**Cao kuangzheng, Li Xudong, Qiu Yuyue**

**Basic Tasks**

1. **Handling Relationships in MongoDB**:
   * **Explanation**: MongoDB manages relationships between documents using two main methods:
     + **Embedded Documents**: Embedding related documents directly within another document (useful for one-to-one or one-to-many relationships).
     + **References**: Storing ObjectIDs of related documents in fields, which can then be used to fetch related data separately (useful for many-to-many or complex relationships).
   * Example: For customer data, the customer document could include an embedded array of orders or a reference to another collection storing orders.
2. **Comparison with Relational Databases**:
   * **SQL Approach**: In relational databases like PostgreSQL, a table and relationships are explicitly defined using schema (columns, data types, constraints, primary and foreign keys).
   * **MongoDB Approach**: MongoDB is schema-less, meaning collections do not require a predefined structure. Documents can have different structures, making MongoDB more flexible. This allows for easier evolution of data models over time without complex migrations.
3. **Dynamic Schema in MongoDB**:
   * **Explanation**: In MongoDB, documents within a collection can have different fields. For example:
     + { custId: “abc123”, contact: 4422, loan: 1000 }
     + { custId: “aa33”, contact: 1335, credit: “high” }
     + { custId: “abc123”, Profession: “IT specialist” }
   * The second document includes a "credit" field, while the third introduces a new "Profession" field. MongoDB’s dynamic schema allows this flexibility, unlike rigid SQL schemas.

**Medium Tasks**

1. **SQL to MongoDB Translation**:

CREATE TABLE employee (

empNo CHAR(9) PRIMARY KEY NOT NULL,

lastName CHAR(25) NOT NULL,

firstName VARCHAR(25) NOT NULL,

hours INT,

gender CHAR

);

db.employee.insertOne({

empNo: 'E55',

lastName: 'Carmen',

firstName: 'Silva',

hours: 450,

gender: 'F'

});

INSERT INTO employee (empNo, lastName, firstName, hours, gender)

VALUES('E55','Carmen','Silva',450,'F');

db.employee.insertMany([

{ 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' }

]);

SELECT firstName, lastName FROM employee;

db.employee.find({}, { firstName: 1, lastName: 1, \_id: 0 });

SELECT empNo, lastName, hours FROM employee WHERE hours > 400;

db.employee.find({ hours: { $gt: 400 } }, { empNo: 1, lastName: 1, hours: 1 });

**Advanced Tasks**

1. **Characteristics of Big Data (Vs)**:
   * **Nature of Big Data**: Big Data typically has characteristics referred to as the "4 Vs": Volume (large amounts of data), Velocity (fast-moving data), Variety (different types of data), and Veracity (trustworthiness and quality of data).
   * **Variety of Data**: Refers to the different formats in which data can exist, such as structured, semi-structured, and unstructured data (e.g., text, videos, sensor data).
   * **Handling Variety in MongoDB**: MongoDB’s flexible and schema-less nature allows it to handle data in varied formats using BSON (Binary JSON). This enables the storage of different data types within the same collection.
2. **MongoDB Operations on Book Documents**:

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

db.books.find({ book\_id: { $ne: "552020" } });

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

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

db.books.find({ title: /SQL/ });

db.books.count({ publisher: "Addison-Wesley" });

db.books.find({ year: 2019, title: /Mortals/ }).sort({ title: 1 });

db.books.updateMany({ year: 2019 }, { $set: { subject: "computing" } });