

# Introduction to MongoDB

---

2110322 Database Systems

February 2025

(ตั้นฉบับจาก รศ.ดร.ดวงดาว วิชาดาภุล)



## ประเภทหลัก ๆ ของฐานข้อมูลกลุ่ม NoSQL (aka "not only SQL")

- Document Databases (e.g., MongoDB)
- Key-values stores (e.g., Redis)
- Column-oriented databases (e.g., Apache Hbase, Apache Parquet)
- Graph databases (e.g., Neo4j)
- Time series databases (e.g., InfluxDB)

NoSQL คืออะไรก็ตามที่

จัดการฐานข้อมูลได้

นอกจาก SQL

# ແນະໜໍາ MongoDB ເລີກ ຖ້າຍ ຕຸ

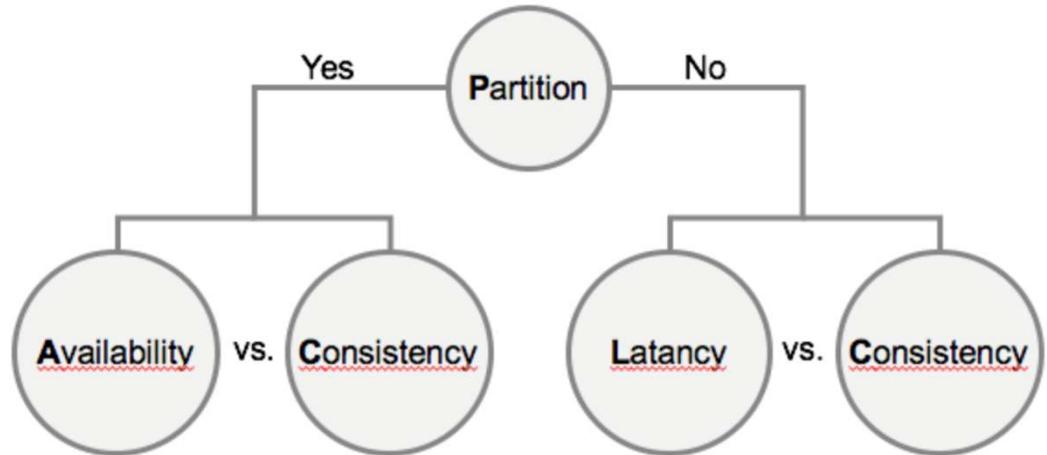
- MongoDB เป็น **schema-less** NoSQL **document** database
- เราสามารถเก็บข้อมูลในรูปแบบ **JSON/BSON** ได้โดยง่าย
- เหมาะกับข้อมูล **unstructured** ข้อมูลที่มีความซับซ้อน หรือที่อาจจะทำให้การกำหนดสกີມາຕັ້ງແຕ່ຕັ້ນທຳໄດ້ຍາກ
- Scale **ໄດ້ດີ** ທີ່ເຮືອງ data volume และ network traffic

# Consistency and Availability of Databases

<https://medium.com/@sumitsethia94/consistency-or-availability-of-databases-do-you-really-understand-cap-8ecf2b3bb099>

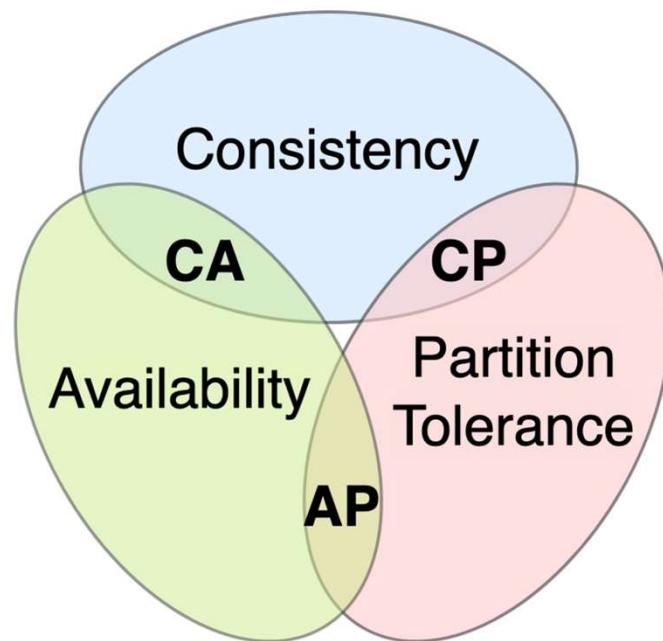
ACID, CAP, PACELC

ทฤษฎี CAP คือเราได้แค่ 2 ใน 3 ของ  
Consistency, Availability, Partition  
เท่านั้น (ไม่ได้ทั้ง 3 พร้อมกัน)

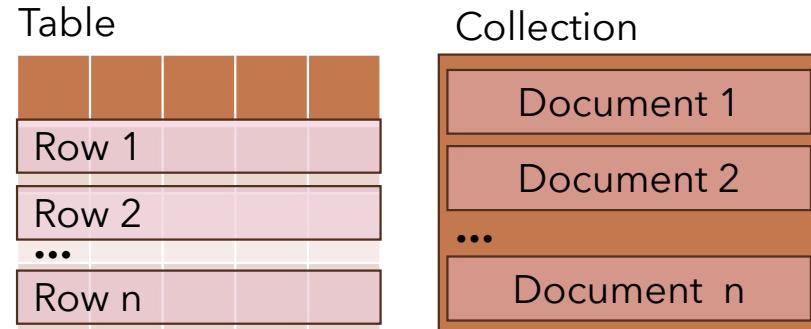


# CAP Theory

เลือกแค่ 2 ใน 3

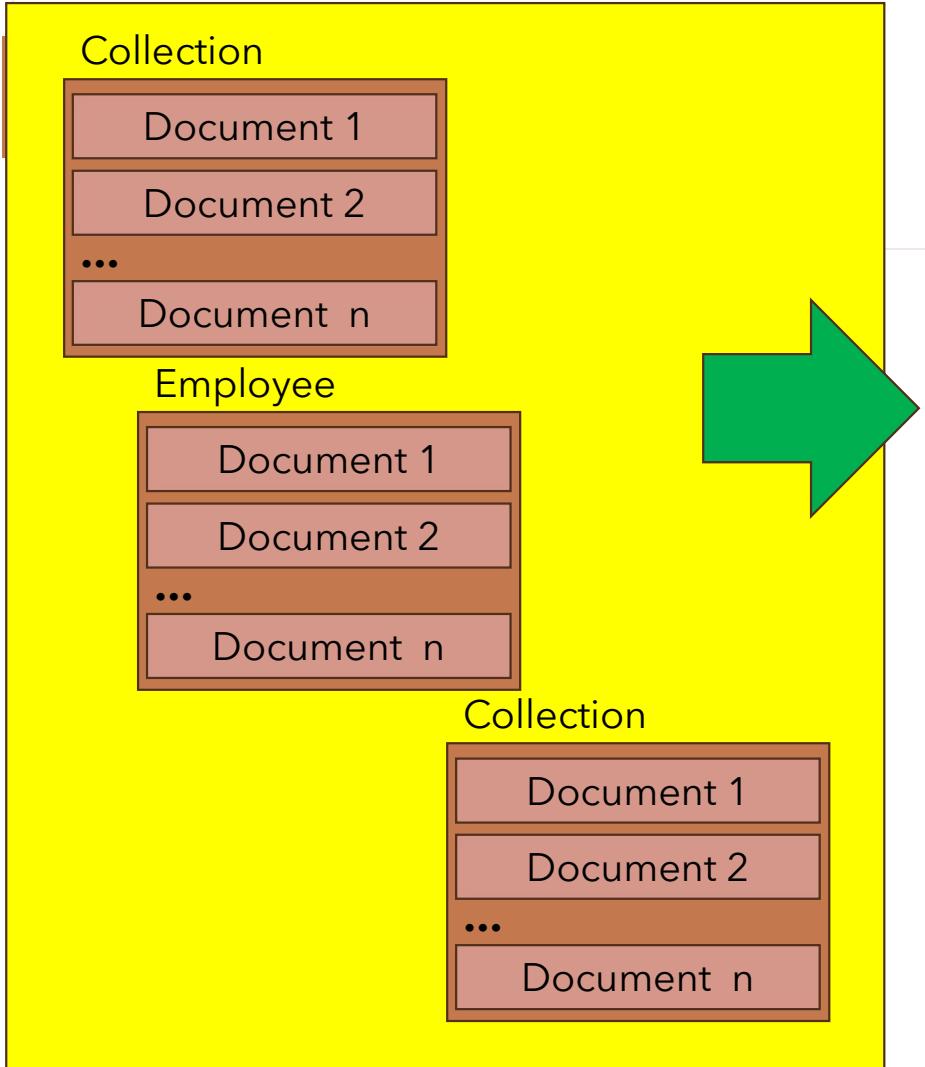


# เปรียบเทียบคำศัพท์ที่ใช้



<https://blog.sqlauthority.com/2020/05/23/mongodb-fundamentals-mapping-relational-sql-day-6-of-6/>

# NoSQL MongoDB Database



## Employee

```
{  
  "name": "Jane Doe",  
  "age": 42,  
}
```

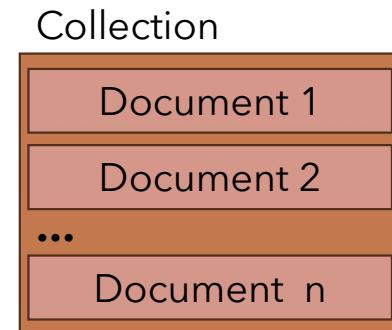
```
{  
  "name": "John Smith",  
  "age": 45,  
}  
...
```

```
{  
  "name": "Susan White",  
  "age": 45,  
}
```

หนึ่ง Collection (table) Employee มี

คล้าย Document (Row)

# Document Schema



- เป็น JSON object ที่ใช้ในการอธิบายองค์ประกอบของเนื้อหาภายใน document และ embedded document ในแต่ละ collection
- เราสามารถใช้ สกีมา ในการระบุ required fields ระบุ content ของ fields และ validate schema เมื่อมีการ update / insert

<https://docs.mongodb.com/realm/mongodb/document-schemas/>

# Filed level schema definition

## ตัวอย่างข้อมูล

```
{  
  "name": "Jane Doe",  
  "age": 42,  
  "favoriteColors": [  
    { "rank": 1, "name": "RebeccaPurple", "hexCode": "#663399" },  
    { "rank": 2, "name": "DodgerBlue", "hexCode": "#1E90FF" },  
    { "rank": 3, "name": "SeaGreen", "hexCode": "#2E8B57" }  
  ]  
}
```

## ตัวอย่างスキมาของข้อมูล

```
{  
  "bsonType": "object",  
  "required": ["name", "age", "favoriteColors"],  
  "properties": {  
    "name": {  
      "bsonType": "string"  
    },  
    "age": {  
      "bsonType": "int",  
      "minimum": 13,  
      "exclusiveMinimum": false  
    },  
    "favoriteColors": {  
      "bsonType": "array",  
      "uniqueItems": true,  
      "items": {  
        "bsonType": "object",  
        "properties": {  
          "rank": { "bsonType": "int" },  
          "name": { "bsonType": "string" },  
          "hexCode": {  
            "bsonType": "string",  
            "pattern": "^(#([A-Fa-f0-9]{6}){6})$"  
          }  
        }  
      }  
    }  
  }  
}
```

<https://docs.mongodb.com/realm/mongodb/enforce-a-document-schema/>

# Filed level schema definition

ตัวอย่างข้อมูล

```
{  
  "name": "Jane Doe",  
  "age": 42,  
  "favoriteColors": [  
    { "rank": 1, "name": "RebeccaPurple", "hexCode": "#663399" },  
    { "rank": 2, "name": "DodgerBlue", "hexCode": "#1E90FF" },  
    { "rank": 3, "name": "SeaGreen", "hexCode": "#2E8B57" }  
  ]  
}
```

ตัวอย่างスキมาของข้อมูล

```
{  
  "bsonType": "object",  
  "required": ["name", "age", "favoriteColors"],  
  "properties": {  
    "name": {  
      "bsonType": "string"  
    },  
    "age": {  
      "bsonType": "int",  
      "minimum": 13,  
      "exclusiveMinimum": false  
    },  
    "favoriteColors": {  
      "bsonType": "array",  
      "uniqueItems": true,  
      "items": {  
        "bsonType": "object",  
        "properties": {  
          "rank": { "bsonType": "int" },  
          "name": { "bsonType": "string" },  
          "hexCode": {  
            "bsonType": "string",  
            "pattern": "^(#([A-Fa-f0-9]{6}){6})$"  
          }  
        }  
      }  
    }  
  }  
}
```

<https://docs.mongodb.com/realm/mongodb/enforce-a-document-schema/>

# Filed level schema definition

ตัวอย่างข้อมูล

```
{  
  "name": "Jane Doe",  
  "age": 42,  
  "favoriteColors": [  
    { "rank": 1, "name": "RebeccaPurple", "hexCode": "#663399" },  
    { "rank": 2, "name": "DodgerBlue", "hexCode": "#1E90FF" },  
    { "rank": 3, "name": "SeaGreen", "hexCode": "#2E8B57" }  
  ]  
}
```

ตัวอย่างスキมาของข้อมูล

```
{  
  "bsonType": "object",  
  "required": ["name", "age", "favoriteColors"],  
  "properties": {  
    "name": {  
      "bsonType": "string"  
    },  
    "age": {  
      "bsonType": "int",  
      "minimum": 13,  
      "exclusiveMinimum": false  
    },  
    "favoriteColors": {  
      "bsonType": "array",  
      "uniqueItems": true,  
      "items": {  
        "bsonType": "object",  
        "properties": {  
          "rank": { "bsonType": "int" },  
          "name": { "bsonType": "string" },  
          "hexCode": {  
            "bsonType": "string",  
            "pattern": "^(#([A-Fa-f0-9]{6}){6})$"  
          }  
        }  
      }  
    }  
  }  
}
```

<https://docs.mongodb.com/realm/mongodb/enforce-a-document-schema/>

# Filed level schema definition

ตัวอย่างข้อมูล

```
{  
  "name": "Jane Doe"  
  "age": 42,  
  "favoriteColors": [  
    { "rank": 1, "name": "RebeccaPurple", "hexCode": "#663399" },  
    { "rank": 2, "name": "DodgerBlue", "hexCode": "#1E90FF" },  
    { "rank": 3, "name": "SeaGreen", "hexCode": "#2E8B57" }  
  ]  
}
```

ตัวอย่างスキมาของข้อมูล

```
{  
  "bsonType": "object",  
  "required": ["name", "age", "favoriteColors"],  
  "properties": {  
    "name": {  
      "bsonType": "string"  
    },  
    "age": {  
      "bsonType": "int",  
      "minimum": 13,  
      "exclusiveMinimum": false  
    },  
    "favoriteColors": {  
      "bsonType": "array",  
      "uniqueItems": true,  
      "items": {  
        "bsonType": "object",  
        "properties": {  
          "rank": { "bsonType": "int" },  
          "name": { "bsonType": "string" },  
          "hexCode": {  
            "bsonType": "string",  
            "pattern": "^(#([A-Fa-f0-9]{6}){6})$"  
          }  
        }  
      }  
    }  
  }  
}
```

<https://docs.mongodb.com/realm/mongodb/enforce-a-document-schema/>

# Filed level schema definition

ตัวอย่างข้อมูล

```
{  
  "name": "Jane Doe",  
  "age": 42,  
  "favoriteColors": [  
    { "rank": 1, "name": "RebeccaPurple", "hexCode": "#663399" },  
    { "rank": 2, "name": "DodgerBlue", "hexCode": "#1E90FF" },  
    { "rank": 3, "name": "SeaGreen", "hexCode": "#2E8B57" }  
  ]  
}
```

ตัวอย่างスキมาของข้อมูล

```
{  
  "bsonType": "object",  
  "required": ["name", "age", "favoriteColors"],  
  "properties": {  
    "name": {  
      "bsonType": "string"  
    },  
    "age": {  
      "bsonType": "int",  
      "minimum": 13,  
      "exclusiveMinimum": false  
    },  
    "favoriteColors": {  
      "bsonType": "array",  
      "uniqueItems": true,  
      "items": {  
        "bsonType": "object",  
        "properties": {  
          "rank": { "bsonType": "int" },  
          "name": { "bsonType": "string" },  
          "hexCode": {  
            "bsonType": "string",  
            "pattern": "^(#([A-Fa-f0-9]{6}){6})$"  
          }  
        }  
      }  
    }  
  }  
}
```

<https://docs.mongodb.com/realm/mongodb/enforce-a-document-schema/>

# Common types in MongoDB



- Object
- Array
- String
- Number
- Boolean
- UUID (16 bytes)
- ObjectId
- Binary Data
- Mixed
- Sets
- Dictionaries

<https://docs.mongodb.com/realm/mongodb/document-schemas/>

# Validation based-on defined schema

```
{  
  "properties":{  
    "_id": { "bsonType": "objectId" },  
    "name": { "bsonType": "string" }  
  }  
}
```

จะมีการ Update JSON Object ด้วย  
การ set name ให้เป็นค่าใหม่คือ 42 ซึ่ง  
name ควรเป็น string

```
collection.updateOne(  
  { "_id": BSON.ObjectId("5ae782e48f25b9dc5c51c4d0") },  
  { "$set": { "name": 42 } }  
)
```

**WHY not correct?**

<https://docs.mongodb.com/realm/mongodb/document-schemas/>

# ลูกค้าต้องการทำเว็บไซต์ (HIVE) ให้ผู้ใช้มาสร้างคอร์สสอนไลน์ ผ่านโพสของตัวเองโดยต้องการเก็บข้อมูลต่อไปนี้

Case study ทำ website ชื่อ HIVE

- ทุกโพส (post) ต้องมี (1) ชื่อคอร์ส (2) คำอธิบาย (3) ชื่อเจ้าของคอร์ส (4) URL ของคอร์ส และ (5) จำนวนไลค์ (likes)
- ทุกโพสสามารถใส่แท็ก (tag) ได้หลาย ๆ แท็ก
- ทุกโพสอาจมีคอมเม้นต์ (comments) จากผู้ใช้คนอื่น ๆ โดยแต่ละ comment มี (1) ชื่อคนคอมเม้นต์ (2) วัน-เวลาที่คอมเม้นต์ (3) ตัวเนื้อความคอมเม้นต์ และ (4) จำนวนไลค์ของคอมเม้นต์

ถ้าต้องออกแบบスキีมาของ RDBMS จะมีกี่ตาราง?

# HiVE's ER Diagram



## HiVE's RDBMS schema design w/ referential integrity

tag

<u>pid</u>	<u>tag_id</u>	tag_message
------------	---------------	-------------

post

<u>pid</u>	title	description	owner	likes
------------	-------	-------------	-------	-------

comments

<u>pid</u>	<u>comment_id</u>	user	message	dateCreated	likes
------------	-------------------	------	---------	-------------	-------

จากความรู้ที่ผ่านมา เราสามารถออกแบบ ERD

ได้ตามนี้และมี Schema ตามนี้

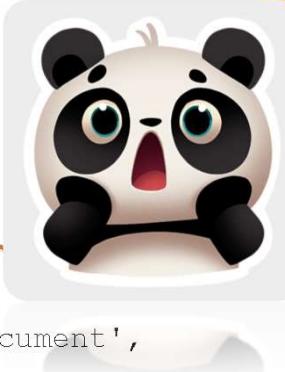
แต่ใน MongoDB เราอาจจะรวมตารางเป็น

หนึ่งเดียวเลย ไม่ใช้กฎที่เรียนมาใน SQL

## HiVE's MongoDB schema design (Only one collection “post”)

### Post collection

```
{  
    title: 'MongoDB Overview',  
    description: 'An intro to MongoDB/document',  
    owner: 'lionking',  
    tags: ['mongodb', 'database', 'NoSQL'],  
    likes: 340  
    comments: [  
        {  
            user: 'hikaru',  
            message: 'nice intro',  
            dateCreated: '2021-10-10',  
            likes: 5  
        },  
        {  
            user: 'miko',  
            message: 'thanks',  
            dateCreated: '2020-11-2',  
            likes: 4  
        } , ...  
    ]  
}
```



แทนการแตกต่างในตัวอย่างนี้เรารอออกแบบสกีมา  
จากการ embed ข้อมูลที่เกี่ยวข้องเข้ามาอยู่ร่วมกัน

```
{  
    "bsonType": "object",  
    "required": ["title", "owner"],  
    "properties": {  
        "title": {  
            "bsonType": "string"  
        },  
        "owner": {  
            "bsonType": "string",  
        },  
        ...  
        "comments": {  
            "bsonType": "array",  
            "uniqueItems": true,  
            "items": {  
                "bsonType": "object",  
                "properties": {  
                    "user": { "bsonType": "objectId"},  
                    "message": { "bsonType": "string" },  
                    "dateCreated": { "bsonType": "date"},  
                    "like": { "bsonType": "int" }  
                }  
            }  
        }  
    }  
}
```

จากความรู้ที่ผ่านมา เราไม่สามารถใช้กู๊ดที่  
เรียนมาจาก RDB เดิมได้

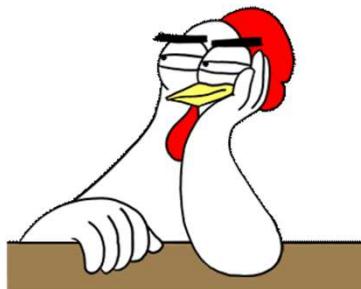
เราจะทำอย่างไรดี

การออกแบบ MongoDB  
schema มีกู๊ดเกณฑ์อย่างไร

ถึงจุดนี้อาจจะมีคำว่า  
แล้วเมื่อไหร่จะรวมตารางกล้ายเป็น  
single collection ดี

## การออกแบบスキีมาของ MongoDB

- ไม่มีขันตอนที่เป็นมาตรฐาน
- ไม่มีอัลกอริทึมจำเพาะ
- ไม่มีกฎเกณฑ์ตายตัว



<https://www.mongodb.com/developer/products/mongodb/mongodb-schema-design-best-practices/>

# MongoDB Best Practices

- <https://www.mongodb.com/developer/products/mongodb/mongodb-schema-design-best-practices/>

Best Practices vs. Standards?



## แล้วเราจะใช้ **Referencing** หรือ **Embedding** ดี

- การออกแบบスキมาของ MongoDB ต้องตัดสินใจว่าจะเก็บข้อมูลในลักษณะใด ต่อไปนี้
  - Embedding คือ เก็บใน document เดียวกันเป็น Key-value ใหม่
  - Referencing คือ เก็บเป็น collection ใหม่ และลิงค์กันผ่าน operator ชื่อ \$lookup

<https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/>

# Embedding vs. Referencing

Database: Web Virgil

Collection: Students

```
{  
  "_id": "61cd69289f5b6514920574f",  
  "name": "Jesse Pinkman",  
  "address":  
    {  
      "street": "0908 Margo Street",  
      "city": "Albuquerque",  
      "state": "New Mexico",  
      "zip": "12345"  
    },  
  "email": "pinkmanindahouse@gmail.com"  
}
```

Embedded Document

Database: Web Virgil

Collection: Students

Document

```
{  
  "id": "61cd69289f5b6514920574f",  
  "name": "Jesse Pinkman",  
  "email": "pinkmanindahouse@gmail.com"  
}
```

Collection: Address

Document

```
{  
  "studentId": "61cd69289f5b6514920574f",  
  "street": "0908 Margo Street",  
  "city": "Albuquerque",  
  "state": "New Mexico",  
  "zip": "12345"  
}
```

## ข้อดี ข้อเสีย ของ Embedding

### Pro

- ดึงข้อมูลทั้งหมดได้ใน query เดียว
- เลี่ยงการต้องลิงค์ข้อมูลระหว่าง collection ผ่าน \$lookup เพื่อลด overhead
- การแก้ไขข้อมูลเป็น atomic มีการประกัน ACID property
- ถ้าใช้ multiple operations ต้องใช้ transaction และก็มีรองรับเรื่อง transaction ตั้งแต่ MongoDB 4

### Con

- Document จะมีขนาดใหญ่เกินทำให้มี overhead ในการดึงข้อมูล โดยข้อมูลที่ไม่จำเป็นในการ query ก็ต้องโหลดลงหน่วยความจำด้วย
- MongoDB จะจำกัดขนาด document ในรูปแบบ BSON ที่ 16 MB

<https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/>

## ข้อดี ข้อเสีย ของ Referencing

### Pro

- ขนาดของ document ไม่ใหญ่เกินและไม่เกินข้อจำกัดของ MongoDB
- ลดความซ้ำซ้อนของข้อมูล แต่ไม่ใช่ประจำนิยมมากนัก เพราะถ้าซ้ำแล้ว performance ดีตาม use cases ก็ไม่เป็นไร

### Con

- ต้องใช้อย่างน้อยสอง queries หรือใช้ operator \$lookup

<https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/>

# กฎข้อที่ 1 ของการออกแบบสกีมา

1

1. เลือกใช้ embedding ก่อนถ้าทำได้ รวมทั้งกรณีของ one-to-few ด้วยให้ใช้ embedding

Embedding First



## ประเภทความสัมพันธ์ของข้อมูล one-to-one



employee document

```
{  
    "_id": "ObjectId('AAA')",  
    "name": "Joe Karlsson",  
    "company": "MongoDB",  
    "twitter": "@JoeKarlsson1",  
    "twitch": "joe_karlsson",  
    "tiktok": "joekarlsson",  
    "website": "joekarlsson.com",  
}
```

รวมทุกอย่างใน  
Document เดียวเท่านั้น  
ไปไหน ไปด้วยกัน เพราะ  
ไม่มีภาระมาก

## กฎข้อที่ 2 One-to-Few รวมอยู่ด้วยกันแบบ Embedding ได้ ยกเว้นต้องทำ Direct access จึงพิจารณาแยกแบบ Referencing

2

2.1 ปกติกรณี One-to-Few คือมีจำนวนไม่มากก็ทำแบบ Embedding ได้เลย

2.2 แต่ถ้ามี use cases ที่ต้องการเข้าถึงข้อมูลโดยตรง ก็ให้ทำข้อมูลนั้นๆ เป็น document ของ collection ใหม่ เช่น ข้อมูล parts ในส่วนหน้าที่แล้ว

(Referencing) ให้แบบกฎข้อ 3 ได้เลย

# ประเภทความสัมพันธ์ ของข้อมูล One-to-Few



Product document

```
{ "name": "left-handed smoke shifter",
  "manufacturer": "Acme Corp",
  "catalog_number": "1234",
  "parts": [
```

```
    { "_id": "ObjectId('AAAA')",
      "partno": "123-aff-456",
      "name": "#4 grommet",
      "qty": "94",
      "price": "3.99"
    },
    { "_id": "ObjectId('AAAA')",
      "partno": "123-aff-456",
      "name": "#4 grommet",
      "qty": "94",
      "price": "3.99"
    }
  ....
```

```
}
```

Few Embedding  
Documents  
(ตาราง Parts ซ่อน  
ในตาราง Product)

<https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/>

# 3

## กฎข้อที่ 3 One-to-Many แยกจากกันแบบ Referencing

3. สร้าง Collection ใหม่สำหรับส่วน Many (Referencing)

<https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/>

## ประเภทความสัมพันธ์ของข้อมูล One-to-Many

Product document

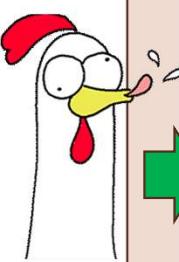


{ "name": "left-handed smoke shifter",  
"manufacturer": "Acme Corp",  
"catalog\_number": "1234",  
"parts": ["ObjectId('AAAA')", "ObjectId('BBBB')", "ObjectId('CCCC')"] }

Access ตรง

แยก Document ออกจากกัน  
แบบ Referencing

Part document



{ "\_id" : "ObjectId('AAAA')",  
"partno" : "123-aff-456",  
"name" : "#4 grommet",  
"qty": "94",  
"price": "3.99" }

Access ตรง

{ "\_id" : "ObjectId('BBBB')",  
.....  
}

{ "\_id" : "ObjectId('CCCC')",  
.....  
}

## กฎข้อที่ 4 One-to-Squillions (อ้างถึงข้อมูลจำนวนมากๆ)

4

4. กรณีกฎข้อ 1 ข้อมูลที่ Embed อยู่ในรูปแบบ array ต้องมีการกำหนดขอบเขต  
ไว้ด้วย เพราะจะบวมจนระเบิดได้ เช่น

- เดิมมี items เป็นหลักร้อย ก็ใช้แบบ 1-many reference ids ไปก่อน
- แต่ถ้ามี items หลักพัน ก็ชี้กลับแบบ one-to-squillions

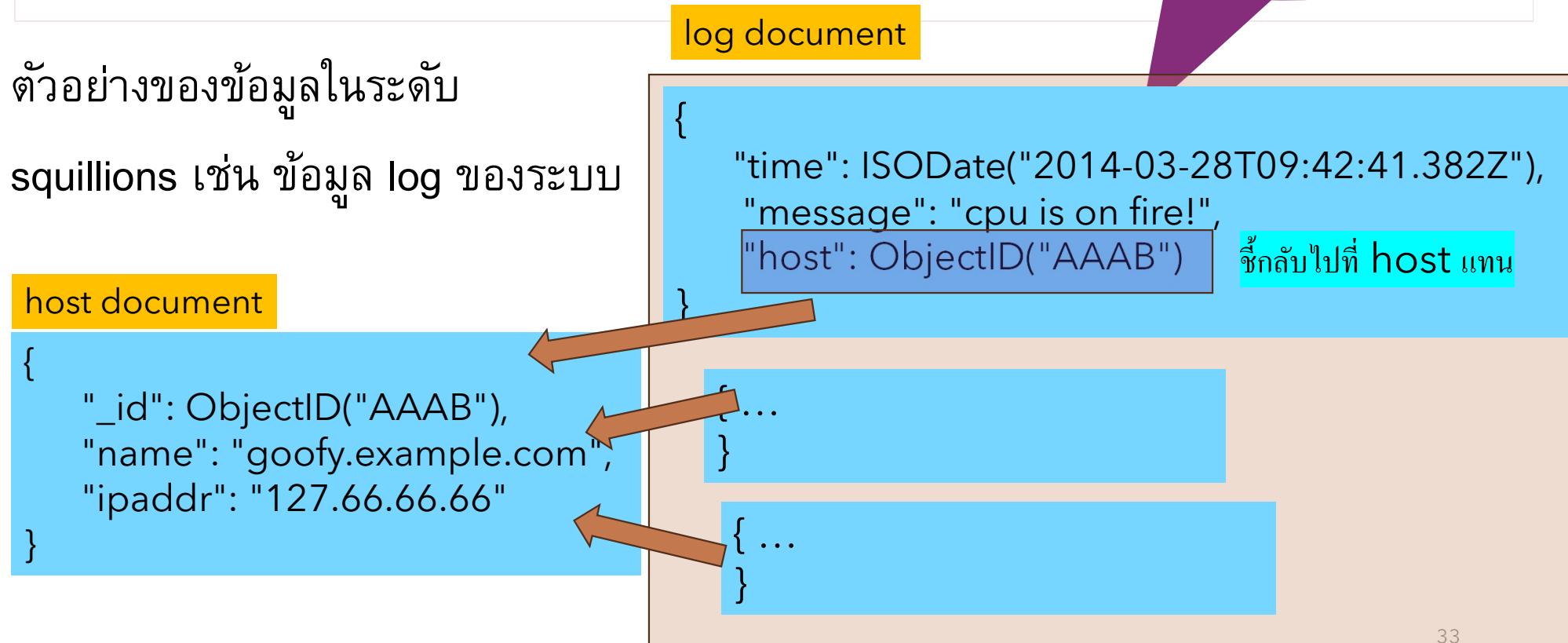
สรุปค่า cardinality สูงๆ เป็นเหตุผลหลักในการแยกออกไปเป็น document ใน collection ใหม่

<https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/>

## ประเภทความสัมพันธ์ของข้อมูล one-to-squillions

ตัวอย่างของข้อมูลในระดับ  
squillions เช่น ข้อมูล log ของระบบ

ในการนี้เราระบายนอกจาก host ออกมายัง host



## ประเภทความสัมพันธ์ของข้อมูล many-to-many

- 

<https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/>

## กฎข้อที่ 5 Many-to-Many

5

- ออกแบบให้มีการ Referencing ไปกลับระหว่าง 2 Collections
- ทั้งนี้เราจะออกแบบสกีมาย่างไร ขึ้นอยู่กับ use cases และรูปแบบการเข้าถึงข้อมูลเป็นหลัก ว่ามีการ queries / updates ข้อมูลในลักษณะใดบ้าง โดยออกแบบสกีมามเพื่อให้สอดคล้องกับการใช้งาน
- ตัวอย่างแอปพลิเคชัน TODO ใน Sprint tasks ที่สมาชิกในทีม dev สามารถอาสา (volunteer) ในการทำได้หลาย tasks และ แต่ละ tasks ก็มีสมาชิกช่วยทำได้หลายคน

<https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/>

## ประเภทความสัมพันธ์ของข้อมูล many-to-many

ในการนี้เรามารถทำ reference  
ทั้งไปและกลับ

developer document

```
{   "_id": ObjectId("AAF1"),
    "name": "Kate Monster",
    "tasks": [ObjectId("ADF9"), ObjectId("AE02"), ObjectId("AE73")]
}
```

task document

```
{   "_id": ObjectId("ADF9"), "description":
    "Write blog post about MongoDB schema design",
    "due_date": ISODate("2014-04-01"),
    "volunteers": [ObjectId("AAF1"), ObjectId("BB3G")]
}
```

<https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/>

## สรุป กฎของการออกแบบスキม่า (ถือเป็นแนวทางเท่านั้น)

1. One-to-One: embed ข้อมูลใน document เลย
2. One-to-Few: embed ข้อมูลใน document เลย แต่ก็อย่างว่า ถ้า few นั้นมีการเข้าถึงโดยตรงด้วย บ่อຍๆ ก็แยกไปได้
3. One-to-Many: ถ้า embedding ให้ก็ทำ
4. One-to-Squilllion: ทำ referencing ชี้กลับ
5. Many-to-Many: ทำ referencing ไป-กลับ

<https://www.mongodb.com/developer/article/mongodb-schema-design-best-practices/>



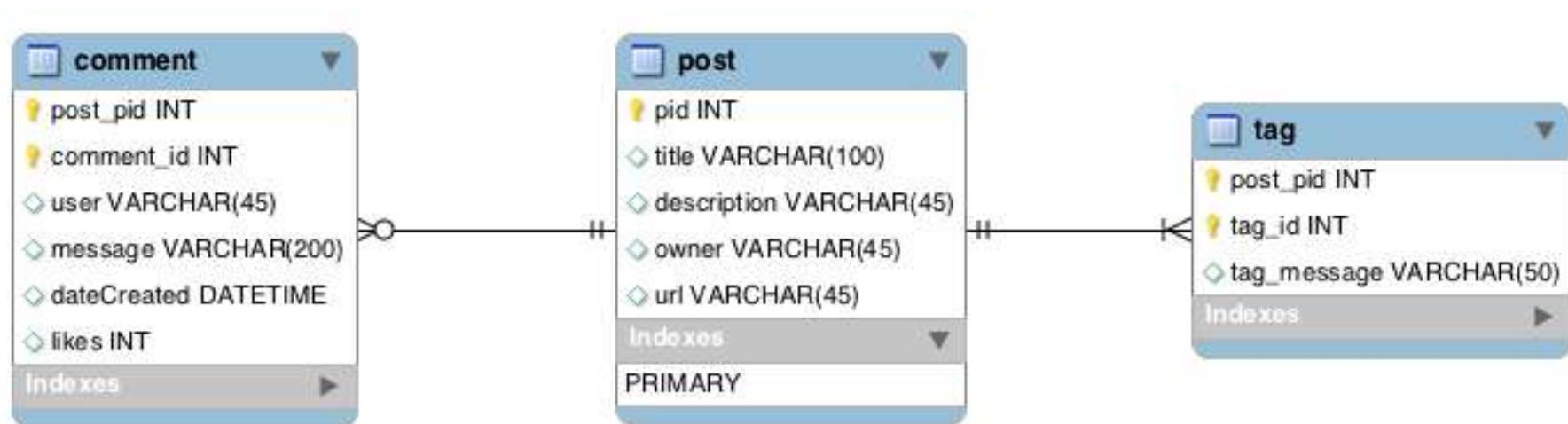
# การใช้ MongoDB

---

## หัวข้อต่อจากนี้ไป

- ใช้ Case study เป็นระบบ Hive
- ติดตั้ง MongoDB Community Server แบบ local
- ติดตั้งและใช้งานผ่าน Mongosh
- เรียนรู้ method ของ MongoDB เพื่อทำ CRUD (คล้าย DML ของ SQL)
- ติดตั้งและใช้งานผ่าน Compass
- ใช้งาน MongoDB ผ่าน Atlas (online MongoDB)

# Case Study ที่ใช้คือ HiVE's ER Diagram



## HiVE's MongoDB schema design (Only one collection “post”)

### Post collection

```
{  
    title: 'MongoDB Overview',  
    description: 'An intro to MongoDB/document',  
    owner: 'lionking',  
    tags: ['mongodb', 'database', 'NoSQL'],  
    likes: 340  
    comments: [  
        {  
            user: 'hikaru',  
            message: 'nice intro',  
            dateCreated: '2021-10-10',  
            likes: 5  
        },  
        {  
            user: 'miko',  
            message: 'thanks',  
            dateCreated: '2020-11-2',  
            likes: 4  
        } , ...  
    ]  
}
```

ออกแบบให้เป็น  
Embedding

แผนการเด็กตารางในตัวอย่างนี้เรารอกรูปแบบสกีม่าโดยการ embed ข้อมูลที่เกี่ยวข้องเข้ามาอยู่ร่วมกัน

```
{  
    "bsonType": "object",  
    "required": ["title", "owner"],  
    "properties": {  
        "title": {  
            "bsonType": "string"  
        },  
        "owner": {  
            "bsonType": "string",  
        },  
        ...  
        "comments": {  
            "bsonType": "array",  
            "uniqueItems": true,  
            "items": {  
                "bsonType": "object",  
                "properties": {  
                    "user": { "bsonType": "objectId"},  
                    "message": { "bsonType": "string" },  
                    "dateCreated": { "bsonType": "date"},  
                    "like": { "bsonType": "int"  
                }  
            }  
        }  
    }  
}
```

## Start to use mongodb

ติดตั้ง MongoDB ไว้แล้ว (โครงสร้างไม่มียกเว้นขึ้น)

1. Start the MongoDB server using

System shell > mongod

2. Login to the MongoDB server via the MongoDB shell

System shell > mongosh

## MongoDB - create database

พิมพ์หน้าจอตามด้วย

- `show dbs` ใช้แสดงฐานข้อมูลที่อยู่ในระบบ
- `use DATABASE_NAME` ใช้ในการ create or use ฐานข้อมูลหนึ่งๆ

```
[test> use hive  
switched to db hive  
[hive>
```

- `db` ใช้เช็คว่าตอนนี้อยู่ db ไหน

```
hive> db  
hive
```

## MongoDB - drop database

พิมพ์หน้าจอตามด้วย

- `db.dropDatabase()` ใช้ในการลบฐานข้อมูลที่ use อยู่ถ้ายังไม่ได้ use เลยฐานชื่อ test จะถูกลบ

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

Case sensitive นะครับ  
ต้อง `db.dropDatabase()`

# MongoDB - create collection

- db.createCollection(name, option)

- name : ชื่อของคอลเลกชันที่จะสร้าง
- option : ออปชันเกี่ยวกับหน่วยความจำที่จะใช้ และการทำ index

Field	Type	Description
capped	Boolean	(Optional) If true, enables a capped collection. Capped collection is a collection fixed size collection that automatically overwrites its oldest entries when it reaches its maximum size. <b>If you specify true, you need to specify size parameter also.</b>
autoIndexID	Boolean	(Optional) If true, automatically create index on <code>_id</code> field. Default value is false.
size	number	(Optional) Specifies a maximum size in bytes for a capped collection. If <b>capped is true</b> , then you need to specify this field also.
max	number	(Optional) Specifies the maximum number of documents allowed in the capped collection.

Deprecated since version 3.2

# MongoDB - create collection

## 3 วิธีในการสร้าง collection ใหม่

พิมพ์หน้าจอตามด้วย

```
> use hive
switched to db hive
1 > db.createCollection("post")
{ "ok" : 1 }
> show collections
post
2 > db.createCollection("post_w_options", {capped : true, size : 6142800, max : 10000 })
{ "ok" : 1 }
> show collections
post
post_w_options
3 > db.post1.insert({ "owner_name" : "lionking" })
writeResult({ "nInserted" : 1 })
> show collections
post
post1
post1ok - switched to db hive
post_w_options
```

collection  
size in bytes

Max number  
of documents

Collection post1 ถูกสร้างโดย  
อัตโนมัติ โดยการ insert  
ข้อมูลเข้าไปเลย

## MongoDB drop collection

พิมพ์หน้าจอตามด้วย

- db.COLLECTION\_NAME.drop()

The screenshot shows a terminal window with the MongoDB shell. The user has switched to the 'hive' database and listed its collections. The 'post1' collection is highlighted with a red box. The user then executes the command 'db.post1.drop()' which returns 'true'. Finally, the user runs 'show collections' again, and the 'post1' collection is no longer listed.

```
> use hive
switched to db hive
> show collections
post
post1
post_w_options
> db.post1.drop()
true
> show collections
post
post_w_options
```

- **String** : This is most commonly used datatype to store the data. String in mongodb must be UTF-8 valid.
- **Integer** : This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
- **Boolean** : This type is used to store a boolean (true/ false) value.
- **Double** : This type is used to store floating point values.
- **Min/ Max keys** : This type is used to compare a value against the lowest and highest BSON elements.
- **Arrays** : This type is used to store arrays or list or multiple values into one key.
- **Timestamp** : ctimestamp. This can be handy for recording when a document has been modified or added.
- **Object** : This datatype is used for embedded documents.
- **Null** : This type is used to store a Null value.
- **Symbol** : This datatype is used identically to a string however, it's generally reserved for languages that use a specific symbol type.
- **Date** : This datatype is used to store the current date or time in UNIX time format. You can specify your own date time by creating object of Date and passing day, month, year into it.
- **Object ID** : This datatype is used to store the document's ID.
- **Binary data** : This datatype is used to store binay data.
- **Code** : This datatype is used to store javascript code into document.
- **Regular expression** : This datatype is used to store regular expression

## MongoDB datatypes

# ทดลองใช้ MongoDB Compass

- เปิด app ชื่อ MongoDBCompass
- แล้วทดลองสำรวจ Database ชื่อ hive
- แสดง collections ที่มีและ Document

# ເປີດ app MongoDBCompass

MongoDB Compass - localhost:27017/hive

Connections Edit View Help

Compass

My Queries

CONNECTIONS (2)

localhost:27017 + ...

Search connections

localhost:27017 admin config

hive +

localhost:27017 > hive

Open MongoDB shell Create collection Refresh

Sort by Collection Name View

post

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
4.10 kB	0	0 B	1	4.10 kB

post\_w\_option

Storage size:	Documents:	Avg. document size:	Indexes:	Total index size:
4.10 kB	0	0 B	1	4.10 kB

## MongoDB - Insert document

- db.COLLECTION\_NAME.insert(document)

จะเห็นว่าเราไม่ได้  
กำหนด schema มา  
ก่อนเลย

```
hive> db.post.insertOne({  
...     title: 'MongoDB Overview',  
...     'description': 'An introduction to MongoDB / document database',  
...     'owner': 'lionking',  
...     'url': 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',  
...     'tags': ['mongodb', 'database', 'NoSQL'],  
...     'likes': 250})  
{  
    acknowledged: true,  
    insertedId: ObjectId("652d05c44ce35c5655398092")  
}.
```

Response มาจากตัว  
MongoDB

## MongoDB - Insert

```
hive> db.post.insertMany([
...     {
...         title: 'MongoDB Overview',
...         description: 'An introduction to MongoDB / document database',
...         owner: 'lionking',
...         url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',
...         tags: ['mongodb', 'database', 'NoSQL'],
...         likes: 250
...     },
...     {
...         title: 'MongoDB 101',
...         description: 'An introduction to MongoDB / document database via compass',
...         owner: 'lionking',
...         url: 'http://mdslab.unime.it/sites/default/files/mongodb_compass.pdf',
...         tags: ['mongodb', 'database', 'NoSQL', 'compass'],
...         likes: 500,
...         comments:[
...             {
...                 user: 'fiola',
...                 message: 'thanks! lionking',
...                 dateCreated: new Date(2020,7,26,1,30),
...                 likes: 0
...             }
...         ]
...     }
... ]))
```

document  
(w/ multiple documents)

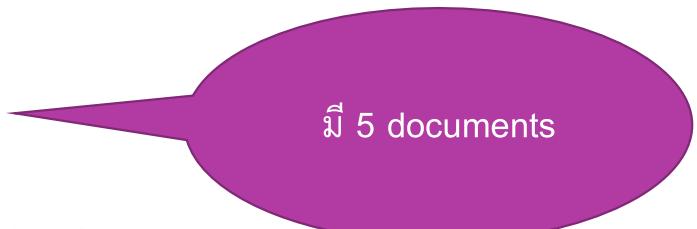
## MongoDB – Insert more document

```
hive> db.post.insert({  
...     title: 'MySQL Overview',  
...     description: 'An introduction to MySQL / a RDBMS database',  
...     owner: 'Hikaru',  
...     url: 'http://mysql.com/mongodb_tutorial.pdf',  
...     tags: ['mysql','database','SQL'],  
...     likes: 180  
... })  
{  
    acknowledged: true,  
    insertedIds: { '0': ObjectId("652d09064ce35c5655398095") }  
}  
hive> db.post.insert({  
...     title: 'Compass tutorial',  
...     description: 'An introduction to MongoDB and Compass',  
...     owner: 'Hikaru',  
...     url: 'http://compass.com/tutorial',  
...     tags: ['NoSQL','mongodb','database','compass'],  
...     likes: 70  
... })  
{  
    acknowledged: true,  
    insertedIds: { '0': ObjectId("652d09064ce35c5655398096") }  
}
```

# MongoDB - query document

- db.COLLECTION\_NAME.find()    SELECT \* FROM post;

```
hive> db.post.find()
[  
  {  
    _id: ObjectId("652d05c44ce35c5655398092"),  
    title: 'MongoDB Overview',  
    description: 'An introduction to MongoDB / document database',  
    owner: 'lionking',  
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',  
    tags: [ 'mongodb', 'database', 'NoSQL' ],  
    likes: 250  
  },  
  {  
    _id: ObjectId("652d07564ce35c5655398093"),  
    title: 'MongoDB Overview',  
    description: 'An introduction to MongoDB / document database',  
    owner: 'lionking',  
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',  
    tags: [ 'mongodb', 'database', 'NoSQL' ],  
    likes: 250  
  },  
  {  
    _id: ObjectId("652d07564ce35c5655398094"),  
    title: 'MongoDB Overview',  
    description: 'An introduction to MongoDB / document database',  
    owner: 'lionking',  
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',  
    tags: [ 'mongodb', 'database', 'NoSQL' ],  
    likes: 250  
  },  
  {  
    _id: ObjectId("652d07564ce35c5655398095"),  
    title: 'MongoDB Overview',  
    description: 'An introduction to MongoDB / document database',  
    owner: 'lionking',  
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',  
    tags: [ 'mongodb', 'database', 'NoSQL' ],  
    likes: 250  
  },  
  {  
    _id: ObjectId("652d07564ce35c5655398096"),  
    title: 'MongoDB Overview',  
    description: 'An introduction to MongoDB / document database',  
    owner: 'lionking',  
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',  
    tags: [ 'mongodb', 'database', 'NoSQL' ],  
    likes: 250  
  }]
```



⌘ 5 documents

## MongoDB query document : RDBMS Vs MongoDB

Operation	Syntax	Example	RDBMS equivalent
Equality	{key:value}	db.post.find({owner: "lionking"})	where owner = "lionking"
Less than	{key:{\$lt:value}}	db.post.find({likes: {\$lt:100}})	where likes < 100
Less than or equal	{key:{\$lte:value}}	db.post.find({likes: {\$lte:100}})	where likes <= 100
Greater than	{key:{\$gt:value}}	db.post.find({likes: {\$gt:100}})	where likes > 100
Greater than or eqaul	{key:{\$gte:value}}	db.post.find({likes: {\$gte:100}})	where likes >= 100
Not equal	{key:{\$ne:value}}	db.post.find({likes: {\$ne:100}})	where likes != 100

```
hive> db.post.find({likes:{$gte:50}})  
[  
 {  
 _id: ObjectId("652d05c44ce35c5655398092"),  
 title: 'MongoDB Overview',  
 description: 'An introduction to MongoDB / document database',  
 owner: 'lionking',  
 url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',  
 tags: [ 'mongodb', 'database', 'NoSQL' ],  
 likes: 250  
 },  
 {  
 _id: ObjectId("652d07564ce35c5655398093"),  
 title: 'MongoDB Overview',  
 description: 'An introduction to MongoDB / document database',  
 owner: 'lionking',  
 url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',  
 tags: [ 'mongodb', 'database', 'NoSQL' ],  
 likes: 250  
 },  
 {  
 _id: ObjectId("652d07564ce35c5655398094"),  
 title: 'MongoDB 101',  
 description: 'An introduction to MongoDB / document database via compass',  
 owner: 'lionking',  
 url: 'http://mdslab.unime.it/sites/default/files/mongodb_compass.pdf',  
 tags: [ 'mongodb', 'database', 'NoSQL', 'compass' ],  
 likes: 500,  
 comments: [
```

## MongoDB query document find() w/ conditions

```
db.post.find({ likes: { $  
 gte:50 } })
```

```
SELECT * FROM post WHERE likes >= 50;
```

## Find() w/ AND in MongoDB

```
hive> db.post.find({likes:{$gte:50},owner:"Hikaru"})
[
  {
    _id: ObjectId("652d09064ce35c5655398095"),
    title: 'MySQL Overview',
    description: 'An introduction to MySQL / a RDBMS database',
    owner: 'Hikaru',
    url: 'http://mysql.com/mongodb_tutorial.pdf',
    tags: [ 'mysql', 'database', 'SQL' ],
    likes: 180
  },
  {
    _id: ObjectId("652d09064ce35c5655398096"),
    title: 'Compass tutorial',
    description: 'An introduction to MongoDB and Compass',
    owner: 'Hikaru',
    url: 'http://compass.com/tutorial',
    tags: [ 'NoSQL', 'mongodb', 'database', 'compass' ],
    likes: 70
  }
]
```

```
{
  attr:{$op:value},
  attr:{$op:value},
  ...
}
```

AND

```
[hive] > db.post.find({$or:[{likes:{$gte:50}}, {owner:"hikaru"}]})  
[  
  {  
    _id: ObjectId("652d05c44ce35c5655398092"),  
    title: 'MongoDB Overview',  
    description: 'An introduction to MongoDB / document database',  
    owner: 'lionking',  
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',  
    tags: [ 'mongodb', 'database', 'NoSQL' ],  
    likes: 250  
  },  
  {  
    _id: ObjectId("652d07564ce35c5655398093"),  
    title: 'MongoDB Overview',  
    description: 'An introduction to MongoDB / document database',  
    owner: 'lionking',  
    url: 'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',  
    tags: [ 'mongodb', 'database', 'NoSQL' ],  
    likes: 250  
  },  
  ...  
]
```

## Find() w/ OR in MongoDB

```
{  
  $or:  
  [  
    {attr:{$op:value}},  
    {attr:{$op:value}},  
    ...  
  ]  
}
```

OR

```
SELECT * FROM post WHERE  
likes >= 50 OR owner = "hikaru";
```

```
hive> db.post.find({title:{$eq:"MongoDB 101"},$or:[{likes:{$gte:200}},{owner:"hikaru"}]})  
[  
 {  
 _id: ObjectId("652d07564ce35c5655398094"),  
 title: 'MongoDB 101',  
 description: 'An introduction to MongoDB / document database via compass',  
 owner: 'lionking',  
 url: 'http://mdslab.unime.it/sites/default/files/mongodb_compass.pdf',  
 tags: [ 'mongodb', 'database', 'NoSQL', 'compass' ],  
 likes: 500,  
 comments: [  
 {  
 user: 'fiola',  
 message: 'thanks! lionking',  
 dateCreated: ISODate("2020-08-25T18:30:00.000Z"),  
 likes: 0  
 }  
 ]  
 }]
```

## Find() w/ AND plus OR in MongoDB

```
{  
 $and: {$op:value},  
 $and: {$op:value},  
 $or:  
 [  
 {attr:{$op:value}},  
 {attr:{$op:value}},  
 ...  
 ]  
 }
```

AND plus  
OR

## MongoDB - update document

- db.COLLECTION\_NAME.updateOne(SELECTION\_CRITERIA, UPDATED\_DATA)

```
[hive]> db.post.updateOne({title:"MongoDB 101"},{$set:{title:"MongoDB HelloWorld"})  
{  
    acknowledged: true,  
    insertedId: null,  
    matchedCount: 1,  
    modifiedCount: 1,  
    upsertedCount: 0  
}
```

UPDATE post SET title = "MongoDB Hello World" WHERE  
title = "MongoDB 101";

## MongoDB - delete document

```
db.COLLECTION_NAME.remove(DELETION_CRITERIA) เท่ากับ deleteMany()
```

หรือ

```
db.COLLECTION_NAME.deleteOne(DELETION_CRITERIA)
```

```
db.COLLECTION_NAME.deleteMany(DELETION_CRITERIA)
```

```
hive> db.post.find()
[{"_id": ObjectId("652d05c44ce35c5655398092"),
 "title": "MongoDB Overview",
 "description": "An introduction to MongoDB / document database",
 "owner": "lionking",
 "url": "http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf",
 "tags": [ "mongodb", "database", "NoSQL" ],
 "likes": 250},
 {"_id": ObjectId("652d07564ce35c5655398093"),
 "title": "MongoDB Overview",
 "description": "An introduction to MongoDB / document database",
 "owner": "lionking",
 "url": "http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf",
 "tags": [ "mongodb", "database", "NoSQL" ],
 "likes": 250},
 {"_id": ObjectId("652d07564ce35c5655398094"),
 "title": "MongoDB HelloWorld",
 "description": "An introduction to MongoDB / document database via compass",
 "owner": "lionking",
 "url": "http://mdslab.unime.it/sites/default/files/mongodb_compass.pdf",
 "tags": [ "mongodb", "database", "NoSQL", "compass" ],
 "likes": 500,
 "comments": [{"_id": ObjectId("652d07564ce35c5655398095"),
 "text": "Great tutorial! I learned a lot about MongoDB from this."}]}]
```

```
[hive> db.post.remove({title:"MongoDB Overview"})
{ acknowledged: true, deletedCount: 2 }
```

```
[hive> db.post.deleteMany({title:"MongoDB Overview"})
{ acknowledged: true, deletedCount: 0 }
```

DELETE FROM post WHERE title = "MongoDB Overview";

```
hive> db.post.deleteMany({})
{ acknowledged: true, deletedCount: 3 }
```

```
|hive> db.post.find()
```

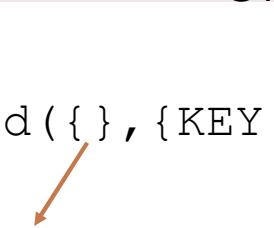
DELETE \* FROM post;



เหลือแต่ collection ว่างๆ ใน  
MongoDB หรือเหลือแต่  
ตารางเปล่าๆ ใน MySQL

# MongoDB projection

**Select only some columns**

- db.COLLECTION\_NAME.find( { } , {KEY:1} )
- condition
- ON/OFF
- 

อย่าลืม insert ข้อมูลกลับเข้า  
ไปก่อน มี 5 documents จาก  
ตัวอย่างตอนต้น

# MongoDB projection

**Select only some columns**

```
hive> db.post.find({}, {title:1})  
[  
  {  
    _id: ObjectId("652d13974ce35c5655398097"),  
    title: 'MongoDB Overview'  
  },  
  {  
    _id: ObjectId("652d139e4ce35c5655398098"),  
    title: 'MongoDB Overview'  
  },  
  { _id: ObjectId("652d139e4ce35c5655398099"), title: 'MongoDB 101' },  
  {  
    _id: ObjectId("652d13a84ce35c565539809a"),  
    title: 'MySQL Overview'  
  },  
  {  
    _id: ObjectId("652d13a84ce35c565539809b"),  
    title: 'Compass tutorial'  
  }  
]
```

```
SELECT title FROM post;
```

## MongoDB projection

*Select only some columns*

```
hive> db.post.find({}, {title:1, _id:0})  
[  
  { title: 'MongoDB Overview' },  
  { title: 'MongoDB Overview' },  
  { title: 'MongoDB 101' },  
  { title: 'MySQL Overview' },  
  { title: 'Compass tutorial' }  
]
```

SELECT title FROM post;

## MongoDB projection

**Select only some columns (array ออกมาได้เลย)**

```
hive> db.post.find({}, {title:1, _id:0, tags:1})  
[  
  {  
    title: 'MongoDB Overview',  
    tags: [ 'mongodb', 'database', 'NoSQL' ]  
  },  
  {  
    title: 'MongoDB Overview',  
    tags: [ 'mongodb', 'database', 'NoSQL' ]  
  },  
  {  
    title: 'MongoDB 101',  
    tags: [ 'mongodb', 'database', 'NoSQL', 'compass' ]  
  },  
  { title: 'MySQL Overview', tags: [ 'mysql', 'database', 'SQL' ] },  
  {  
    title: 'Compass tutorial',  
    tags: [ 'NoSQL', 'mongodb', 'database', 'compass' ]  
  }  
]
```

## MongoDB projection

**Select only some columns \$all (all items in array)**

```
[hive> db.post.find({tags: {$all:["mongodb", "database"]}}, {title:1, _id:0, tags:1})  
[  
 {  
   title: 'MongoDB Overview',  
   tags: [ 'mongodb', 'database', 'NoSQL' ]  
 },  
 {  
   title: 'MongoDB Overview',  
   tags: [ 'mongodb', 'database', 'NoSQL' ]  
 },  
 {  
   title: 'MongoDB 101',  
   tags: [ 'mongodb', 'database', 'NoSQL', 'compass' ]  
 },  
 {  
   title: 'Compass tutorial',  
   tags: [ 'NoSQL', 'mongodb', 'database', 'compass' ]  
 }  
]
```

# MongoDB projection

**Select only some columns \$in (some items in array)**

```
hive> db.post.find({tags: {$in:["compass", "database"]}}, {title:1, _id:0, tags:1})  
[  
  {  
    title: 'MongoDB Overview',  
    tags: [ 'mongodb', 'database', 'NoSQL' ]  
  },  
  {  
    title: 'MongoDB Overview',  
    tags: [ 'mongodb', 'database', 'NoSQL' ]  
  },  
  {  
    title: 'MongoDB 101',  
    tags: [ 'mongodb', 'database', 'NoSQL', 'compass' ]  
  },  
  { title: 'MySQL Overview', tags: [ 'mysql', 'database', 'SQL' ] },  
  {  
    title: 'Compass tutorial',  
    tags: [ 'NoSQL', 'mongodb', 'database', 'compass' ]  
  }  
]
```

## MongoDB - Limit / Skip records

- db.COLLECTION\_NAME.find().limit(NUMBER)  
hive> db.post.find({}, {title:1, \_id:0}).limit(3);  
[  
  { title: 'MongoDB Overview' },  
  { title: 'MongoDB Overview' },  
  { title: 'MongoDB 101' }  
]

```
SELECT title FROM post LIMIT 3;
```

- db.COLLECTION\_NAME.find().limit(NUMBER).skip(NUMBER)  
hive> db.post.find({}, {title:1, \_id:0}).limit(3).skip(2)  
[  
  { title: 'MongoDB 101' },  
  { title: 'MySQL Overview' },  
  { title: 'Compass tutorial' }  
]

```
SELECT title FROM post OFFSET 2 FETCH FIRST 3 ROW ONLY;
```

## MongoDB - sort document

ON/OFF, ASC/DESC

- db.COLLECTION\_NAME.find().sort({KEY:1})

```
[hive> db.post.find({}, {title:1, _id:0})
[
  { title: 'MongoDB Overview' },
  { title: 'MongoDB Overview' },
  { title: 'MongoDB 101' },
  { title: 'MySQL Overview' },
  { title: 'Compass tutorial' }
]
[hive> db.post.find({}, {title:1, _id:0}).sort({title:1})]
[
  { title: 'Compass tutorial' },
  { title: 'MongoDB 101' },
  { title: 'MongoDB Overview' },
  { title: 'MongoDB Overview' },
  { title: 'MySQL Overview' }
]
```

ACS เรียงหน่อยไปมาก by default

SELECT title FROM post ORDER BY title;

## MongoDB - sort document

ON/OFF, ASC/DESC

- db.COLLECTION\_NAME.find().sort({KEY:1})

```
|hive> db.post.find({}, {title:1, _id:0})  
[
```

```
  { title: 'MongoDB Overview' },  
  { title: 'MongoDB Overview' },  
  { title: 'MongoDB 101' },  
  { title: 'MySQL Overview' },  
  { title: 'Compass tutorial' }
```

```
]
```

```
|hive> db.post.find({}, {title:1, id:0}).sort({title:-1})  
[
```

```
  { title: 'MySQL Overview' },  
  { title: 'MongoDB Overview' },  
  { title: 'MongoDB Overview' },  
  { title: 'MongoDB 101' },  
  { title: 'Compass tutorial' }
```

```
]
```

SELECT title FROM post ORDER BY title DESC;

-1 เรียงจากมากไปน้อย

## MongoDB - indexing

Field ที่จะใช้ทำ index

- db.COLLECTION\_NAME.ensureIndex( {KEY:1} )

```
[hive> db.post.ensureIndex({title:1})  
[ 'title_1' ]
```

## MongoDB - aggregation

- `db.COLLECTION_NAME.aggregate(AGGREGATION PIPELINE)`
- An aggregation pipeline consists of one or more stages that process documents
- For example,

```
db.<>CollectionName<>.aggregate([ <>stage1<>, <>stage2<> ] )
```

Collection

```
db.orders.aggregate( [  
    $match stage → { $match: { status: "A" } },  
    $group stage → { $group: { _id: "$cust_id", total: { $sum: "$amount" } } }  
])
```

orders

```
{  
    cust_id: "A123",  
    amount: 500,  
    status: "A"  
}  
  
{  
    cust_id: "A123",  
    amount: 250,  
    status: "A"  
}  
  
{  
    cust_id: "B212",  
    amount: 200,  
    status: "A"  
}  
  
{  
    cust_id: "A123",  
    amount: 300,  
    status: "D"  
}
```

\$match

```
{  
    cust_id: "A123",  
    amount: 500,  
    status: "A"  
}  
  
{  
    cust_id: "A123",  
    amount: 250,  
    status: "A"  
}  
  
{  
    cust_id: "B212",  
    amount: 200,  
    status: "A"  
}
```

\$group

```
{  
    _id: "A123",  
    total: 750  
}  
  
{  
    _id: "B212",  
    total: 200  
}
```

# MongoDB – aggregate functions: sum, avg

- db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION)

```
[hive> db.post.aggregate([{$group: {_id:"$owner", num_content: {$sum: 1}}}] )
[
  { _id: 'lionking', num_content: 3 },
  { _id: 'Hikaru', num_content: 2 }
]
hive> db.post.aggregate([{$group: {_id:"$owner", total_likes: {$sum: "$likes"}}}])
[
  { _id: 'lionking', total_likes: 1000 },
  { _id: 'Hikaru', total_likes: 250 }
]
hive> db.post.aggregate([{$group: {_id:"$owner", total_likes: {$avg: "$likes"}}}])
[
  { _id: 'lionking', total_likes: 333.3333333333333 },
  { _id: 'Hikaru', total_likes: 125 }
]
```

# MongoDB – aggregation with sorting

- db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION)

```
hive> db.post.aggregate([{$group: {_id:"$owner", total_likes: {$sum: "$likes"}}, {$sort: {total_likes:1}}})
[
  { _id: 'Hikaru', total_likes: 250 },
  { _id: 'lionking', total_likes: 1000 }
]
hive> db.post.aggregate([{$group: {_id:"$owner", total_likes: {$sum: "$likes"}}, {$sort: {total_likes:-1}}})
)
[
  { _id: 'lionking', total_likes: 1000 },
  { _id: 'Hikaru', total_likes: 250 }
]
```

# MongoDB – aggregate function \$push

- db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION)

```
[hive]> db.post.aggregate([{$group: {_id:"$owner", urls: {$push: "$url"}}}])  
[  
  {  
    _id: 'lionking',  
    urls: [  
      'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',  
      'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',  
      'http://mdslab.unime.it/sites/default/files/mongodb_compass.pdf'  
    ]  
  },  
  {  
    _id: 'Hikaru',  
    urls: [  
      'http://mysql.com/mongodb_tutorial.pdf',  
      'http://compass.com/tutorial'  
    ]  
  }  
]
```

# MongoDB – aggregate function \$addToSet

- db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION)

```
hive> db.post.aggregate([{$group: {_id:"$owner", urls: {$addToSet: "$url"}}}])  
[  
  {  
    _id: 'Hikaru',  
    urls: [  
      'http://compass.com/tutorial',  
      'http://mysql.com/mongodb_tutorial.pdf'  
    ]  
  },  
  {  
    _id: 'lionking',  
    urls: [  
      'http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf',  
      'http://mdslab.unime.it/sites/default/files/mongodb_compass.pdf'  
    ]  
  }  
]
```

## MongoDB – aggregate function \$last

- db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION)

```
hive> db.post.aggregate([{$group: {_id:"$owner", urls: {$last: "$url"}}}])  
[  
  { _id: 'Hikaru', urls: 'http://compass.com/tutorial' },  
  {  
    _id: 'lionking',  
    urls: 'http://mdslab.unime.it/sites/default/files/mongodb_compass.pdf'  
  }  
]
```

## MongoDB – having

- db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION)

```
hive> db.post.aggregate([{$group: {_id:"$owner", num_content: {$sum: 1}}}])
[
  { _id: 'lionking', num_content: 3 },
  { _id: 'Hikaru', num_content: 2 }
]
hive> db.post.aggregate([{$group: {_id:"$owner", num_content: {$sum: 1}}}, {$match:{num_content:{$gte:2}} }])
[
  { _id: 'lionking', num_content: 3 },
  { _id: 'Hikaru', num_content: 2 }
]
```

# MongoDB – aggregation expressions

Expression	Description	Example
\$sum	Sums up the defined value from all documents in the collection.	db.mycol.aggregate([{\$group : {_id : "\$by_user", numTutorial : {\$sum : "\$likes"}}}])
\$avg	Calculates the average of all given values from all documents in the collection.	db.mycol.aggregate([{\$group : {_id : "\$by_user", numTutorial : {\$avg : "\$likes"}}}])
\$min	Gets the minimum of the corresponding values from all documents in the collection.	db.mycol.aggregate([{\$group : {_id : "\$by_user", numTutorial : {\$min : "\$likes"}}}])
\$max	Gets the maximum of the corresponding values from all documents in the collection.	db.mycol.aggregate([{\$group : {_id : "\$by_user", numTutorial : {\$max : "\$likes"}}}])
\$push	Inserts the value to an array in the resulting document.	db.mycol.aggregate([{\$group : {_id : "\$by_user", url : {\$push: "\$url"}}}])
\$addToSet	Inserts the value to an array in the resulting document but does not create duplicates.	db.mycol.aggregate([{\$group : {_id : "\$by_user", url : {\$addToSet : "\$url"}}}])
\$first	Gets the first document from the source documents according to the grouping. Typically this makes only sense together with some previously applied "\$sort"-stage.	db.mycol.aggregate([{\$group : {_id : "\$by_user", firstUrl : {\$first : "\$url"}}}])
\$last	Gets the last document from the source documents according to the grouping. Typically this makes only sense together with some previously applied "\$sort"-stage.	db.mycol.aggregate([{\$group : {_id : "\$by_user", lastUrl : {\$last : "\$url"}}}])

# \$lookup

post2

```
{  
    "title" : "my first post",  
    "author" : "Jim",  
    "likes" : 5  
},  
{  
    "title" : "my second post",  
    "author" : "Jim",  
    "likes" : 2  
},  
{  
    "title" : "hello world",  
    "author" : "Joe",  
    "likes" : 3  
}
```

comment2

```
{  
    "postTitle" : "my first post",  
    "comment" : "great read",  
    "likes" : 3  
},  
{  
    "postTitle" : "my second post",  
    "comment" : "good info",  
    "likes" : 0  
},  
{  
    "postTitle" : "my second post",  
    "comment" : "i liked this post",  
    "likes" : 12  
},  
{  
    "postTitle" : "hello world",  
    "comment" : "not my favorite",  
    "likes" : 8  
},  
{  
    "postTitle" : "my last post",  
    "comment" : null,  
    "likes" : 0  
}
```

<https://www.stackchief.com/tutorials/%24lookup%20Examples%20%7C%20MongoDB>

```
hive> db.post2.insertMany([
... {
...   "title": "my first post",
...   "author": "Jim",
...   "likes": 5
... },
... {
...   "title": "my second post",
...   "author": "Jim",
...   "likes": 2
... },
... {
...   "title": "hello world",
...   "author": "Joe",
...   "likes": 3
... }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("652d31a183c372d7ba29d0be"),
    '1': ObjectId("652d31a183c372d7ba29d0bf"),
    '2': ObjectId("652d31a183c372d7ba29d0c0")
  }
}
```

```
hive> db.comment2.insert([
... {
...   "postTitle": "my first post",
...   "comment": "great read",
...   "likes": 3
... },
... {
...   "postTitle": "my second post",
...   "comment": "good info",
...   "likes": 0
... },
... {
...   "postTitle": "my second post",
...   "comment": "i liked this post",
...   "likes": 12
... },
... {
...   "postTitle": "hello world",
...   "comment": "not my favorite",
...   "likes": 8
... },
... {
...   "postTitle": "my last post",
...   "comment": null,
...   "likes": 0
... }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("652d329f83c372d7ba29d0c1"),
    '1': ObjectId("652d329f83c372d7ba29d0c2"),
    '2': ObjectId("652d329f83c372d7ba29d0c3"),
    '3': ObjectId("652d329f83c372d7ba29d0c4"),
    '4': ObjectId("652d329f83c372d7ba29d0c5")
  }
}
```

# \$lookup

```
db.post2.aggregate([
  { $lookup:
    {
      from: "comment2",
      localField: "title",
      foreignField: "postTitle",
      as: "comments"
    }
  }
]).pretty()
```

```
{
  "_id" : ObjectId("6176a821795674f634e37a6d"),
  "title" : "my first post",
  "author" : "Jim",
  "likes" : 5,
  "comments": [
    {
      "_id" : ObjectId("6176a883795674f634e37a6e"),
      "postTitle" : "my first post",
      "comment" : "great read",
      "likes" : 3
    }
  ]
}
```

# \$lookup

```
hive> db.post2.aggregate([
...     { $lookup:
...         {
...             from: "comment2",
...             localField: "title",
...             foreignField: "postTitle",
...             as: "comments"
...         }
...     }
... ])
```

```
[{
  "_id": ObjectId("652d31a183c372d7ba29d0be"),
  "title": "my first post",
  "author": "Jim",
  "likes": 5,
  "comments: [
    {
      "_id": ObjectId("652d329f83c372d7ba29d0c1"),
      "postTitle": "my first post",
      "comment": "great read",
      "likes": 3
    }
  ],
  {
    "_id": ObjectId("652d31a183c372d7ba29d0bf"),
    "title": "my second post",
    "author": "Jim",
    "likes": 2,
    "comments: [
      {
        "_id": ObjectId("652d329f83c372d7ba29d0c2"),
        "postTitle": "my second post",
        "comment": "good info",
        "likes": 0
      },
      {
        "_id": ObjectId("652d329f83c372d7ba29d0c3"),
        "postTitle": "my second post",
        "comment": "i liked this post",
        "likes": 12
      }
    ],
    {
      "_id": ObjectId("652d31a183c372d7ba29d0c0"),
      "title": "hello world",
      "author": "Joe",
      "likes": 3,
      "comments: [
        {
          "_id": ObjectId("652d329f83c372d7ba29d0c4"),
          "postTitle": "hello world",
          "comment": "not my favorite",
          "likes": 8
        }
      ]
    }
  ]
}]
```

# Next topics

- MongoDB backup and restore
- MongoDB Atlas
- MongoDB compass
- MongoDB authentication and authorization
- MongoDB connection via mongoose

# MongoDB - create backup (via system shell)

- mongodump

```
(base) ~/Desktop/2110322_DBSYS_2566_1 $ mongodump
2023-10-16T20:03:42.760+0700      writing admin.system.version to dump/admin/system.version.bson
2023-10-16T20:03:42.769+0700      done dumping admin.system.version (1 document)
2023-10-16T20:03:42.770+0700      writing hive.post to dump/hive/post.bson
2023-10-16T20:03:42.770+0700      writing test.post to dump/test/post.bson
2023-10-16T20:03:42.770+0700      writing hive.post2 to dump/hive/post2.bson
2023-10-16T20:03:42.771+0700      writing hive.comment2 to dump/hive/comment2.bson
2023-10-16T20:03:42.772+0700      done dumping hive.comment2 (5 documents)
2023-10-16T20:03:42.772+0700      done dumping hive.post (5 documents)
2023-10-16T20:03:42.773+0700      writing hive.post_w_options to dump/hive/post_w_options.bson
2023-10-16T20:03:42.773+0700      done dumping hive.post2 (3 documents)
2023-10-16T20:03:42.773+0700      done dumping hive.post_w_options (0 documents)
2023-10-16T20:03:42.774+0700      done dumping test.post (4 documents)
(base) ~/Desktop/2110322_DBSYS_2566_1 $
```

จาก current directory ที่เรียกคำสั่ง mongodump จะมี directory dump เกิดขึ้น

## MongoDB create backup (cont.)

```
cd dump/hive/  
ls
```

comment2.bson comment2.metadata.json	post.bson post.metadata.json	post2.bson post2.metadata.json	post_w_options.bson post_w_options.metadata.json
---	---------------------------------	-----------------------------------	---

Syntax	Description	Example
mongodump --host HOST_NAME --port PORT_NUMBER	This command will backup all databases of specified mongod instance.	mongodump --host tutorialspoint.com – port 27017
mongodump --dbpath DB_PATH --out BACKUP_DIRECTORY		mongodump --dbpath /data/db/ --out /data/backup/
mongodump --collection COLLECTION --db DB_NAME	This command will backup only specified collection of specified database.	mongodump --collection mycol --db test

## MongoDB export a specific collection (via system shell)

- mongoexport

```
(base) ~/Desktop/2110322_DBSYS_2566_1 $ mongoexport --collection=post --db=hive --out=post.json
2023-10-16T20:08:44.533+0700      connected to: mongodb://localhost/
2023-10-16T20:08:44.541+0700      exported 5 records
```

# MongoDB deployment (via system shell)

```
(base) ~/Desktop/2110322_DBSYS_2566_1 $ mongostat
insert query update delete getmore command dirty used flushes vsize    res qrw arw net_in net_out conn      time
*0  *0  *0  *0  0  0|0  0.0% 0.0%      0 393G 31.0M 0|0 0|0  111b  67.3k  27 Oct 16 20:09:38.461
*0  *0  *0  *0  0  2|0  0.0% 0.0%      0 393G 31.0M 0|0 0|0  221b  68.1k  27 Oct 16 20:09:39.462
*0  *0  *0  *0  0  3|0  0.0% 0.0%      0 393G 31.0M 0|0 0|0  222b  68.2k  27 Oct 16 20:09:40.462
*0  *0  *0  *0  0  0|0  0.0% 0.0%      0 393G 31.0M 0|0 0|0  111b  67.5k  27 Oct 16 20:09:41.462
*0  *0  *0  *0  0  1|0  0.0% 0.0%      0 393G 31.0M 0|0 0|0  112b  67.6k  27 Oct 16 20:09:42.461
*0  *0  *0  *0  0  0|0  0.0% 0.0%      0 393G 31.0M 0|0 0|0  111b  67.5k  27 Oct 16 20:09:43.462
*0  *0  *0  *0  0  4|0  0.0% 0.0%      0 393G 31.0M 0|0 0|0  514b  68.5k  27 Oct 16 20:09:44.461
*0  *0  *0  *0  0  2|0  0.0% 0.0%      0 393G 31.0M 0|0 0|0  246b  67.9k  27 Oct 16 20:09:45.461
*0  *0  *0  *0  0  0|0  0.0% 0.0%      0 393G 31.0M 0|0 0|0  111b  67.5k  27 Oct 16 20:09:46.462
*0  *0  *0  *0  0  2|0  0.0% 0.0%      0 393G 31.0M 0|0 0|0  174b  68.1k  27 Oct 16 20:09:47.459
insert query update delete getmore command dirty used flushes vsize    res qrw arw net_in net_out conn      time
*0  *0  *0  *0  0  1|0  0.0% 0.0%      0 393G 31.0M 0|0 0|0  259b  67.7k  27 Oct 16 20:09:48.462
*0  *0  *0  *0  0  3|0  0.0% 0.0%      0 393G 31.0M 0|0 0|0  222b  68.2k  27 Oct 16 20:09:49.462
*0  *0  *0  *0  0  2|0  0.0% 0.0%      0 393G 31.0M 0|0 0|0  221b  68.2k  27 Oct 16 20:09:50.462
*0  *0  *0  *0  0  0|0  0.0% 0.0%      0 393G 31.0M 0|0 0|0  111b  67.5k  27 Oct 16 20:09:51.462
```

- mongostat
- แสดงจำนวน ของคำสั่งใน DB inserts, queries, updates, deletes, และ cursors

# ตัวอย่างการทำงานของ cursors

## ใน MongoDB shell ลองคำสั่งต่อไปนี้

```
var myCursor = db.post.find()  
  
while (myCursor.hasNext()) {  
    printjson(myCursor.next());  
}  
}
```

	ns	total	read	write	
admin.\$cmd.aggregate		0ms	0ms	0ms	2023-10-16T20:11:38+07:00
admin.atlascli		0ms	0ms	0ms	
admin.system.version		0ms	0ms	0ms	
config.collections		0ms	0ms	0ms	
config.system.sessions		0ms	0ms	0ms	
config.transactions		0ms	0ms	0ms	
hive.comment2		0ms	0ms	0ms	
hive.post		0ms	0ms	0ms	
hive.post2		0ms	0ms	0ms	
hive.post_w_options		0ms	0ms	0ms	
	ns	total	read	write	2023-10-16T20:11:39+07:00
admin.\$cmd.aggregate		0ms	0ms	0ms	
admin.atlascli		0ms	0ms	0ms	
admin.system.version		0ms	0ms	0ms	
config.collections		0ms	0ms	0ms	
config.system.sessions		0ms	0ms	0ms	
config.transactions		0ms	0ms	0ms	
hive.comment2		0ms	0ms	0ms	
hive.post		0ms	0ms	0ms	
hive.post2		0ms	0ms	0ms	
hive.post_w_options		0ms	0ms	0ms	
	ns	total	read	write	2023-10-16T20:11:40+07:00
admin.\$cmd.aggregate		0ms	0ms	0ms	
admin.atlascli		0ms	0ms	0ms	
admin.system.version		0ms	0ms	0ms	
config.collections		0ms	0ms	0ms	
config.system.sessions		0ms	0ms	0ms	
config.transactions		0ms	0ms	0ms	
hive.comment2		0ms	0ms	0ms	
hive.post		0ms	0ms	0ms	
hive.post2		0ms	0ms	0ms	
hive.post_w_options		0ms	0ms	0ms	

## MongoDB deployment

- mongotop NUMBER  
แสดง stats ของแต่ละคอลเลคชัน โดย NUMBER คือจำนวนวินาทีที่จะอัพเดตข้อมูลแต่ละรอบ

MongoDB Atlas



Products

Solutions

Resources

Company

Pricing



Sign In

Try Free

Database

Overview

Features

Deep Dive

Resources

Explore

View all platform details



ATLAS

# Database. Deploy a multi-cloud



<https://www.mongodb.com/cloud/atlas>

95

Atlas Duangdao's ... Access Manager Billing All Clusters Get Help Duangdao

Project 0 Data Services App Services Charts

Overview DEPLOYMENT Database Data Lake

SERVICES Device Sync Triggers Data API Data Federation Search Stream Processing

SECURITY M0 Cluster Provisioning... Database Access

DUANGDAO'S ORG - 2023-10-18 > PROJECT 0

## Security Quickstart

To access data stored in Atlas, you'll need to create users and set up network security controls. [Learn more about security setup](#)

1 How would you like to authenticate your connection?

Your first user will have permission to read and write any data in your project.

**Username and Password** **Certificate**

We autogenerated a username and password for your first database user in this project using your MongoDB Cloud registration information.

เลื่อนลงไปเพื่อสร้าง user/password  
มี autogenerated อู้ฟ์แต่เปลี่ยนได้

Atlas Duangdao's ... Access Manager Billing All Clusters Get Help Duangdao

Project 0 Data Services App Services Charts

Overview

**DEPLOYMENT**

- Database
- Data Lake

**SERVICES**

- Device Sync
- Triggers
- Data API
- Data Federation
- Search
- Stream Processing

**SECURITY**

**Quickstart**

- Backup
- Database Access
- Network Access

Where would you like to connect from?

Enable access for any network(s) that need to read and write data to your cluster.

**My Local Environment**

Use this to add network IP addresses to the IP Access List. This can be modified at any time.

**Cloud Environment**

Use this to configure network access between Atlas and your cloud or on-premise environment. Specifically, set up IP Access Lists, Network Peering, and Private Endpoints.

**ADVANCED**

We added your current IP address. You can connect to your cluster locally from this device.

Add entries to your IP Access List

Only an IP address you add to your Access List will be able to connect to your project's clusters. You can manage existing IP entries via the [Network Access Page](#).

IP Address	Description
Enter IP Address	Enter description
<b>Add My Current IP Address</b>	



Atlas   Duangdao's ...   Access Manager   Billing   All Clusters   Get Help   Duangdao

Project 0   Data Services   App Services   Charts

Overview   DEPLOYMENT   SERVICES   SECURITY

Database   Data Lake

Device Sync   Triggers   Data API   Data Federation   Search   Stream Processing

Add Data   Load Sample Data   Data Modeling Templates

+ Add Tag

DUANGDAO'S ORG - 2023-10-18 > PROJECT 0

## Overview

### Database Deployments

Cluster0   CONNECT   EDIT CONFIGURATION   FREE   SHARED

### Resources Center

#### Resources

GO   Golang Starter Application

AI/ML ENRICHED   Embedding GenAI into your App

LEARN   Learn MongoDB

GENERAL   Get Started with Atlas, Developer Center, Ask the MongoDB Community, Data Modeling Patterns, MongoDB Query API

## Connect to Cluster0

1 Set up connection security    2 Choose a connection method    3 Connect

### Connect to your application

 Drivers  
Access your Atlas data using MongoDB's native drivers (e.g. Node.js, Go, etc.)

### Access your data through tools

 Compass  
Explore, modify, and visualize your data with MongoDB's GUI

 Shell  
Quickly add & update data using MongoDB's Javascript command-line interface

 MongoDB for VS Code  
Work with your data in MongoDB directly from your VS Code environment



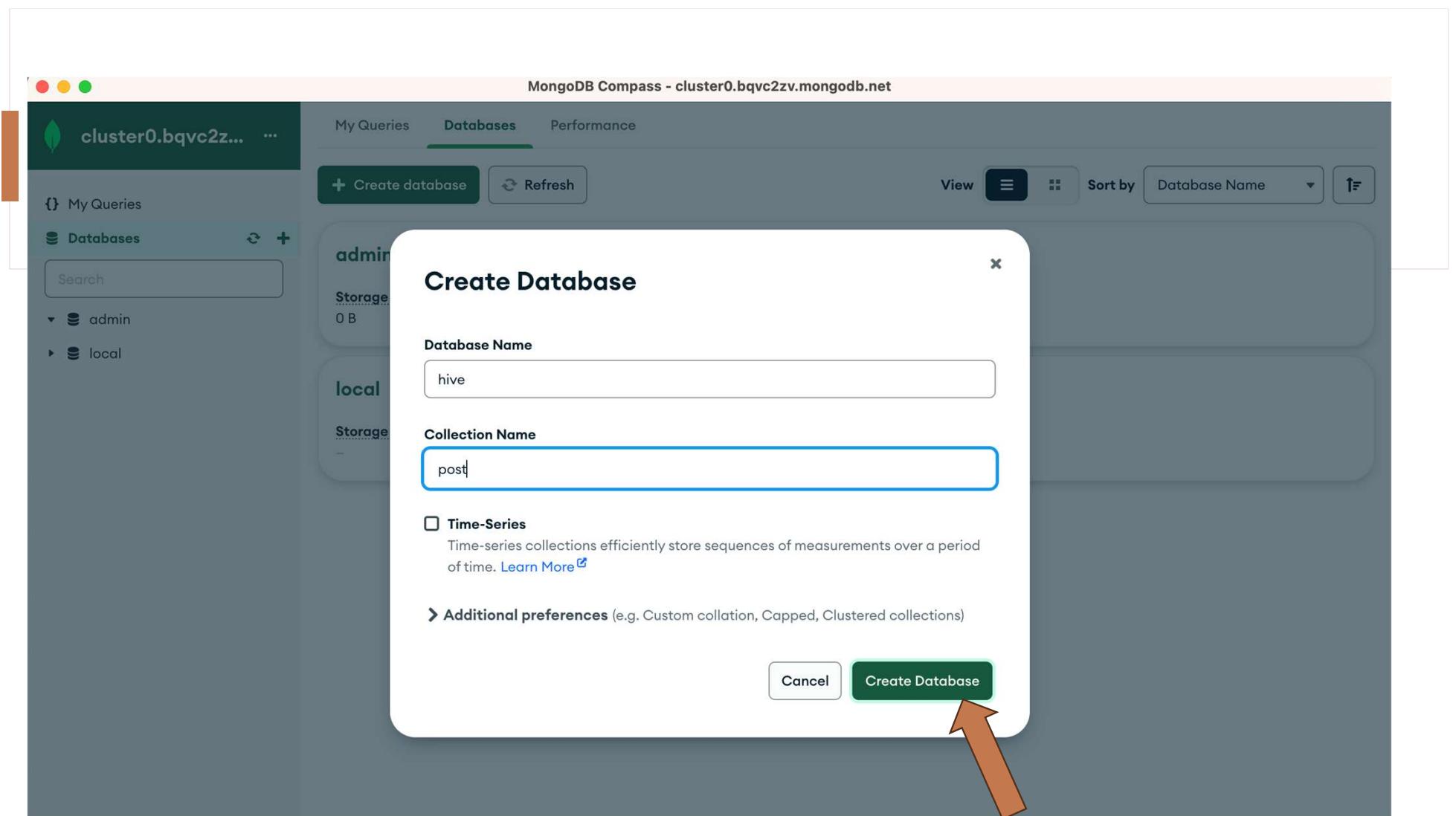
The screenshot shows the MongoDB Atlas interface with a modal window titled "Connect to Cluster0". The modal is divided into three steps: "Set up connection security" (completed), "Choose a connection method" (completed), and "Connect" (step 3). The "Choose a connection method" step is highlighted with a green border. Below it, there are two options: "I don't have MongoDB Compass installed" and "I have MongoDB Compass installed", with the latter being selected. The "I have MongoDB Compass installed" button is outlined in green. The "Connect" step contains instructions for connecting with MongoDB Compass, including a dropdown menu set to "1.12 or later" and a note about checking the "About Compass" section. It also provides a connection string: `mongodb+srv://wichadak:<password>@cluster0.bqvc2zv.mongodb.net/`. A green arrow points from the text "When entering your password, make sure that any special characters are URL encoded." to the copy icon (a clipboard with a plus sign) located next to the connection string. The left sidebar shows the project navigation with sections like Overview, Deployment, Services, Security, and Resources.

## Compass on local machine

The screenshot shows the MongoDB Compass application interface. On the left, there's a sidebar with 'Compass' at the top, followed by 'New connection +', 'Saved connections' (listing 'localhost' with a timestamp), and 'Recents'. The main area is titled 'New Connection' with the sub-instruction 'Connect to a MongoDB deployment'. It features a 'URI' input field containing the connection string 'mongodb+srv://wichadak:<password>@cluster0.bqvc2zv.mongodb.net/'. To the right of the input field is a 'FAVORITE' button with a star icon. Below the input field is a 'Edit Connection String' toggle switch. At the bottom of the main panel are three buttons: 'Save', 'Save & Connect', and 'Connect'. A large red arrow points from the text above towards the 'URI' input field. In the bottom-left corner of the main panel, there's a light green callout box with the text 'New to Compass and don't have a cluster?'. It includes a link to 'MongoDB Atlas' and a 'CREATE FREE CLUSTER' button. In the bottom-right corner, there's another callout box with the question 'How do I find my connection string in Atlas?'.

## Compass on local machine

A screenshot of the MongoDB Compass application interface. The title bar reads "MongoDB Compass - cluster0.bqvc2zv.mongodb.net". The main navigation bar has three tabs: "My Queries" (which is active and highlighted in green), "Databases", and "Performance". On the left, there's a sidebar with a green header containing the database name "cluster0.bqvc2zv...". Below the header, there are two sections: "My Queries" and "Databases". The "Databases" section includes a search bar labeled "Search" and lists two databases: "admin" and "local". A large orange arrow points from the text "Search your queries" in the introduction to the search bar in the sidebar. The main content area is titled "No saved queries yet." It features a magnifying glass icon inside curly braces and a sub-section with the text "Start saving your aggregations and find queries, you'll see them here." At the bottom, there's a link "Not sure where to start? Visit our Docs →".



MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post

cluster0.bqvc2zv... ... Documents hive.post +

My Queries Databases Search

hive.post

0 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } or [Generate query](#)

EXPLAIN RESET FIND OPTIONS

ADD DATA EXPORT DATA

Import JSON or CSV file

Insert document

0 - 0 of 0 ⏪ ⏴ ⏵ ⏩ ⏷ ⏸ ⏹

This collection has no data

It only takes a few seconds to import data from a JSON or CSV file.

Import Data

An orange arrow points to the 'Import JSON or CSV file' button in the center of the Compass interface.

MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post

cluster0.bqvc2zv...    Documents    +

My Queries    Databases    Search

hive.post    5 DOCUMENTS    1 INDEXES

Documents    Aggregations    Schema    Indexes    Validation

Filter    Type a query: { field: 'value' } or [Generate query](#)

Explain    Reset    Find    Options

1 - 5 of 5    ADD DATA    EXPORT DATA

`_id: ObjectId('652d13974ce35c5655398097')  
title: "MongoDB Overview"  
description: "An introduction to MongoDB / document database"  
owner: "lionking"  
url: "http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf"  
tags: Array (3)  
likes: 250`

`_id: ObjectId('652d139e4ce35c5655398098')  
title: "MongoDB Overview"  
description: "An introduction to MongoDB / document database"  
owner: "lionking"  
url: "http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf"  
tags: Array (3)  
likes: 250`

Import completed.  
5 documents imported.

105

MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post

cluster0.bqvc2z... ... Documents hive.post +

My Queries Databases Search

admin hive post local

hive.post

5 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter { owner:"lionking"} 1 Generate query Explain Reset Find Options ▾

+ ADD DATA EXPORT DATA 1 - 3 of 3

3

2

`_id: ObjectId('652d13974ce35c5655398097')  
title: "MongoDB Overview"  
description: "An introduction to MongoDB / document database"  
owner: "lionking"  
url: "http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf"  
tags: Array (3)  
likes: 250`

`_id: ObjectId('652d139e4ce35c5655398098')  
title: "MongoDB Overview"  
description: "An introduction to MongoDB / document database"  
owner: "lionking"  
url: "http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf"  
tags: Array (3)  
likes: 250`

<code>_id: ObjectId('652d13974ce35c5655398097') title: "MongoDB Overview" description: "An introduction to MongoDB / document database" owner: "lionking" url: "http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf" tags: Array (3) likes: 250</code>
<code>_id: ObjectId('652d139e4ce35c5655398098') title: "MongoDB Overview" description: "An introduction to MongoDB / document database" owner: "lionking" url: "http://mdslab.unime.it/sites/default/files/mongodb_tutorial.pdf" tags: Array (3) likes: 250</code>

MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post

cluster0.bqvc2zv...    +

My Queries    Databases    Search    +

hive.post    5 DOCUMENTS    1 INDEXES

Documents    Aggregations    Schema    Indexes    Validation

Filter {owner:"lionking"}

Project {owner:1}    2

Sort { field: -1 } or [ { field: 'owner' }, { field: -1 } ]

Collation { locale: 'simple' }

Generate query Explain Reset Find Options

Skip 0

MaxTimeMS 0

EXPORT DATA

1 - 3 of 3

4

3

1

```
_id: ObjectId('652d13974ce35c5655398097')
owner: "lionking"

_id: ObjectId('652d139e4ce35c5655398098')
owner: "lionking"

_id: ObjectId('652d139e4ce35c5655398099')
owner: "lionking"
```

MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post

cluster0.bqvc2zv... 250

Documents hive.post +

My Queries Databases Search

hive.post

5 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Filter {owner:"lionking"} 1

Project {owner:1, likes:1}

Sort { field: -1 } or [['field', -1]]

Collation { locale: 'simple' }

Generate query Explain Reset Find Options MaxTimeMS 0

Skip 0

EXPORT DATA 3 1-3 of 3

1

2

3

`_id: ObjectId('652d13974ce35c5655398097')  
owner: "lionking"  
likes: 250`

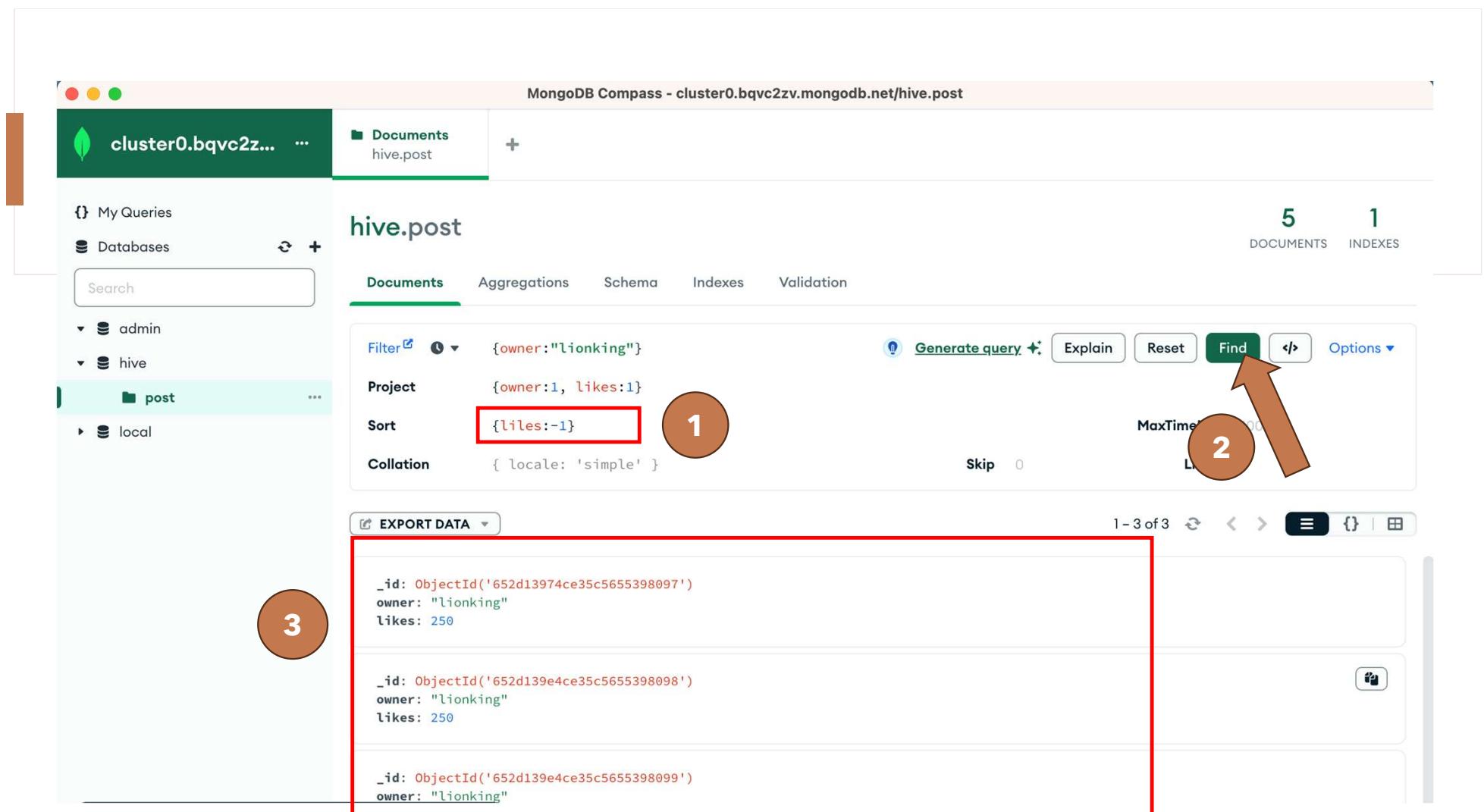
`_id: ObjectId('652d139e4ce35c5655398098')  
owner: "lionking"  
likes: 250`

`_id: ObjectId('652d139e4ce35c5655398099')  
owner: "lionking"`

The screenshot shows the MongoDB Compass interface for a database named 'hive.post'. The left sidebar lists databases: admin, hive, post (selected), and local. The main area displays the 'Documents' tab for the 'post' collection. A query is being run with the following parameters:

- Filter: {owner:"lionking"}
- Project: {owner:1, likes:1} (highlighted with a red box)
- Sort: { field: -1 } or [['field', -1]]
- Collation: { locale: 'simple' }

The results show three documents, each with an '\_id' field (ObjectId), an 'owner' field ('lionking'), and a 'likes' field (250). The third document's '\_id' is highlighted with a red box. An orange arrow labeled '2' points from the 'MaxTimeMS' button to this highlighted '\_id'. An orange circle labeled '3' is positioned near the bottom-left corner of the result list.



MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post

cluster0.bqvc2zv... ···

Aggregations  
hive.post

hive.post

1

5 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Pipeline Your pipeline is currently empty. [Generate aggregation](#)

Untitled - modified [SAVE](#) [CREATE NEW](#) [EXPORT TO LANGUAGE](#) [EXPLAIN](#) [EXPORT](#) [Run](#) [More Options](#)

PREVIEW {} STAGES TEXT

Stage 1 Select

1

No Preview Documents

2

+ Add Stage

MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post

cluster0.bqvc2zv... Aggregations +

My Queries Databases Search

hive.post 5 1 DOCUMENTS INDEXES

Documents Aggregations Schema Indexes Validation

Pipeline Your pipeline is currently empty. [Generate aggregation](#)

Untitled - modified [SAVE](#) [CREATE NEW](#) [EXPORT TO LANGUAGE](#) PREVIEW { STAGES } TEXT

Stage 1 \$group

1 \$group Groups documents by a specified expression.

No Preview Documents

+ Add Stage

111

MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post

cluster0.bqvc2zv... ... Aggregations hive.post +

My Queries Databases Search

hive.post 5 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Indexes Validation

Pipeline \$group Generate aggregation Explain Export Run More Options

Untitled - modified SAVE CREATE NEW EXPORT TO LANGUAGE PREVIEW {} STAGES TEXT

1

2

3

Output after \$group stage (Sample of 2 documents)

```
1  /***
2   * _id: The id of the group.
3   * fieldN: The first field name.
4   */
5  [
6    {
7      _id: "$owner",
8      total_likes: {
9        $sum: "$likes"
10     }
11   }
12 ]
```

```
_id: "lionking"
total_likes: 1000
```

```
_id: "Hikaru"
total_likes: 250
```

+ Add Stage Learn more about aggregate pipeline stages

```
1  /**
2   * _id: The id of the group.
3   * fieldN: The first field name.
4   */
5  [
6    {
7      _id: "$owner",
8      total_likes: {
9        $sum: "$likes"
10     }
11   }
12 ]
```

```
_id: "lionking"
total_likes: 1000
```

```
_id: "Hikaru"
total_likes: 250
```

MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post

cluster0.bqvc2zv... ... Aggregations +

My Queries Databases Search

Documents Aggregations Schema Indexes Validation

hive.post 5 1 DOCUMENTS INDEXES

Pipeline \$group Generate aggregation Explain Export Run More Options

Untitled - modified SAVE CREATE NEW EXPORT TO LANGUAGE PREVIEW STAGES TEXT

Stage 2 \$sort

1 \$sort Reorders the document stream by a specified sort key and direction.

\$sort Groups incoming documents based on the value of a specified expression, then computes the count of documents in each distinct group.

113

The screenshot shows the MongoDB Compass interface for a database named 'hive.post'. The left sidebar lists databases: admin, hive, and local, with 'post' selected. The main area displays an aggregation pipeline under the 'Aggregations' tab for the 'hive.post' collection. The pipeline consists of two stages: a '\$group' stage and a '\$sort' stage. The '\$sort' stage is highlighted with a red box and contains the following code:

```
1 /**
2  * Provide any number of field/order pair
3  */
4 {
5     total_likes: -1
6 }
```

The results of the '\$sort' stage are also highlighted with a red box, showing a sample of two documents:

```
_id: "lionking"
total_likes: 1000
```

```
_id: "Hikaru"
total_likes: 250
```

The screenshot shows the MongoDB Compass interface for a database named 'cluster0.bqvc2zv.mongodb.net/hive.post'. The 'Aggregations' tab is selected, displaying a pipeline with a '\$group' stage. The results pane shows two documents: one with \_id 'lionking' and total\_likes 1000, and another with \_id 'Hikaru' and total\_likes 250. A red box highlights the results, and a blue box highlights the pipeline editor. A brown circle labeled '1' points to the 'Run' button, and a brown circle labeled '2' points to the results list.

MongoDB Compass - cluster0.bqvc2zv.mongodb.net/hive.post

cluster0.bqvc2zv... ... Aggregations hive.post +

My Queries Databases Search

hive.post

Documents Aggregations Schema Indexes Validation

Pipeline Explain Export Run More Options ▾

ALL RESULTS OUTPUT OPTIONS ▾ Showing 1 – 2 count results VIEW ⌂ { }

5 DOCUMENTS 1 INDEXES

1

2

```
_id: "lionking"
total_likes: 1000

_id: "Hikaru"
total_likes: 250
```