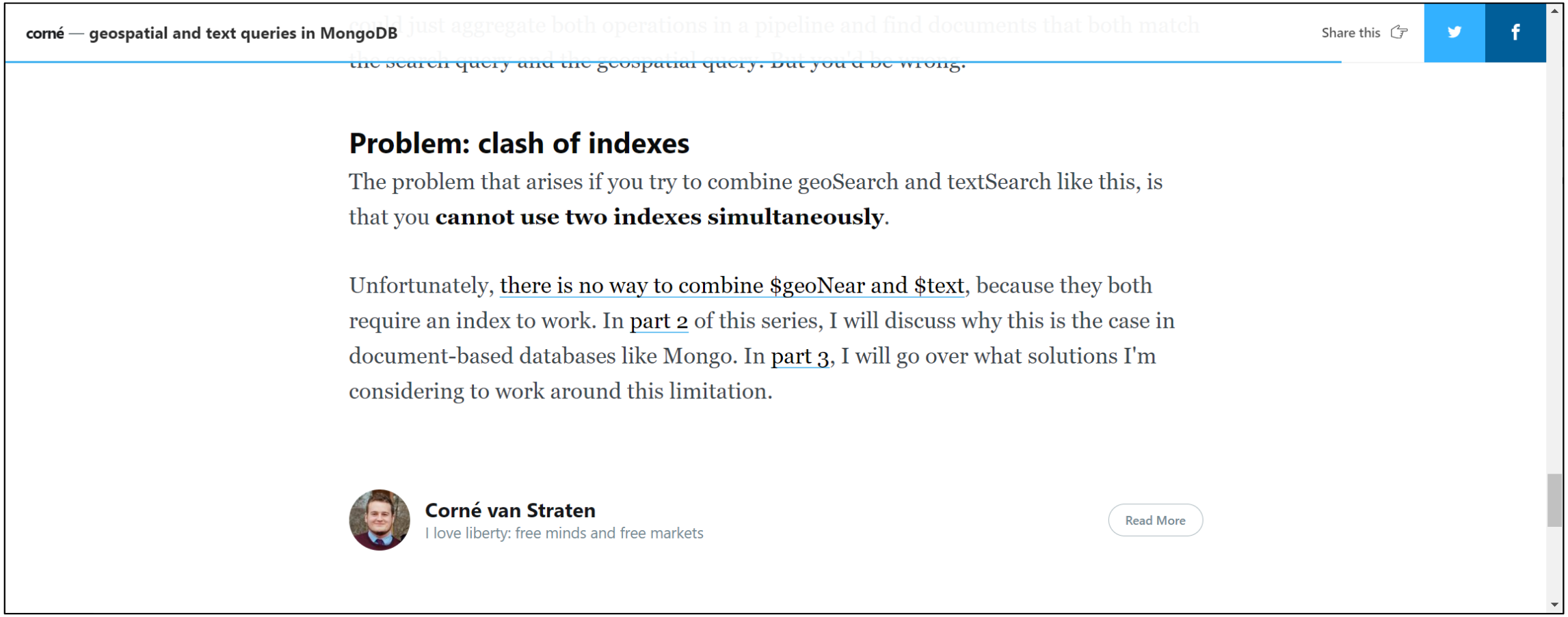
❌ รองรับแค่ 1 text index ต่อ collection

❌ ค้นหาไม่รองรับ wildcard (*) หรือ regex

❌ ไม่สามารถใช้ $text กับ field ที่เป็น array ของ embedded documents

✅ ทำงานได้เร็วขึ้นมากหากใช้ text index อย่างเหมาะสม

ข้อควรระวังของการค้นหา Text และ Geo



การใช้งาน $geoNear และ $text ร่วมกัน

$geoNear และ $text เป็น aggregation stages ที่สามารถใช้ร่วมกันใน aggregation pipeline เพื่อค้นหา document โดยพิจารณาทั้งตำแหน่งทางภูมิศาสตร์ (GeoSpatial Search) และ ข้อความ (Full-Text Search) พร้อมกัน

```
db.places.aggregate([
    {
        $geoNear: {
            near: { type: "Point", coordinates: [100.52, 13.74] },
            distanceField: "distance",
            maxDistance: 5000, // ระยะทางสูงสุด 5 กม.
            spherical: true
        }
    },{
        $match: { $text: { $search: "laptop" } }
    },{
        $sort: { score: { $meta: "textScore" }, distance: 1 }
    }
])
```

การใช้งาน $geoNear และ $text ร่วมกัน

ข้อมูล

```
{ "_id": 1, "name": "Tech Shop", "description": "We sell computers and accessories",
"location": { "type": "Point", "coordinates": [100.523186, 13.736717] } }
{ "_id": 2, "name": "Coffee House", "description": "Best coffee in Bangkok", "location":
{ "type": "Point", "coordinates": [100.500000, 13.750000] } }
{ "_id": 3, "name": "Bookstore", "description": "A place to find books and gifts",
"location": { "type": "Point", "coordinates": [100.530000, 13.740000] } }
{ "_id": 4, "name": "Laptop World", "description": "Laptops and accessories available
here", "location": { "type": "Point", "coordinates": [100.510000, 13.730000] } }
```

สร้าง index ที่จำเป็น

```
// สร้าง 2dsphere index สำหรับการค้นหาตำแหน่ง
db.places.createIndex({ location: "2dsphere" });

// สร้าง text index สำหรับการค้นหาข้อความ
db.places.createIndex({ name: "text", description: "text" });
```

การใช้งาน $geoNear และ $text ร่วมกัน

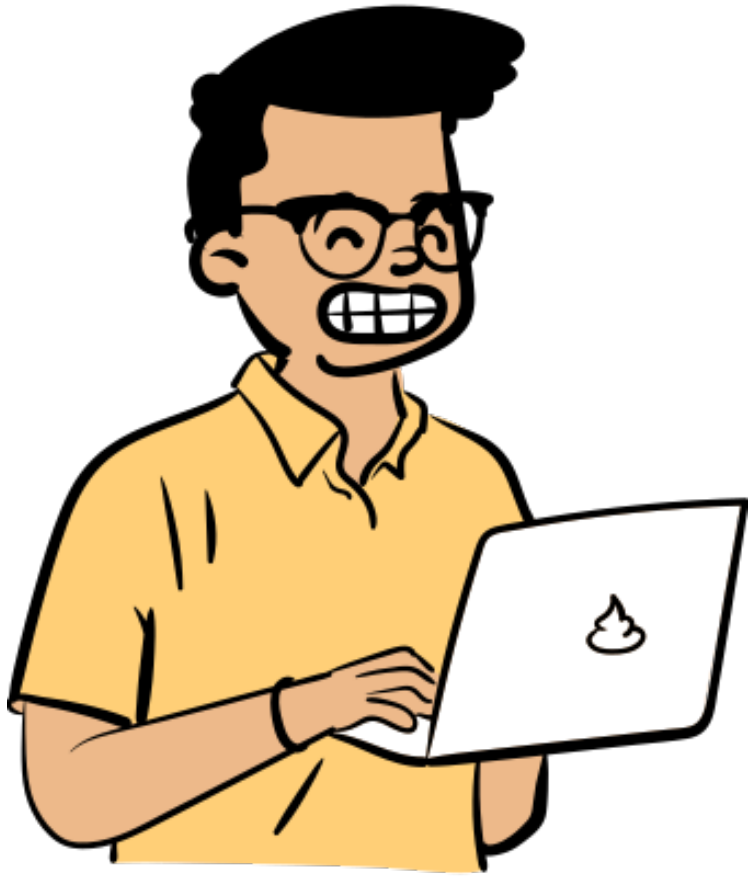
ข้อมูล

```
{ "_id": 1, "name": "Tech Shop", "description": "We sell computers and accessories",
"location": { "type": "Point", "coordinates": [100.523186, 13.736717] } }
{ "_id": 2, "name": "Coffee House", "description": "Best coffee in Bangkok", "location":
{ "type": "Point", "coordinates": [100.500000, 13.750000] } }
{ "_id": 3, "name": "Bookstore", "description": "A place to find books and gifts",
"location": { "type": "Point", "coordinates": [100.530000, 13.740000] } }
{ "_id": 4, "name": "Laptop World", "description": "Laptops and accessories available
here", "location": { "type": "Point", "coordinates": [100.510000, 13.730000] } }
```

ผลลัพธ์ที่คาดหวัง

```
{ "_id": 4, "name": "Laptop World", "description": "Laptops and accessories available
here", "location": { "type": "Point", "coordinates": [100.510000, 13.730000] },
"distance": 1350 }
```

"Laptop World" ถูกเลือกเพราะอยู่ ใกล้พิกัดที่กำหนด และ มีคำว่า "laptop"



การเรียกดู Document โดยใช้ Aggregation

การใช้ Aggregation

Aggregation เป็นการดำเนินการโดยใช้คำสั่งในการประมวลผล เพื่อ**รวบรวมข้อมูลจาก Document ต่าง ๆ ให้รวมเป็นกลุ่มเดียวกัน** และแสดงออกมาเป็นผลลัพธ์ที่ได้จากการคำนวณ เช่น **sum, avg, group, max, min** เป็นต้น ซึ่งใน MongoDB มี 3 วิธี ให้เลือกใช้งาน ได้แก่

- Aggregation Pipeline
- Map-Reduce Function
- Single Purpose Aggregation Methods

หลักการทำงาน Single Purpose Aggregation

สำหรับใน MongoDB นั้น Single Purpose Aggregation มีหน้าทีรวบรวมข้อมูลจาก Document ต่าง ๆ เพื่อให้ได้ผลลัพธ์ตามจุดประสงค์ของแต่ละคำสั่ง ได้แก่ **count()** และ **distinct()**

```
db.COLLECTION_NAME.count({condition})

db.COLLECTION_NAME.distinct({key}, {condition})
```

ตัวอย่างการใช้งาน **distinct()** เพื่อใช้แสดงค่าที่ไม่ซ้ำกัน อาทิเช่น

```
>>> db.inventory.find()
{ "_id": 1, "dept": "A", "item": { "sku": "111", "color": "red" }, "sizes": [ "S", "M" ] }
{ "_id": 2, "dept": "A", "item": { "sku": "111", "color": "blue" }, "sizes": [ "M", "L" ] }
{ "_id": 3, "dept": "B", "item": { "sku": "222", "color": "blue" }, "sizes": "S" }
{ "_id": 4, "dept": "A", "item": { "sku": "333", "color": "black" }, "sizes": [ "S" ] }

>>> db.inventory.distinct("dept")
[ "A", "B" ]
```

หลักการทำงาน Single Purpose Aggregation

```
>>> db.inventory.find()
{ "_id": 1, "dept": "A", "item": { "sku": "111", "color": "red" }, "sizes": [ "S", "M" ] }
{ "_id": 2, "dept": "A", "item": { "sku": "111", "color": "blue" }, "sizes": [ "M", "L" ] }
{ "_id": 3, "dept": "B", "item": { "sku": "222", "color": "blue" }, "sizes": "S" }
{ "_id": 4, "dept": "A", "item": { "sku": "333", "color": "black" }, "sizes": [ "S" ] }


>>> db.inventory.distinct("dept")
[ "A", "B" ]

>>> db.inventory.distinct( "item.sku" )
[ "111", "222", "333" ]

>>> db.inventory.distinct( "item.sku", { dept: "A" } )
[ "111", "333" ]
```

หลักการทำงาน Single Purpose Aggregation

ขณะที่ การใช้งาน **count()** เพื่อนับจำนวน **Document** เช่น

```
>>> db.Item.count()
6

>>> db.Item.count({amount:{$gt:20}})
2
```

อย่างไรก็ตาม การใช้คำสั่ง **count()** แบบ Aggregation จะให้ผลลัพธ์คล้ายกับคำสั่งต่อไปนี้

```
>>> db.Item.find().count()
6

>>> db.Item.find({amount:{$gt:20}}).count()
2
```