6410545495 Naphatson Phuangmanee

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

• It depends on my specific needs and use case. The relational model is better suited for structured data with clearly defined relationships between tables, whereas MongoDB is better suited for unstructured or semi-structured data. If the sensor measurements are highly structured and require complex queries involving multiple tables with well-defined relationships, a relational database may be a better choice. However, if the sensor measurements are more unstructured or semi-structured, and if you need to be able to easily add or modify fields without altering the entire schema, then MongoDB could be a good fit.

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: For IoT, a NoSQL database like MongoDB would be a better choice as the data generated from IoT devices is typically unstructured or semi-structured, and it can be difficult to predict the data format in advance. MongoDB can store and query such data easily with its flexible document-oriented data model.

• b. E-commerce: E-commerce applications typically involve structured data with well-defined relationships between tables, such as customer data, order data, and product data. For this reason, a relational database like MySQL or PostgreSQL may be a better choice, as it can handle complex transactions and queries across multiple tables.

• c. Gaming: Gaming applications often require handling a large volume of unstructured or semi-structured data, such as player data, game session data, and leaderboard data. For this reason, a NoSQL database like MongoDB would be a better choice, as it can handle such data easily with its flexible data model.

• d. Finance: Finance applications typically require strict data consistency and strong transactional support. For this reason, a relational database like Oracle or SQL Server may be a better choice, as they offer features like ACID transactions, strict data consistency, and data integrity checks.

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"})

Create MongoDB database from above information.

```
switched to db marks
marks> db.students.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":"Stev
e","subject":"science","marks":77},    {"name":"Jan","subject":"english","marks":0,"reason":"absent"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("64293cf6a8b554addde66d8d"),
    '1': ObjectId("64293cf6a8b554addde66d8e"),
    '2': ObjectId("64293cf6a8b554addde66d8f"),
    '3': ObjectId("64293cf6a8b554addde66d90"),
    '4': ObjectId("64293cf6a8b554addde66d91"),
    '5': ObjectId("64293cf6a8b554addde66d92"),
    '6': ObjectId("64293cf6a8b554addde66d93"),
    '7': ObjectId("64293cf6a8b554addde66d94"),
    '8': ObjectId("64293cf6a8b554addde66d95"),
    '9': ObjectId("64293cf6a8b554addde66d96"),
    '10': ObjectId("64293cf6a8b554addde66d97"),
    '11': ObjectId("64293cf6a8b554addde66d98"),
    '12': ObjectId("64293cf6a8b554addde66d99")
  }
}
```

Give MongoDB statements (with results) for the following queries.

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

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

• Find the maximum marks scored in each subject.

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

• Find the minimum marks scored by each student.

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

• Find the top two subjects based on average marks.

```
marks> db.students.aggregate([ { $group: { _id: "$subject", avgMarks: { $avg: "$marks" } } },{$sort:{avgMarks:-1}},{$limit:2}])
[
  { _id: 'maths', avgMarks: 78.5 },
  { _id: 'science', avgMarks: 77.75 }
]
```