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

I will use MongoDB for this database because MongoDB uses a document-based model that allows for flexible and dynamic data structures. This is particularly useful for storing sensor data, which is real-time data that can vary in format and structure. Additionally, MongoDB can handle large volumes of data and allows for easy scalability.

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 : MongoDB

Schema:

```
{
  device_id: ObjectId,
  timestamp: ISODate,
  sensor_data: {
    sensor_1: {reading_1: value, reading_2: value,...},
    sensor_2: {reading_1: value, reading_2: value,...},
    ...
  }
}
```

b. E-commerce : MongoDB

```
{
  customer_id: ObjectId,
  order_time: ISODate,
  order_items: {
    item_1: {item_id: value, item_name: value,...}
```

```
    item_2: {item_id: value, item_name: value,...}
  }
}
```

c. Gaming: MongoDB

```
{
  user_id: ObjectId,
  game_id: ObjectId,
  score: int,
  achievements: [ObjectId],
  items: {
    item_1: {quantity: int, level: int, ...},
    item_2: {quantity: int, level: int, ...},
    ...
  },
  friends: [ObjectId],
  ...
}
```

d. Finance: relational model

```
accounts (account_id, name, balance,...)
transactions (transaction_id, account_id, transaction_date, amount)
```

3) Create MongoDB database with following information.

1) ({"name":"Ramesh","subject":"maths","marks":87})
2) ({"name":"Ramesh","subject":"english","marks":59})
3) ({"name":"Ramesh","subject":"science","marks":77})
4) ({"name":"Rav","subject":"maths","marks":62})
5) ({"name":"Rav","subject":"english","marks":83})
6) ({"name":"Rav","subject":"science","marks":71})
7) ({"name":"Alison","subject":"maths","marks":84})
8) ({"name":"Alison","subject":"english","marks":82})
9) ({"name":"Alison","subject":"science","marks":86})
10) ({"name":"Steve","subject":"maths","marks":81})
11) ({"name":"Steve","subject":"english","marks":89})
12) ({"name":"Steve","subject":"science","marks":77})
13) ({"name":"Jan","subject":"english","marks":0,"reason":"absent"})

Give MongoDB statements (with results) for the following queries

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

> db.marks.aggregate([{$group:{_id:'$name', totalt_marks: {$sum: '$marks'}}}])

```
[
  { _id: 'Ramesh', totalt_marks: 223 },
  { _id: 'Rav', totalt_marks: 216 },
  { _id: 'Jan', totalt_marks: 0 },
  { _id: 'Alison', totalt_marks: 252 },
  { _id: 'Steve', totalt_marks: 247 }
]
```

• Find the maximum marks scored in each subject.

> db.marks.aggregate([{$group: {_id:'$subject', max_mark: {$max: '$marks'}}}])

```
[
  { _id: 'maths', max_mark: 87 },
  { _id: 'english', max_mark: 89 },
  { _id: 'science', max_mark: 86 }
]
```

• Find the minimum marks scored by each student.

> db.marks.aggregate([{$group: {_id:'$name', min_mark: {$min: '$marks'}}}])

```
[
  { _id: 'Jan', min_mark: 0 },
  { _id: 'Ramesh', min_mark: 59 },
  { _id: 'Alison', min_mark: 82 },
  { _id: 'Steve', min_mark: 77 },
  { _id: 'Rav', min_mark: 62 }
]
```

• Find the top two subjects based on average marks.

> db.marks.aggregate([{$group: {_id:'$subject', average_mark: {$avg: '$marks'}}}, {$sort:{average_mark:-1}}, {$limit: 2}])

```
[
  { _id: 'maths', average_mark: 78.5 },
  { _id: 'science', average_mark: 77.75 }
]
```