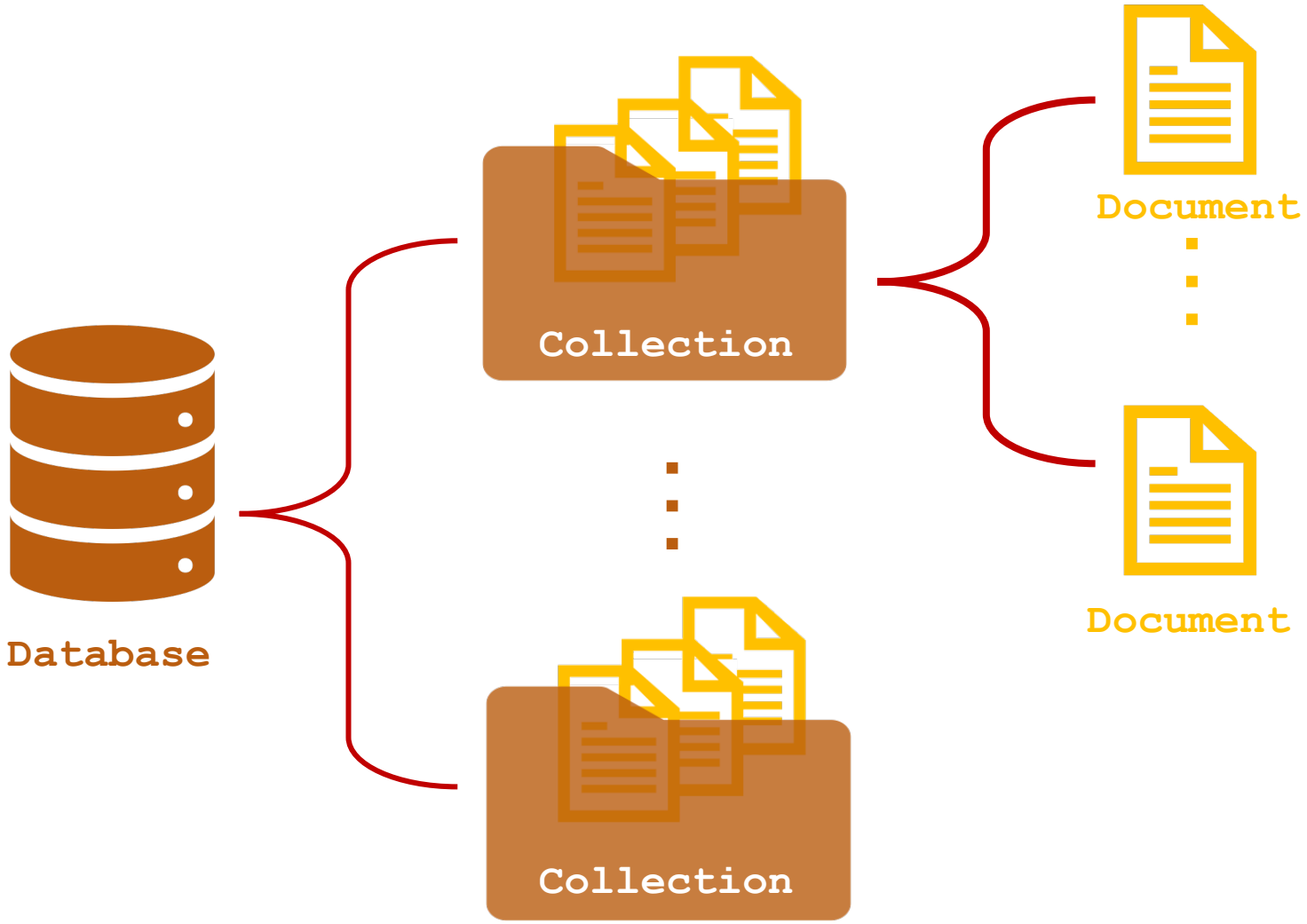
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang

Chapter 13: MongoDB Implementation

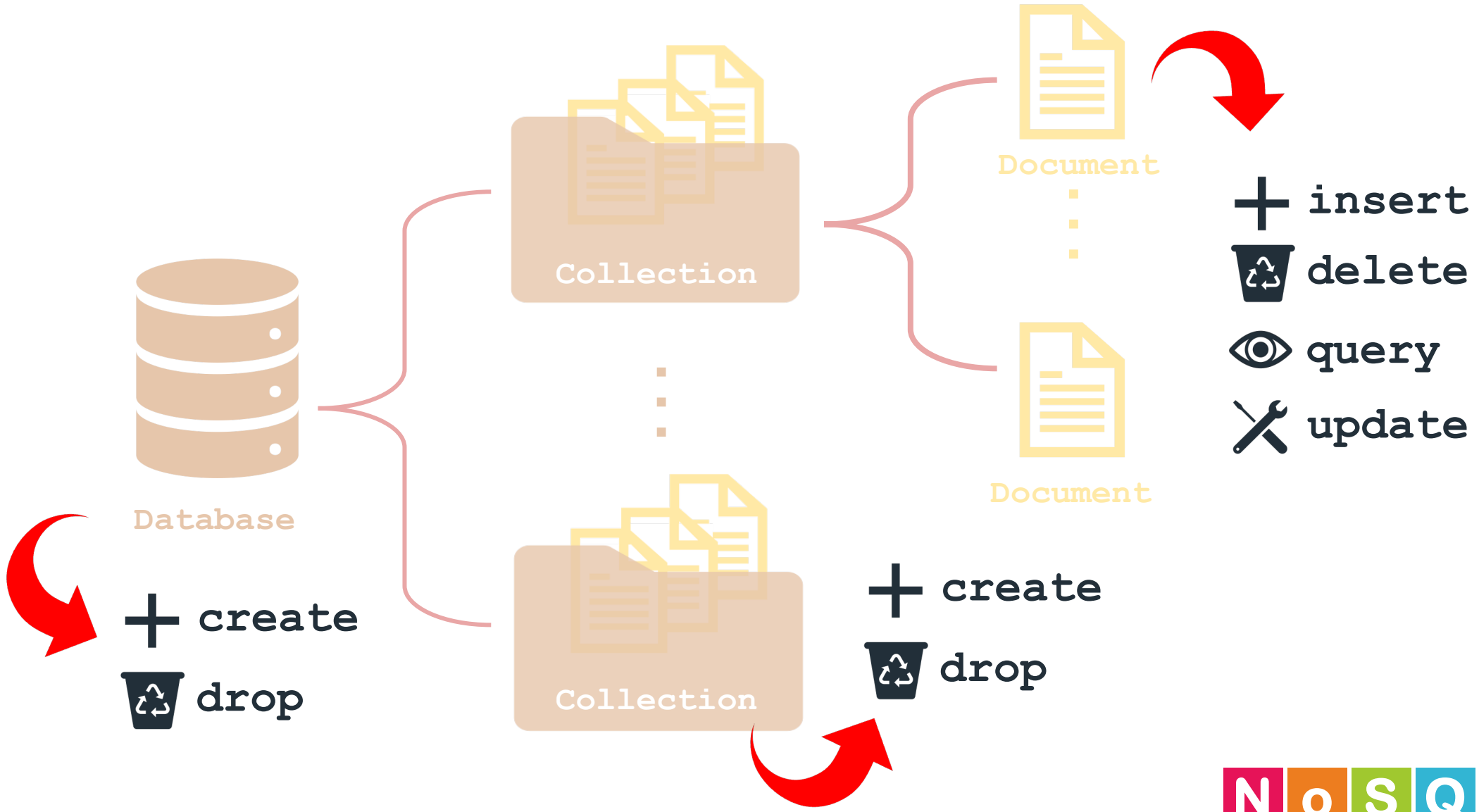
By Asst.Prof.Dr. Taravichet Titijaroonroj

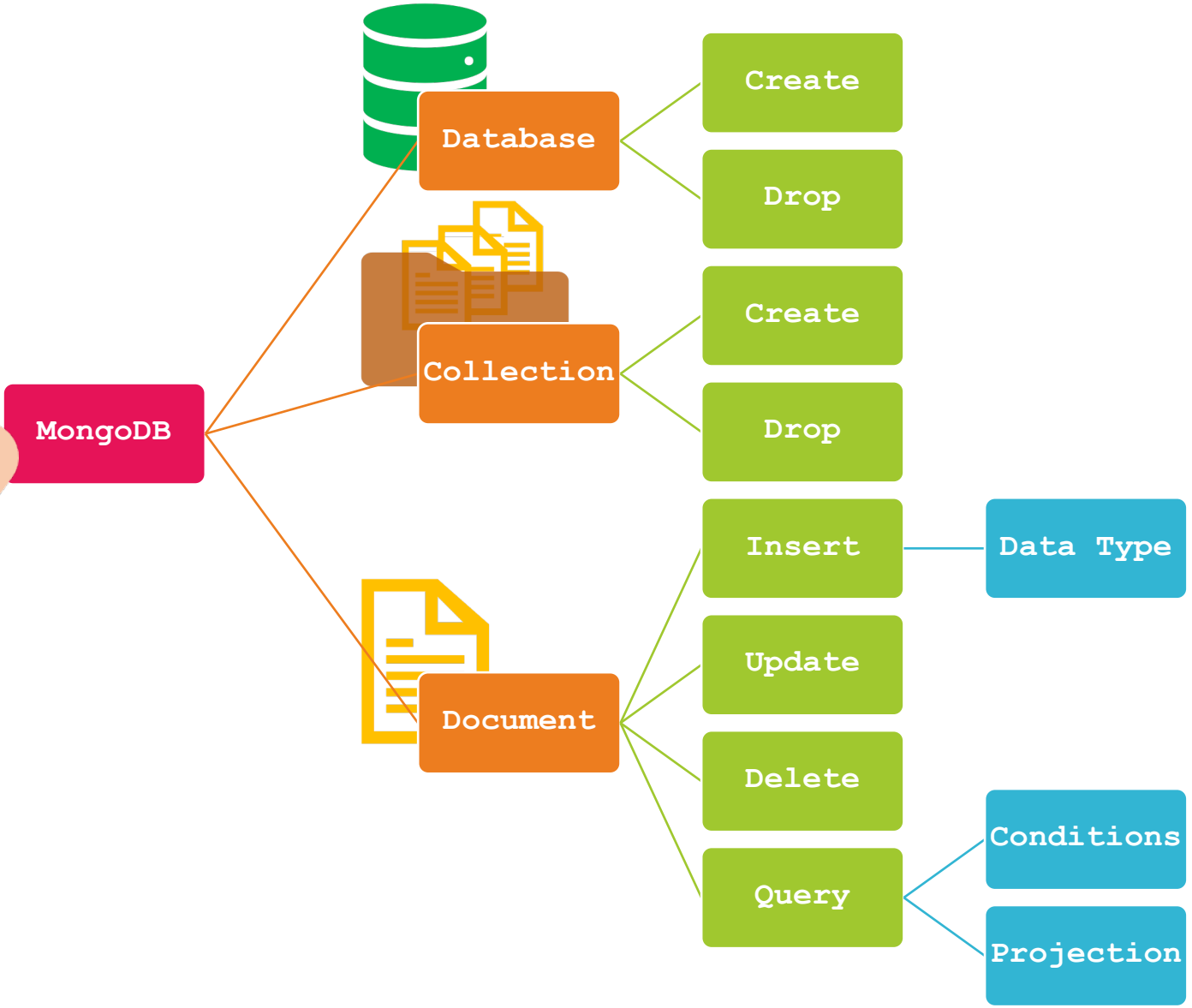
Practical NoSQL Database

โครงสร้าง MongoDB

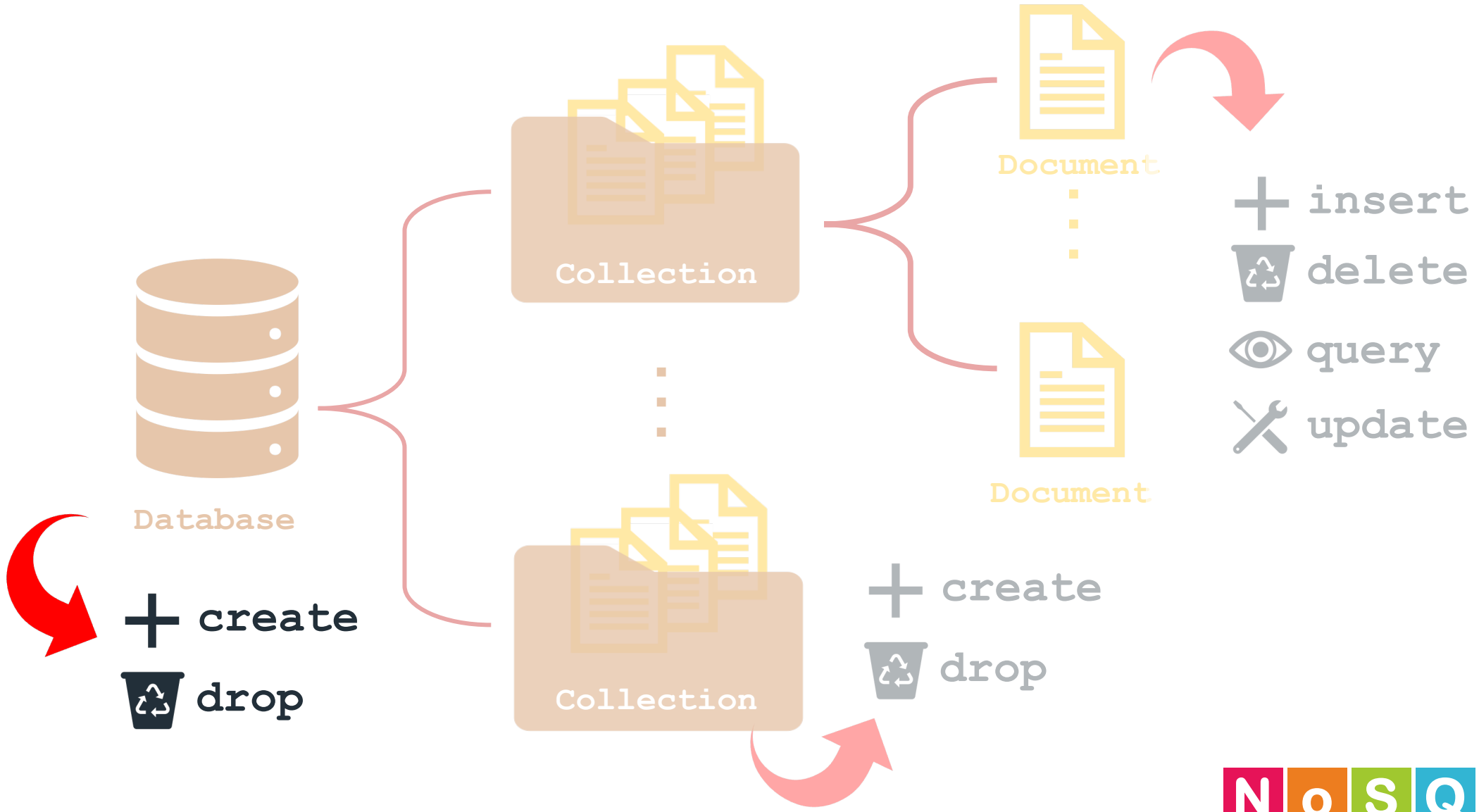


คำสั่งและตัวดำเนินการของ MongoDB




MongoDB
Database
Create
Drop
Collection
Create
Drop
Document
Insert
Data Type
Update
Delete
Query
Conditions
Projection

คำสั่งและตัวดำเนินการของ MongoDB



การสร้างฐานข้อมูล

การสร้างฐานข้อมูลใน MongoDB จะอาศัยคำสั่ง “**use**” สำหรับสร้างฐานข้อมูลใหม่ตามชื่อที่กำหนด (**DATABASE_NAME**) สำหรับกรณีที่ไม่พบฐานข้อมูลดังกล่าวใน MongoDB แต่ถ้าพบฐานข้อมูลดังกล่าวอยู่ก่อนแล้ว MongoDB จะไม่สร้างฐานข้อมูลให้แต่จะย้ายไปที่ฐานข้อมูลดังกล่าวให้

```
use DATABASE_NAME
```

ตัวอย่างเช่น

```
>>> use IT_DB
switched to db IT_DB
```

การสร้างฐานข้อมูล

นอกจากนี้ ถ้าต้องการทราบว่าฐานข้อมูลปัจจุบันที่กำลังทำงานอยู่ ณ ปัจจุบันคือฐานข้อมูลอะไรจะอาศัยคำสั่ง “**db**”

```
>>> db
IT_DB
```

และคำสั่ง “**show dbs**” สำหรับดูว่าใน MongoDB มีฐานข้อมูลกี่รายการและคืออะไรบ้าง

```
>>> show dbs
local      0.78125GB
test       0.23012GB
```

การลบฐานข้อมูล

การลบฐานข้อมูลใน MongoDB จะอาศัยคำสั่ง “**db.dropDatabase()**” สำหรับลบฐานข้อมูลที่อยู่ในปัจจุบัน

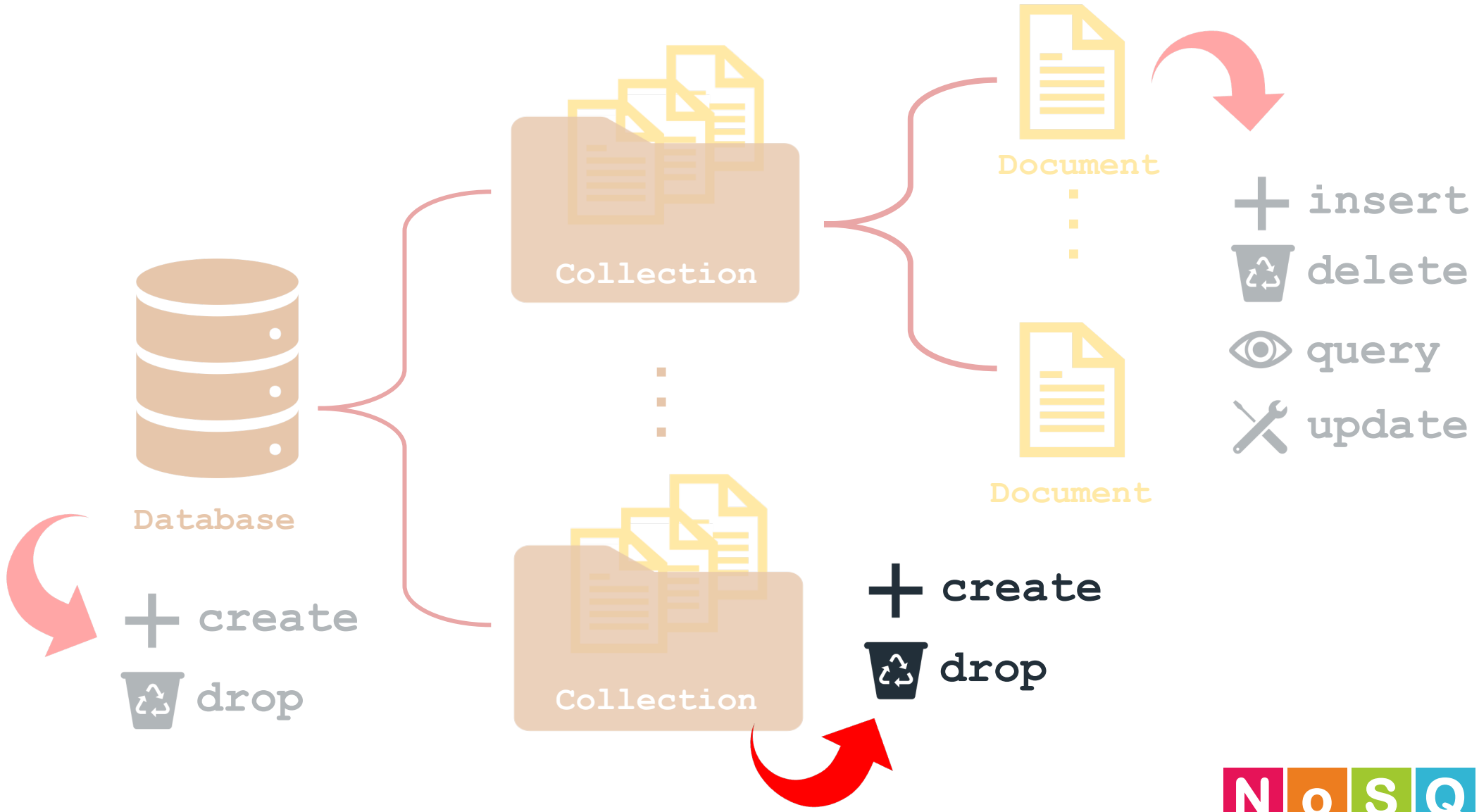
```
db.dropDatabase()
```

ตัวอย่างเช่น

```
>>> db
myTest

>>> db.dropDatabase()
{ "dropped" : "myTest", "ok" : 1 }
```

คำสั่งและตัวดำเนินการของ MongoDB



การสร้าง Collection

การสร้าง Collection ใน MongoDB จะอาศัยคำสั่ง “**db.createCollection("COLLECTION_NAME")**” โดยที่ **COLLECTION_NAME** แทนชื่อ Collection ที่ต้องการสร้าง

```
db.createCollection("mycollection")
```

นอกจากนี้ ถ้าต้องการตรวจสอบว่าในฐานข้อมูลนี้มี Collection อะไรบ้างจะอาศัยคำสั่ง “**show collections**”

```
show collections
```

การสร้าง Collection

ตัวอย่าง

```
>>> db.createCollection("WizardCollection")
{ "ok" : 1 }

>>> show collections
Wizard
WizardCollection
```

การลบ Collection

การลบ Collection ใน MongoDB จะอาศัยคำสั่ง “`db.COLLECTION_NAME.drop()`” โดยที่ `COLLECTION_NAME` แทนชื่อ Collection ที่ต้องการสร้าง

```
db.COLLECTION_NAME.drop()
```

ตัวอย่าง

```
>>> show collections
Wizard
WizardCollection

>>> db.WizardCollection.drop()
True

>>> show collections
Wizard
```

การเปลี่ยนชื่อ Collection

การเปลี่ยนชื่อ Collection ใน MongoDB จะอาศัยคำสั่ง “`db.COLLECTION_NAME.renameCollection(“New_Collection_Name”)`
โดยที่ `New_Collection_Name` แทนชื่อ Collection ใหม่และ `COLLECTION_NAME` แทนชื่อ Collection เดิม

```
db.COLLECTION_NAME.renameCollection(“New_Collection_Name”)
```

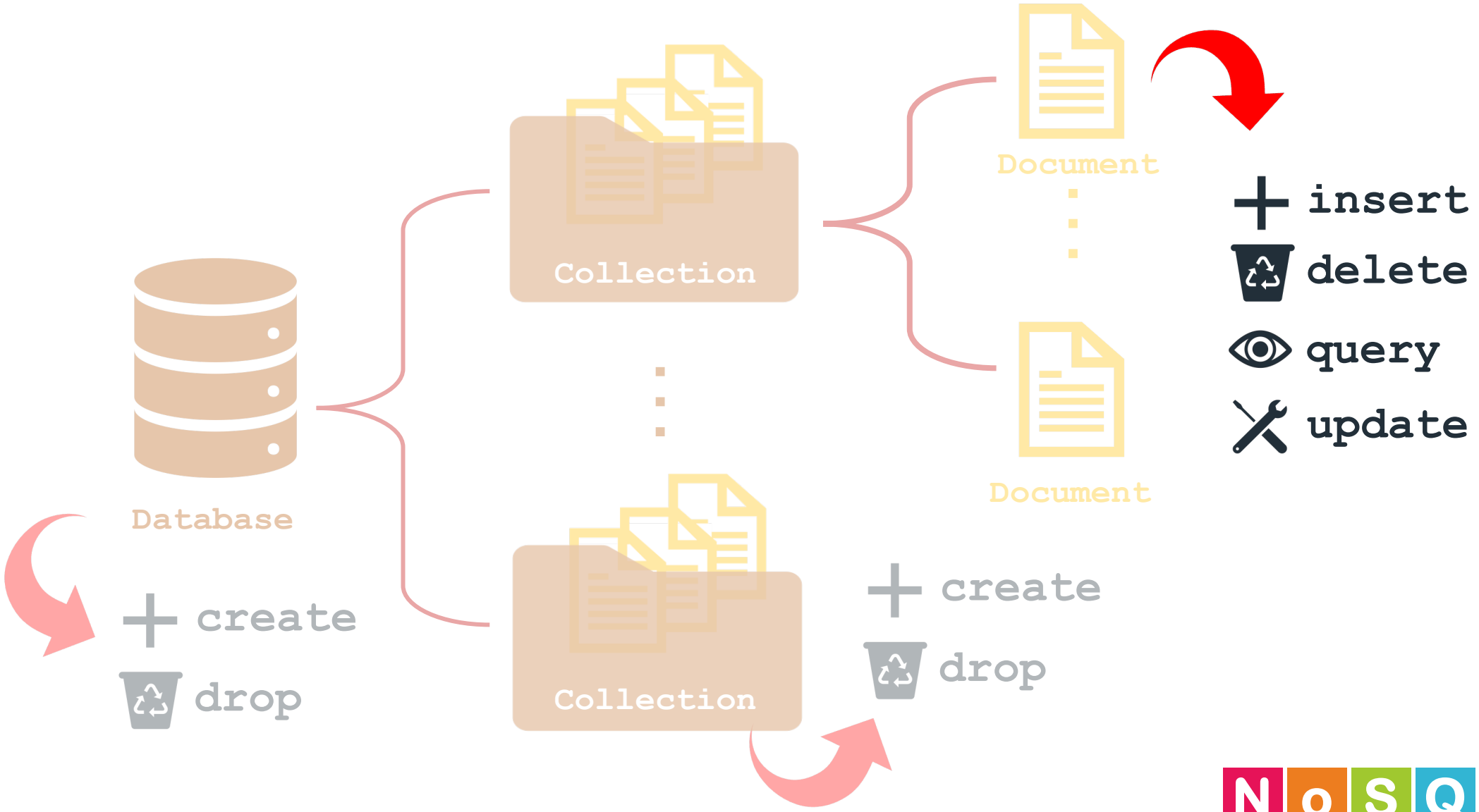
ตัวอย่าง

```
>>> show collections
Wizard
WizardCollection

>>> db.WizardCollection.renameCollection(“Wiz”)
{“ok”:1}

>>> show collections
Wizard
Wiz
```

คำสั่งและตัวดำเนินการของ MongoDB



การเพิ่ม Document ลงในฐานข้อมูล

การเพิ่ม Document ลงในฐานข้อมูล

การเพิ่มข้อมูลลงในฐานข้อมูล MongoDB สามารถทำได้ 3 วิธีการ โดยอาศัยคำสั่งดังต่อไปนี้

1
```
db.COLLECTION_NAME.insert(document)

db.COLLECTION_NAME.insert([document1, document2, … ,documentN ])
```

2
```
db.COLLECTION_NAME.insertOne(document)
```

3
```
db.COLLECTION_NAME.insertMany([document1, document2, … ,documentN ])
```

ซึ่ง document คือ ข้อมูลในรูปแบบ JSON โดยที่ ตัวนอกสุดต้องเป็น Object

การเพิ่ม Document ลงในฐานข้อมูล

คำสั่ง “**db.COLLECTION_NAME.insert(document)**” จะคืนค่ามาเป็นวัตถุ (document) ถึงแม้ว่าการเพิ่มข้อมูลจะสำเร็จหรือไม่สำเร็จก็ตาม ซึ่งค่อนข้างจะไม่สะดวกในทางปฏิบัติในกรณีที่ต้องการจัดการหรือตรวจสอบข้อผิดพลาด นอกจากนี้ คำสั่ง **insert** ยังสามารถรับ document ได้ครั้งละ 1 ถึง N documents

ตัวอย่างเช่น

1

เรียกเป็นๆ 1 document

```
>>> db.Wizard.insert({
    name: "Severus Snape",
    school: "Hogwarts",
    house: "Slytherin",
    pets: ["dog", "frog", "rat"],
    money: 1500,
    position: "teacher"
})
```

```
>>> db.Wizard.insert([
{ "name" : "Harry Potter",
  "school" : "Hogwarts"
}, {
  "name" : "Timmy Catty",
  "school" : "Hogwarts"}
])
```

เวลาพิมพ์ใน *command line* ในส่วนของ *key* อาจจะมี “” หรือไม่มีก็ได้

การเพิ่ม Document ลงในฐานข้อมูล

นอกจากนี้ ถ้าต้องการใส่ข้อมูลหลากหลาย document ลงในคำสั่ง Query เดียวก็สามารถทำได้ในรูปแบบ Array

ตัวอย่างเช่น

```
>>> db.Wizard.insert([{
      _id: ObjectId(),
      name: "Severus Snape",
      school: "Hogwarts",
      house: "Slytherin",
      pets: ["dog", "frog", "rat"],
      money: 1500,
      position: "teacher"
},{
      _id: ObjectId(),
      name: "Harry Potter",
      school: "Hogwarts",
      house: "Gryffindor",
      pets: ["cat", "bird"],
      money: 1000,
      position: "student"
}])
```

การเพิ่ม Document ลงในฐานข้อมูล

ขณะที่ คำสั่ง “**db.COLLECTION_NAME.insertOne(document)**” จะคืนค่ามาเป็นวัตถุ (document) กรณีที่เพิ่มข้อมูลสำเร็จเท่านั้น แต่จะคืนค่า Exception ให้กรณีที่ไม่สำเร็จ ซึ่งค่อนข้างสะดวกกับการจัดการหรือตรวจสอบข้อผิดพลาด แต่อย่างไรก็ตาม คำสั่ง **insertOne** สามารถรับ document ได้ครั้งละ 1 document เท่านั้น

ตัวอย่างเช่น

2

```
>>> db.Wizard.insertOne({      return ออกมาเป็น exception object
    name: "Severus Snape",
    school: "Hogwarts",
    house: "Slytherin",
    pets: ["dog", "frog", "rat"],
    money: 1500,
    position: "teacher”
})
```

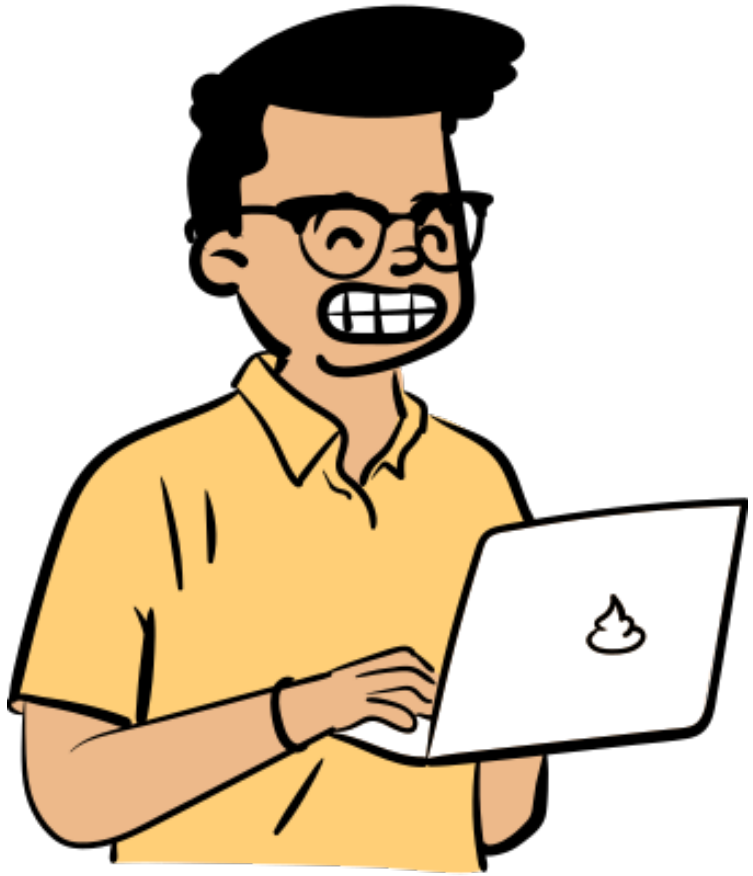
การเพิ่ม Document ลงในฐานข้อมูล

นอกจากนี้ ยังมีคำสั่ง “**db.COLLECTION_NAME.insertMany(document)**” จะคืนค่าเหมือน **insertOne** และสามารถรับ document ได้ครั้งละมากกว่า 1 document เท่านั้น

ตัวอย่างเช่น

3

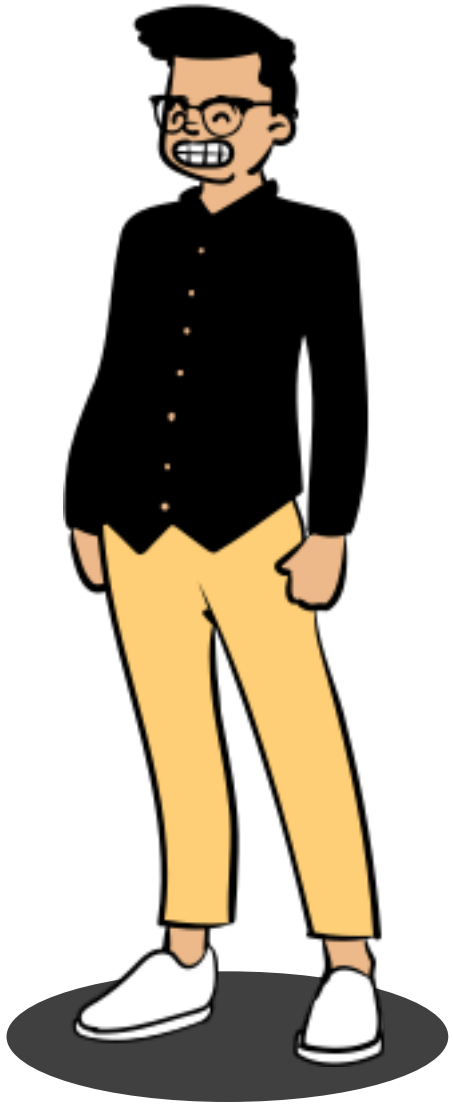
```
>>> db.Wizard.insertMany([
{ “name” : “Harry Potter”,
  “school” : "Hogwarts"
}, {
  “name” : “Timmy Catty”,
  “school” : "Hogwarts"}
])
```



ฐานข้อมูล MongoDB จะเก็บข้อมูลแต่ละ Document ไว้ในรูปแบบการเข้ารหัสข้อมูลไบนารี่ของ JSON หรือเรียกว่า Binary encoded JSON (BSON) ซึ่งออกแบบมาให้มีขนาดเล็ก ทำให้สามารถอ่านและเขียนได้อย่างมีประสิทธิภาพและรวดเร็ว

BSON ยังออกแบบอยู่บนหลักการ Schema-less Database กล่าวคือ เป็นฐานข้อมูลที่สามารถเพิ่มลดคีย์หรือแอททริบิวได้ตลอด ไม่จำเป็นต้องกำหนดโครงสร้าง ทำให้มีความยืดหยุ่นสูง นอกจากนี้ แต่ละ Document ไม่จำเป็นต้องมีคีย์หรือแอททริบิวที่เท่ากันและเหมือนกัน

ชนิดข้อมูล (Data Type) ของ BSON ใน MongoDB



mongoDB Data Type

ยังไม่กำหนดค่า แต่รู้ datatype

Null

Null เป็นชนิดข้อมูลที่จัดการค่า *null* เท่านั้น

ยังไม่กำหนดค่า ไม่รู้ datatype ด้วย

Undefined

Undefined เป็นการไม่กำหนดชนิดข้อมูล

Symbol

Symbol มีลักษณะคล้ายคลึงกับ String
แต่จะใช้รองรับภาษาที่มีการใช้สัญลักษณ์จำเพาะ

Other

String

Strings จะอยู่ในรูปแบบ UTF-8 ซึ่งสะดวกกับภาษาโปรแกรมมิ่งอื่น ๆ เวลาเรียกใช้งาน (serializing และ deserializing)

Date

Date จะเก็บวันที่และเวลาในรูปแบบ UNIX ณ ปัจจุบัน

วันและเวลา

Integer & Double

Integer อาจจะจัดเก็บแบบ 32 หรือ 64-bit ขึ้นอยู่กับเครื่องผู้ให้บริการ ขณะที่ Double ใช้เก็บเลขทศนิยม

Object

Object จะนิยมใช้งานแบบ Embedded Document ในรูปแบบ key-value

Boolean

Boolean ใช้เก็บค่าความจริง ได้แก่ *true* หรือ *false*

t เล็ก f เล็ก

Array

Array จะเป็นการเก็บข้อมูลแบบหนึ่งคีย์แต่หลากหลายค่าต่อหนึ่งคีย์

Timestamp

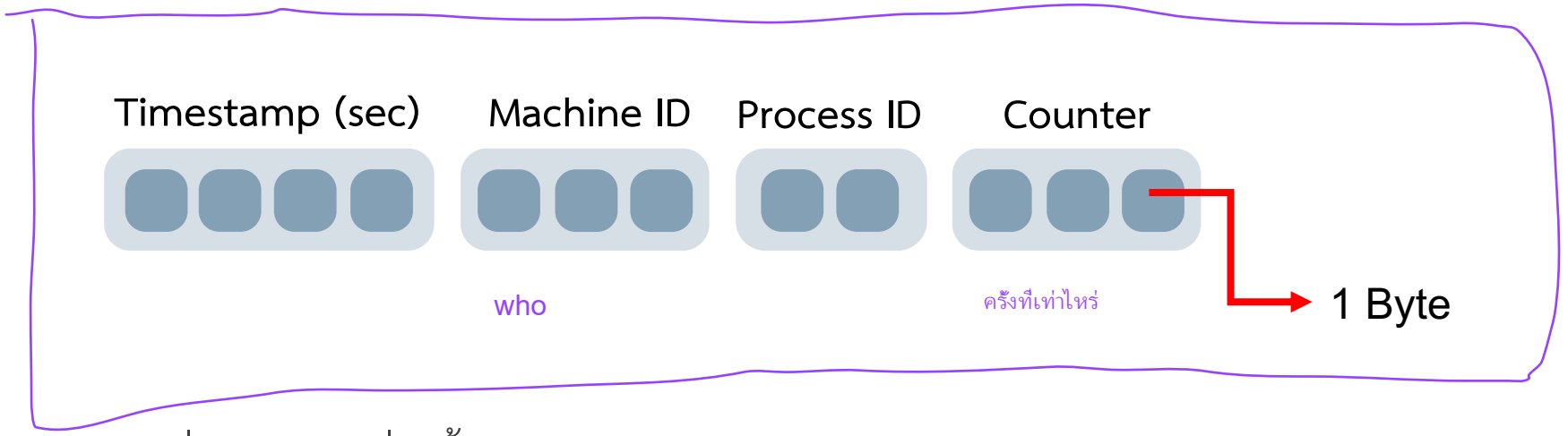
Object จะนิยมใช้งานแบบ Embedded Document ในรูปแบบ key-value

ObjectId

ObjectId() จะเป็นค่าเฉพาะที่ไม่ซ้ำ มีขนาดเล็ก และสามารถสร้างได้อย่างรวดเร็ว

วันและเวลา+id+process id --> มีข้อมูลอย่างอื่นนอกเหนือจากเวลาเก็บไว้ด้วย
ไม่ละ จารย์บอกพุดผิด หมายถึง object id

ObjectId



`ObjectId` จะเป็นชนิดข้อมูลที่มีค่าเฉพาะที่ไม่ซ้ำ ขนาดเล็ก และสามารถสร้างได้อย่างรวดเร็วในรูปแบบเลขฐาน 16 ถ้าเอกสารใด ๆ ได้เพิ่มลงใน collection โดยปราศจากช่อง **`_id`** แล้ว MongoDB จะสร้างให้อัตโนมัติ อย่างไรก็ตาม ผู้ใช้ยังคงสามารถกำหนด _id เองได้

ตัวอย่างที่ 1

```
>>> var id = ObjectId()
>>> db.MyDB.insert({_id:id, Name: "Sirius Black"})
WriteResult({"nInserted":1})
>>> db.MyDB.find().pretty()
{
        "_id" : ObjectId("55c1f00724929f1eea5dcf62"),
        "Name" : "Sirius Black"
}
```

ตัวอย่างที่ 2

```
>>> db.MyDB.insert({_id:ObjectId(), Name: "Sirius Black"})
WriteResult({"nInserted":1})
>>> db.MyDB.find().pretty()
{
        "_id" : ObjectId("55c1f00724929f1eea5dcf62"),
        "Name" : "Sirius Black"
}
```

Date

Date เป็นชนิดข้อมูลที่เก็บวันที่และเวลาในรูปแบบ UNIX ณ ปัจจุบัน ซึ่งจัดเก็บในรูปแบบเลขจำนวนเต็ม 64 บิต คือ หน่วยของมิลลิวินาที โดยที่ MongoDB จะมีเมธอดเกี่ยวกับเวลาดังตารางต่อไปนี้

เมธอด	คำอธิบาย
Date()	วันที่และเวลาในรูปแบบข้อความ “String”
New Date()	วันที่และเวลาในรูปแบบวัตถุของเวลา “Date Object” ในรูปแบบนี้สามารถนำเมธอดเฉพาะมาเรียกใช้คุณลักษณะได้โดยตรง อาทิเช่น getMonth()
ISODate()	

ตัวอย่างที่ 1

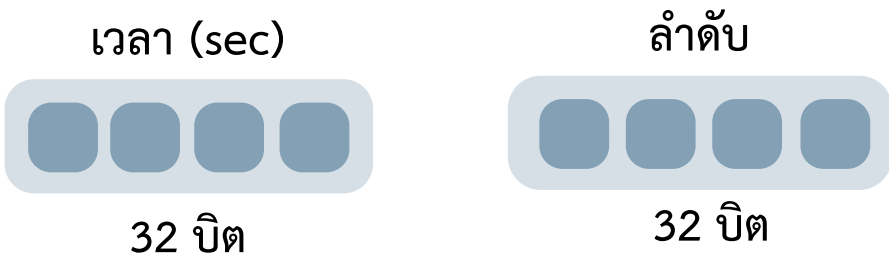
```
>>> var d1 = Date()
>>> var d2 = new Date()
>>> var d3 = ISODate()
>>> var d4 = ISODate("2018-08-16T06:01:17.171z")
>>> db.MyDB.insert({_id:ObjectId(), date1:d1, date2:d2, date3:d3, date4:d4})
WriteResult({"nInserted":1})
>>> db.MyDB.find().pretty()
{
        "_id" : ObjectId("55c1f00724929f1eea5dcf62"),
        "date1" : "Tue May 08 1990 06:30:33 GMT+0700 (Thailand Standard Time)",
        "date2" : ISO("1990-05-08T06:30:33.521z"),
        "date3" : ISO("1990-05-08T06:30:35.781z"),
        "date4" : ISODate("2018-08-16T06:01:17.171z")
}
```

ตัวอย่างที่ 2

```
>>> db.MyDB.insert({date1:new Date(), date2:Date()})
WriteResult({"nInserted":1})
```

Timestamp

เวลา + ลำดับ แต่ไม่ได้เก็บ machine id / process id (ตอนแรกจารย์พูดผิด)



Date เป็นชนิดข้อมูลที่จัดเก็บเวลาในรูปแบบ UNIX ณ เวลาปัจจุบันร่วมกับลำดับของข้อมูล ซึ่งจัดเก็บในรูปแบบเลขจำนวนเต็ม 64 บิต ซึ่งเริ่มนับแต่ละวินาทีตั้งแต่ 01/01/1970 สิ่งนี้มีประโยชน์ในการตรวจสอบเวลาที่เอกสารนั้นมีการเพิ่มหรือแก้ไขใน collection

อย่างไรก็ตาม ชนิดข้อมูล **`ObjectId`** มีการจัดเก็บ Timestamp ไว้อยู่แล้ว (เก็บ ณ ตอนสร้างเท่านั้น ถ้ามีการอัพเดตในภายหลังค่า timestamp จะไม่มีการแก้ไข) ดังนั้น เมื่อต้องการเรียกใช้ค่า Timestamp ที่จัดเก็บใน **`ObjectId`** ก็สามารถเรียกใช้งานผ่านเมธอด “**`getTimestamp()`**”

ตัวอย่างที่ 1

```
>>> var ts = new Timestamp()
Timestamp(0, 0)
>>> db.MyDB.insert({_id:ObjectId(), ID:931, TStamp:ts})
WriteResult({"nInserted":1})
>>> db.MyDB.find().pretty()
{
        "_id" : ObjectId("55c1f00724929f1eea5dcf62"),
        "ID" : 931,
        "TStamp" : Timestamp(1438791098, 1)
}
```

The order of operation

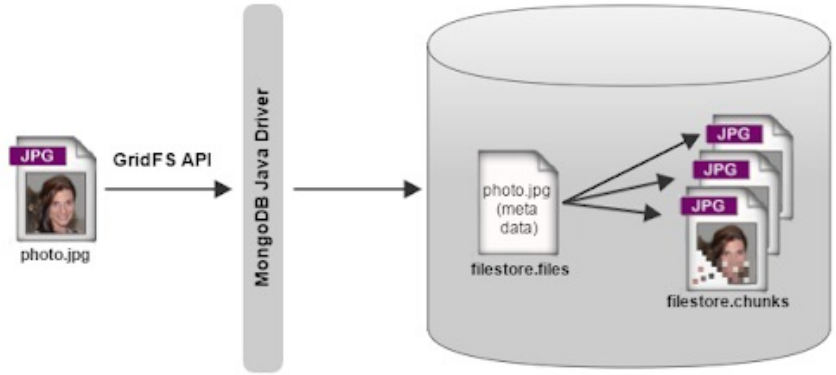
Current timestamp

ตัวอย่างที่ 2

```
>>> db.MyDB.insert({_id:ObjectId(), ID:931})
WriteResult({"nInserted":1})
>>> db.MyDB.find().pretty()
{
        "_id" : ObjectId("55c1f00724929f1eea5dcf62"),
        "ID" : 931
}
>>> var t = ObjectId("55c1f00724929f1eea5dcf62").getTimestamp()
>>> t
ISO("1990-05-08T06:30:35.781z")
```

ข้อจำกัด

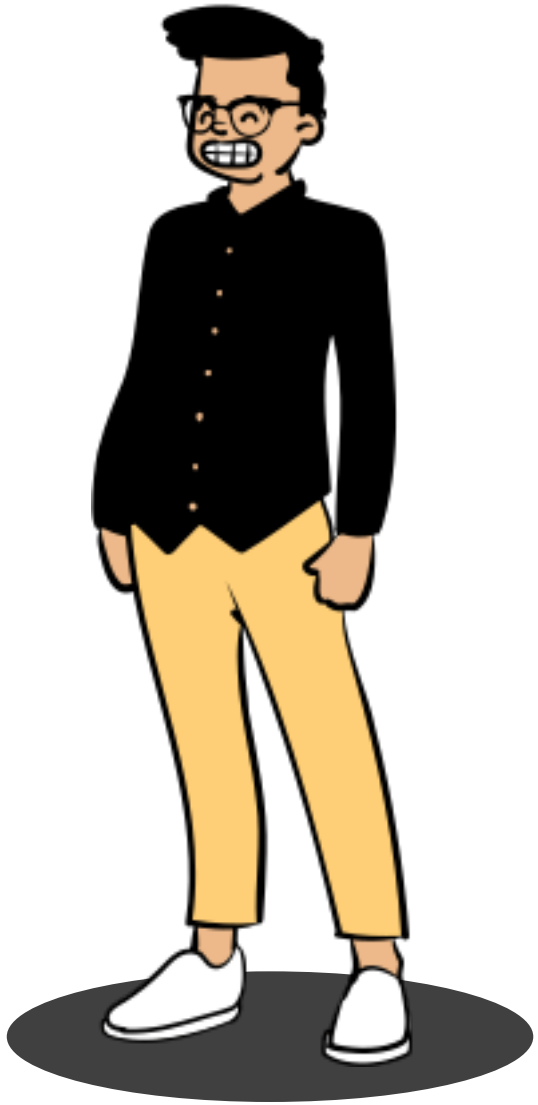
Document ใน MongoDB ต้องมีขนาดไม่เกิน 16 MB ตามขนาดของ BSON ถ้าไฟล์มีขนาดใหญ่เกิน MongoDB จะใช้หลักการเก็บแบบ GridFS แทน ซึ่งเป็นวิธีการเก็บข้อมูลโดยการแบ่งไฟล์ Document ออกเป็น Binary Chunk ขนาดเล็กหลาย ๆ ไฟล์ โดยที่แต่ละไฟล์มีขนาดประมาณ 256 KB จากนั้นทำการบันทึกแต่ละ Chunk เป็นแต่ละ Document กล่าวคือ 1 Binary Chunk ต่อ 1 Document



MongoDB GridFS Structure

นอกจากนี้ mongoDB รองรับ Document ที่มีการเก็บซ้อนกันได้ไม่เกิน 100 ระดับ

การเรียกดู Document ในฐานข้อมูลเบื้องต้น



การเรียกดู Document ในฐานข้อมูล

การเรียกดูข้อมูลจาก collection ในปัจจุบันจากฐานข้อมูล MongoDB จะอาศัยคำสั่ง “**db.COLLECTION_NAME.find()**” โดยคำสั่งดังกล่าวจะคืนค่ามาเป็นทุก document ในแบบไม่มีโครงสร้าง

```
db.COLLECTION_NAME.find()
```

= select * from COLLECTION_NAME ใน sql

ตัวอย่างเช่น

```
>>> db.Wizard.find()
{ "_id" : ObjectId("5d1725a7e11400ac609de718"), "name" : "Severus Snape", "school" :
"Hogwarts", "house" : "Slytherin", "pets" : [ "dog", "frog", "rat" ], "money" : 1500,
"position" : "teacher" }
{ "_id" : ObjectId("5d1725cae11400ac609de719"), "name" : "Harry Potter", "school" :
"Hogwarts", "house" : "Gryffindor", "pets" : [ "cat", "bird" ], "money" : 1000,
"position" : "student" }
```

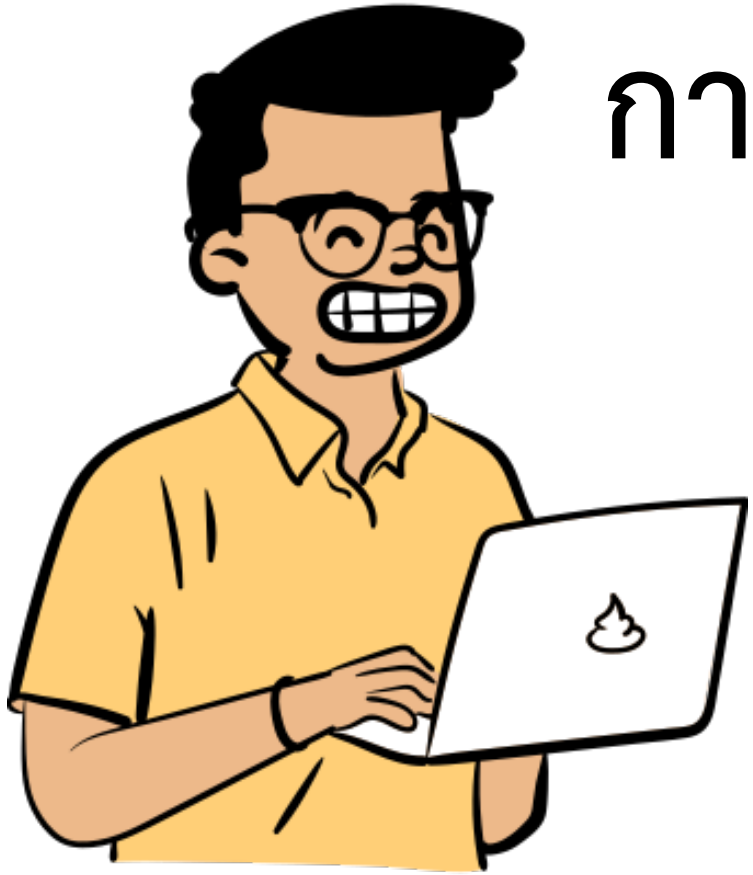
การเรียกดู Document ในฐานข้อมูล

เพื่อให้การเรียกดูข้อมูลอยู่ในรูปแบบที่ง่ายและสะดวกต่อการทำความเข้าใจของมนุษย์ ดังนั้น จะอาศัยคำสั่ง “**pretty()**” ร่วมกับ “**find()**” เสมอ

```
db.COLLECTION_NAME.find().pretty()
```

ตัวอย่างเช่น

```
>>> db.Wizard.find().pretty()
{
        "_id" : ObjectId("5d1725cae11400ac609de719"),
        "name" : "Harry Potter",
        "school" : "Hogwarts",
        "house" : "Gryffindor",
        "pets" : ["cat", "bird"],
        "money" : 1000,
        "position" : "student"
}
```



การกำหนดแอททริบิวที่จะแสดงผลของแต่ละ Document

การกำหนดแอททริบิวที่จะแสดงผล

เมื่อพิจารณาของคำสั่ง SQL ต่อไปนี้

```
SELECT id, user_id, status FROM tbl_users
```

จะพบว่า id, user_id, status คือการบ่งบอกว่าจะให้แสดงแอททริบิวตัวใดออกทางหน้าจอบ้าง ซึ่งใน MongoDB สามารถทำได้ด้วยคำสั่ง find() ซึ่งประกอบไปด้วย 2 พารามิเตอร์หลัก ได้แก่ {condition} และ {display} โดยที่ {display} มีหน้าที่หลักสำหรับกำหนดแอททริบิวที่จะแสดงผล

```
db.ColectionName.find({condition},{display})
```

การกำหนดแอททริบิวที่จะแสดงผล

ตัวอย่างที่ 1

condition where ถ้าจะไม่เอาเงื่อนไขจะใส่ {}

```
db.tbl_users.find({},{user_id: 1, status: 1})
```

จะพบว่าฐานข้อมูล MongoDB จะแสดงผลเพียง _id, user_id และ status เนื่องจาก user_id กับ status ถูกกำหนดเป็น 1 นั้นหมายความว่าให้แสดงผล ขณะที่ _id เป็น Primary key ถ้าไม่กำหนดถือว่าให้แสดงผลด้วย (default กำหนดให้ _id:1)

mongo add attribute _id มาให้ return ค่าได้เป็นๆ obj
แต่ถ้ามี _id เป็นๆ attribute อยู่แล้ว mongo จะไม่สร้างมาให้

ตัวอย่างที่ 2

```
db.tbl_users.find({},{_id:0, user_id: 1, status: 1})
```

จะพบว่าจะแสดงผลเพียง user_id และ status เท่านั้น

_id ถ้าไม่กำหนดค่าจะdefault เป็น 1 (แต่ตัวอื่นจะเป็นoหมด)

การกำหนดแอททริบิวที่จะแสดงผล

ตัวอย่างที่ 1

บ่งบอกว่าไม่มีการกำหนดเงื่อนไข เทียบเคียงได้กับไม่กำหนด where ใน SQL

```
db.tbl_users.find({},{user_id: 1, status: 1})
```

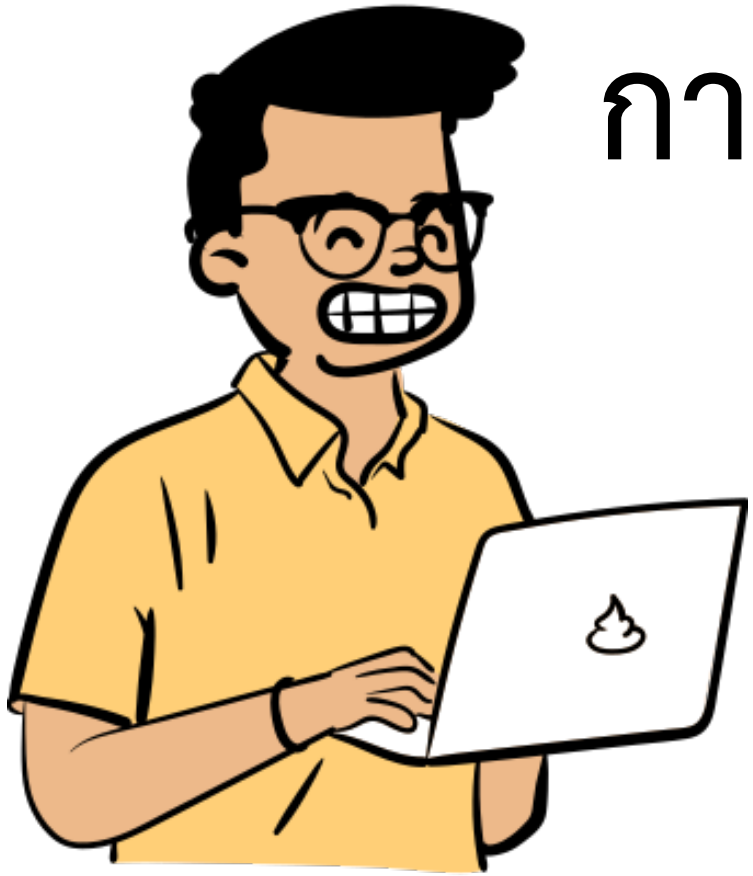
จะพบว่าฐานข้อมูล MongoDB จะแสดงผลเพียง _id, user_id และ status เนื่องจาก user_id กับ status ถูกกำหนดเป็น 1 นั้นหมายความว่าให้แสดงผล ขณะที่ _id เป็น Primary key ถ้าไม่กำหนดถือว่าให้แสดงผลด้วย (default กำหนดให้ _id:1)

ตัวอย่างที่ 2

```
db.tbl_users.find({},{_id:0, user_id: 1, status: 1})
```

จะพบว่าจะแสดงผลเพียง user_id และ status เท่านั้น

ใน MongoDB ค่า 0 แทนการ*ไม่แสดง* และเลขอื่น ๆ แทนการ*แสดงผล*



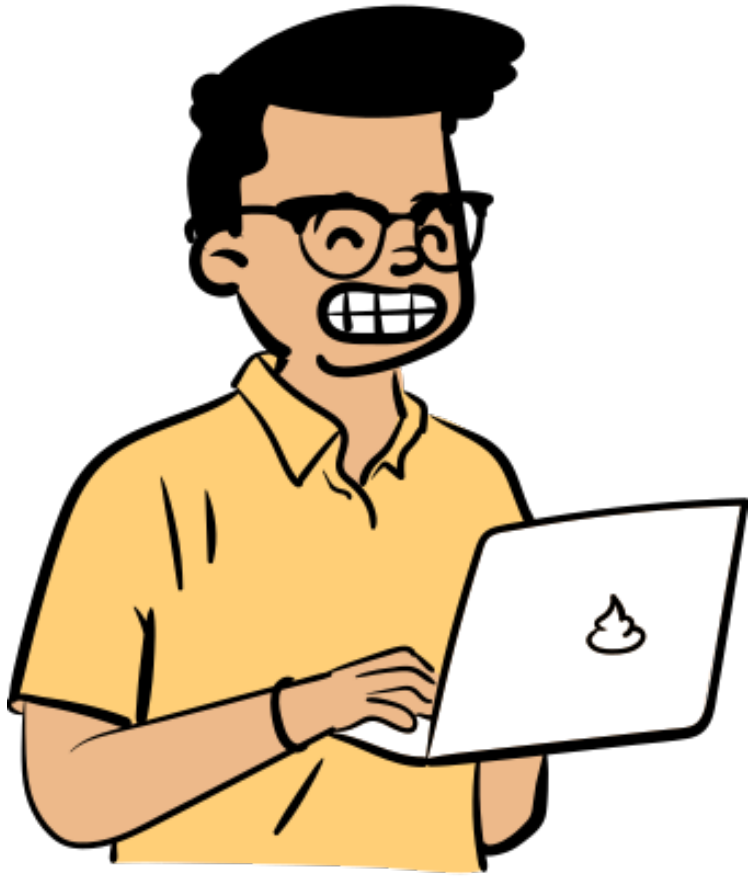
การกำหนดจำนวน Document ที่จะแสดงผล

การกำหนดจำนวน Document ที่จะแสดงผล

การกำหนดจำนวน Document ที่จะแสดงผลสามารถทำได้ โดยอาศัยคำสั่ง limit() เพื่อกำหนดจำนวน Document สูงสุดที่จะคืนค่าให้ เพื่อให้เกิดประสิทธิภาพสูงสุด และป้องการการคืนค่ามากเกินกว่าที่กำหนด แต่ถ้าจำนวน Document ไม่ถึงตามที่กำหนดจะคืนค่า Document ทั้งหมด

```
db.ColectionName.find().limit(N)
```

โดยที่ N แทนจำนวน Document สูงสุดที่จะแสดงผล ซึ่งต้องมีค่ามากกว่า 0 แต่ถ้า N เท่ากับ 0 แสดงว่าไม่มีการจำกัดจำนวน Document ที่จะแสดงผล



การจัดเรียง Document ในฐานข้อมูล

การจัดเรียง Document ในฐานข้อมูล

เมื่อพิจารณาของคำสั่ง SQL ต่อไปนี้

```
SELECT id, user_id, status FROM tbl_users ORDER BY user_id ASC
หรือ
SELECT id, user_id, status FROM tbl_users ORDER BY user_id DESC
```

จะพบว่าลำดับการแสดงผลของแต่ละ Document จะเรียงตาม user_id จากน้อยไปมากกับมากไปน้อย ตามลำดับ ซึ่งใน MongoDB สามารถทำได้ด้วยคำสั่ง sort() ดังตัวอย่างต่อไปนี้

```
db.ColectionName.find({condition},{display}).sort({fieldName: 1 or-1})
```

การจัดเรียง Document ในฐานข้อมูล

ตัวอย่างเช่น

```
SELECT id, user_id, status FROM tbl_users ORDER BY user_id ASC
หรือ
SELECT id, user_id, status FROM tbl_users ORDER BY user_id DESC
```

```
db.tbl_users.find({},{user_id: 1, status: 1}).sort({user_id:1}) // ASC
หรือ
db.tbl_users.find({},{user_id: 1, status: 1}).sort({user_id:-1}) // DESC
```

การจัดเรียง Document ในฐานข้อมูล

ตัวอย่างเช่น

```
>>> > db.my.find({},{_id:0,name:1, age:-2}).sort({age:1,name:1})
{ "name" : "tara", "age" : 9 }
{ "name" : "Taravichet", "age" : 17 }
{ "name" : "alex", "age" : 18 }
{ "name" : "ploy", "age" : 19 }
{ "name" : "Potter", "age" : 29 }
{ "name" : "Potte2r", "age" : 229 }
{ "name" : "Potte3r", "age" : 229 }

> db.my.find({},{_id:0,name:1, age:-2}).sort({age:1,name:-1})
{ "name" : "tara", "age" : 9 }
{ "name" : "Taravichet", "age" : 17 }
{ "name" : "alex", "age" : 18 }
{ "name" : "ploy", "age" : 19 }
{ "name" : "Potter", "age" : 29 }
{ "name" : "Potte3r", "age" : 229 }
{ "name" : "Potte2r", "age" : 229 }
```