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

Because of the structured nature of the data and the well-defined relationships between entities, it is recommended that sensor measurements from a two-dimensional grid be stored in a relational database. Because the relational model is better suited for structured data and offers strong consistency guarantees, it is a better fit for this use case. While MongoDB can handle complex data structures, it may not be the best choice for this use case due to weaker consistency guarantees and well-defined entity relationships.

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

Choice of database: MongoDB. MongoDB's document-based data model and flexible schema make it well-suited for storing and processing unstructured or semi-structured data, such as the measurements collected by IoT devices. MongoDB also has good support for geospatial data, which could be useful for storing the locations of devices.

Collections : Devices, measurement

b. E-commerce

Choice of database: Relational database. E-commerce applications typically involve a high volume of structured data, such as products, customers, orders, and payments, which are well-suited to a relational data model. Relational databases also have strong support for transactions and data integrity constraints, which are important for ensuring the accuracy and consistency of financial data.

Collections : Collection of Products with id, name, desc, price, customers, orders, payment

c. Gaming

Choice of database: MongoDB

Reason: Gaming applications often involve large amounts of data with complex relationships, such as player profiles, game statistics, and social connections. A NoSQL database like MongoDB can be well-suited to handling this type of data, as it allows for flexible schema design and can easily scale horizontally to handle large volumes of data.

Collections : Player with a player information such as email , id, name etc. and games collections.

d. Finance

Choice of database: Relational database. Finance applications involve sensitive financial data that requires strong data integrity constraints and transactional support. Relational databases are well-suited for this type of application, as they provide strong data consistency and reliability through features such as ACID transactions and foreign key constraints.

Collections : accounts with balance, id and transactions

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 a database with

  docker run --name <<Name>> -e MONGO_INITDB_ROOT_USERNAME=myAdmin -e MONGO_INITDB_ROOT_PASSWORD=P@ssw0rd -d mongo

Give MongoDB statements (with results) for the following queries

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

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

• Find the maximum marks scored in each subject.

```
school> db.marks.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.

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

• Find the top two subjects based on average marks.

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