Department of Computer Engineering Faculty of Engineering
Kasetsart University

HW# NoSQL & MongoDB
In this homework, you should create an evidence of your work such as picture with description in PDF and submit the Github link of your homework. Make sure that your Github link is public.

1) You're creating a database to contain a set of sensor measurements from a two-dimensional grid. Each measurement is a time-sequence of readings, and each reading contains ten labeled values. Should you use the relational model or MongoDB? Please justify your answer

MongoDB is a better choice for storing sensor measurements in a two-dimensional grid because it offers:

1. Flexible data structure (schema flexibility) to handle complex data.
2. Easier scalability to manage large volumes of data and high write loads.
3. Efficient querying and indexing for nested data without complex joins.

These features make MongoDB more suitable than a relational database for this scenario.

2) For each of the following applications

a. IoT
b. E-commerce
c. Gaming
d. Finance

Propose an appropriate Relational Model or MongoDB database schema. For each application, clearly justify your choice of database.

a. IoT:

- Choose MongoDB due to its flexible schema that can adapt to diverse data formats and requirements.
- MongoDB's horizontal scaling capabilities make it suitable for handling the large amounts of data generated by IoT devices.

```
{
    "deviceId": "unique_device_id",
    "deviceType": "sensor/actuator",
    "data": [
        {
            "timestamp": "2023-04-23 12:34:56",
            "value": ".....",
            "type" : "....."
        }
    ]
}
```

b. E-commerce:

- Choose a Relational Model because it provides strong support for complex queries and transactions, essential for maintaining data consistency and integrity in e-commerce systems.
- E-commerce applications have a structured data model with clear relationships between entities, which is well-suited for a relational database.

  Users:
  - user_id (PK)
  - username

  Products:
  - product_id (PK)
  - product_name

- price

Orders:
- order_id (PK)
- user_id (FK -> Users)

Order_Items:
- order_item_id (PK)
- order_id (FK -> Orders)
- product_id (FK -> Products)

c. Gaming:

- Choose MongoDB for its flexible schema, which can easily manage diverse data such as player profiles, game state, and in-game items.
- MongoDB's support for geospatial queries and horizontal scaling can be useful for multiplayer and location-based games.

d. Finance:

- Choose a Relational Model for its strong consistency, support for transactions, and ACID properties, which are critical for maintaining data accuracy and reliability in financial systems.
- Financial applications involve structured data with strong relationships and consistency requirements, making a relational database a suitable choice.

Users:
- user_id (PK)
- username

Accounts:
- account_id (PK)
- user_id (FK -> Users)
- account_type
- balance

Transactions:
- transaction_id (PK)
- from_account_id (FK -> Accounts)
- to_account_id (FK -> Accounts)
- amount

3) Create a MongoDB database with the following information.

({"name":"Ramesh","subject":"maths","marks":87})
({"name":"Ramesh","subject":"english","marks":59})
({"name":"Ramesh","subject":"science","marks":77})
({"name":"Rav","subject":"maths","marks":62})
({"name":"Rav","subject":"english","marks":83})
({"name":"Rav","subject":"science","marks":71})
({"name":"Alison","subject":"maths","marks":84})
({"name":"Alison","subject":"english","marks":82})
({"name":"Alison","subject":"science","marks":86})
({"name":"Steve","subject":"maths","marks":81})
({"name":"Steve","subject":"english","marks":89})
({"name":"Steve","subject":"science","marks":77})
({"name":"Jan","subject":"english","marks":0,"reason":"absent"})

```
TypeError: db.school.insertMant is not a function
[NoSQL_Exercises_65_2> db.school.insertMany([
... {
[... "name": "Ramesh",
... "subject": "maths",
... "marks": 87
... },
... {
... "name": "Ramesh",
... "subject": "english",
... "marks": 59
... },
... {
... "name": "Ramesh",
... "subject": "science",
... "marks": 77
... },
... {
... "name": "Rav",
... "subject": "maths",
... "marks": 62
... },
... {
... "name": "Rav",
... "subject": "english",
... "marks": 83
... },
... {
... "name": "Rav",
... "subject": "science",
... "marks": 71
... },
... {
... "name": "Alison",
... "subject": "maths",
... "marks": 84
... },
... {
... "name": "Alison",
... "subject": "english",
... "marks": 82
... },
... {
... "name": "Alison",
... "subject": "science",
... "marks": 86
... },
... {
... "name": "Steve",
... "subject": "maths",
... "marks": 81
... },
... {
... "name": "Steve",
... "subject": "english",
... "marks": 89
... },
... {
... "name": "Steve",
... "subject": "science",
... "marks": 77
... },
... {
... "name": "Jan",
... "subject": "english",
... "marks": 0,
... "reason": "absent"
... }])
```

```
{
[  acknowledged: true,
   insertedIds: {
     '0': ObjectId("6444fb2462ce56ff578ada66"),
     '1': ObjectId("6444fb2462ce56ff578ada67"),
     '2': ObjectId("6444fb2462ce56ff578ada68"),
     '3': ObjectId("6444fb2462ce56ff578ada69"),
     '4': ObjectId("6444fb2462ce56ff578ada6a"),
     '5': ObjectId("6444fb2462ce56ff578ada6b"),
     '6': ObjectId("6444fb2462ce56ff578ada6c"),
     '7': ObjectId("6444fb2462ce56ff578ada6d"),
     '8': ObjectId("6444fb2462ce56ff578ada6e"),
     '9': ObjectId("6444fb2462ce56ff578ada6f"),
     '10': ObjectId("6444fb2462ce56ff578ada70"),
     '11': ObjectId("6444fb2462ce56ff578ada71"),
     '12': ObjectId("6444fb2462ce56ff578ada72")
   }
}
```

Give MongoDB statements (with results) for the following queries

• Find the total marks for each student across all subjects.

```
[NoSQL_Exercises_65_2> db.school.aggregate([ { $group: { _id: "$name", totalMarks: { $sum: "$marks" } } }])
 [
[  { _id: 'Alison', totalMarks: 252 },
   { _id: 'Steve', totalMarks: 247 },
   { _id: 'Jan', totalMarks: 0 },
   { _id: 'Ramesh', totalMarks: 223 },
   { _id: 'Rav', totalMarks: 216 }
 ]
```

• Find the maximum marks scored in each subject.

```
NoSQL_Exercises_65_2> db.school.aggregate([ { $group: { _id: "$subject", maximum_mark_scorce: { $max: "$marks" } } }])
[
  { _id: 'maths', maximum_mark_scorce: 87 },
  { _id: 'english', maximum_mark_scorce: 89 },
  { _id: 'science', maximum_mark_scorce: 86 }
]
```

• Find the minimum marks scored by each student.

```
NoSQL_Exercises_65_2> db.school.aggregate([
{ $group:{
            _id: { name: "$name", subject: "$subject" },
           minMarks: { $min: "$marks" }
            }
           ])
[
 { _id: 'Alison', minimum_mark_scorce: 82 },
 { _id: 'Steve', minimum_mark_scorce: 77 },
 { _id: 'Jan', minimum_mark_scorce: 0 },
 { _id: 'Ramesh', minimum_mark_scorce: 59 },
 { _id: 'Rav', minimum_mark_scorce: 62 }
]
```

Or

```
NoSQL_Exercises_65_2> db.school.aggregate([
{ $sort: { marks: 1 } },
{ $group: {  _id: "$name",
             subject: { $first: "$subject" },
           minMarks: { $first: "$marks" }
           } },
{ $sort : { _id:1} }
           ])

[
 { _id: 'Alison', subject: 'english', minMarks: 82 },
 { _id: 'Jan', subject: 'english', minMarks: 0 },
 { _id: 'Ramesh', subject: 'english', minMarks: 59 },
 { _id: 'Rav', subject: 'maths', minMarks: 62 },
 { _id: 'Steve', subject: 'science', minMarks: 77 }
]
```

• Find the top two subjects based on average marks.

```
NoSQL_Exercises_65_2> db.school.aggregate([ {$group : {_id:"$subject",avg_mark:{$avg:"$marks"} } }, { $sort : {avg_marks:1  }},{ $limit : 2 }])
[
  { _id: 'maths', avg_mark: 78.5 },
  { _id: 'english', avg_mark: 62.6 }
]
```

Sorry for sending late because I was sick and took a week off from school, which I already informed Teacher Supaporn and other teachers. Sorry for sending late.