MEMBER:Yue Ma , Yuheng Li and Conger Yang

BASIC TASKS

1.

MongoDB handles relationships between documents through embedded documents, reference documents, and reference databases. The first is embedded document, people can use MongoDB to embed a new document directly into the existing document, which is suitable for one-to-many relationship; Second, people in MongoDB can reference another document by storing a field id, which is suitable for one-to-one or one-to-many relationships; Finally, the relationship between documents is handled by referring to the database.

2.

The reason of MongoDB is schema-less is: first, compared to postgresql, MongoDB does not need to determine the type of identifiers and inserted data when creating documents; Second, MongoDB does not have the same strict definition and format as postgresql. In general, MongoDB is more free to edit documents and is able to handle large amounts of data.

3.

From the above three examples, MongoDB's dynamic mode is reflected in the flexibility and variability of keys, such as the first two documents content is contact, to the third can be replaced by NULL, you can directly omit and not write, you do not need to define the collection of schemas (i.e. fields and types) before inserting the document. Documents in a collection can have different fields and structures. MongoDB automatically manages the structure of the collection based on the inserted documents.

MEDIUM TASKS

4)

```
>_MONGOSH
> db.employee.insertMany([
    { empNo: 'E55', lastName: 'Carmen', firstName: 'Silva', hours: 450, gender: 'F' },
    { empNo: 'E44', lastName: 'Ryad', firstName: 'Patel', hours: 380, gender: 'M' },
    { empNo: 'E66', lastName: 'Susan', firstName: 'Joyner', hours: 350, gender: 'F' },
    { empNo: 'E77', lastName: 'Waiman', firstName: 'Zhu', hours: 400, gender: 'M' }
  ]);
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('670fdc6fdc4d2694804fa323'),
      '1': ObjectId('670fdc6fdc4d2694804fa324'),
      '2': ObjectId('670fdc6fdc4d2694804fa325'),
      '3': ObjectId('670fdc6fdc4d2694804fa326')
    }
  }
> db.employee.find({}, { firstName: 1, lastName: 1, _id: 0 });
< {
    lastName: 'Carmen',
    firstName: 'Silva'
  }
  {
    lastName: 'Ryad',
    firstName: 'Patel'
  }
  {
    lastName: 'Susan',
    firstName: 'Joyner'
  }
  {
```

```
>_MONGOSH
    lastName: 'Waiman',
    firstName: 'Zhu'
  }
> db.employee.find({ hours: { $gt: 400 } }, { empNo: 1, lastName: 1, hours: 1, _id: 0 });
< {
    empNo: 'E55',
    lastName: 'Carmen',
    hours: 450
  }
> db.employee.find({ lastName: /^S/ }, { lastName: 1, _id: 0 });
< {
    lastName: 'Susan'
  }
> db.employee.find({ gender: 'F' }, { firstName: 1, lastName: 1, _id: 0 }).sort({ firstName: 1 });
< {
    lastName: 'Susan',
    firstName: 'Joyner'
  }
  {
    lastName: 'Carmen',
    firstName: 'Silva'
  }
> db.employee.countDocuments({ hours: { $lt: 400 } });
< 2
> db.employee.find({ $or: [{ gender: 'F' }, { hours: { $gt: 350 } }] }, { firstName: 1, lastName: 1, _id: 0 }).sort({ lastName: 1 });
< {
    lastName: 'Carmen',
    firstName: 'Silva'
  }
```

```
  {
    lastName: 'Ryad',
    firstName: 'Patel'
  }
  {
    lastName: 'Susan',
    firstName: 'Joyner'
  }
  {
    lastName: 'Waiman',
    firstName: 'Zhu'
  }
test>
```

5)

a) Big data revolutionizes the way we handle and analyze information through its "V" characteristics—volume, velocity, variety, variability, veracity, and value. These traits not only define the scale and complexity of big data but also drive the need for new storage, processing, and analytical technologies to extract valuable insights from massive amounts of

data.

b) The variety in big data refers to the wide range of data types, including structured, semi-structured, and unstructured data. This variety poses challenges in data integration, standardization, and interpretation because data from different sources and formats needs to be uniformly processed and analyzed to ensure the accuracy and usability of the analysis results.

c) MongoDB addresses data variety with its flexible document model, dynamic schema, and powerful aggregation framework. It allows storing documents of different structures in the same collection without a predefined schema, supports multiple data types, and provides complex data processing operations, making it simple and straightforward to extract information from diverse data.

ADVANCED TASKS

6.

i)

```
db.books.find({"year": {"$gte": "2019", "$lte": "2024"}})
```

ii)

```
db.books.find({"book_id": {"$ne": "552020"}})
```

iii)

```
db.books.find({"$or": [{"author": "D. Sullivan"}, {"ISBN": "9780134023212"}]})
```

iv)

```
db.books.find({"$or": [{"ISBN": "9876543210"}, {"ISBN": "0123456789"}]})
```

v)

```
db.books.find({"title": {"$regex": "SQL"}})
```

vi)

```
db.books.aggregate([
    { "$match": { "publisher": "Addison-Wesley" } },
    { "$group": { "_id": null, "count": { "$sum": 1 } } }
])
```

vii)

```
db.books.find({"year": "2019", "title": {"$regex": "Mortals"}}).sort({"title": 1})
```