BIG DATA

SECTION D

Fall’2023

Yogya Sharma

ys5250

No-SQL database and Mongo DB

a) A one-page summary about No-SQL databases

NoSQL databases constitute a game-changing approach to data management, addressing the limitations of traditional relational databases. These systems are engineered for versatility and scalability, ushering in a new era of data storage and retrieval.

1. Data Diversity: NoSQL databases embrace an assortment of data types, whether structured, semi-structured, or unstructured. This adaptability accommodates the dynamic, multifaceted data landscape that characterizes the digital era.

2. Schema Flexibility: Unlike the structured, table-centric design of SQL databases, NoSQL systems champion schema flexibility. They eliminate the need for rigid data models, empowering users to mold their data storage on-the-fly. This feature is pivotal for modern, ever-evolving data environments.

3. Scalability: NoSQL databases specialize in horizontal scalability. By distributing data across multiple servers or clusters, they excel in handling immense datasets and high user traffic. This scalability factor is mission-critical for contemporary web applications and Big Data endeavors.

4. NoSQL Varieties: NoSQL databases are not a one-size-fits-all proposition. They comprise diverse types, each suited for distinct use cases:

- Document-Based: Ideal for storing semi-structured data, exemplified by MongoDB.
- Column-Family: Optimized for managing data in columns rather than rows, as seen in Apache Cassandra.
- Key-Value Stores: Utilize a simple key-value data model, such as Redis.
- Graph Databases: Prioritize data relationships, typified by Neo4j.

5. Real-World Utilization:
NoSQL databases are employed in an array of applications:

- Web Applications: NoSQL databases cope seamlessly with surges in user traffic and swiftly changing data, making them a preferred choice for modern web applications, e-commerce platforms, and social media sites. They ensure fast response times, even during peak usage.

- Big Data: These databases are the backbone of handling and dissecting extensive datasets in Big Data applications. They excel in distributed data storage and processing, making them indispensable for data-driven insights in various domains, including finance, healthcare, and marketing.

- IoT (Internet of Things): The dynamic and flexible nature of NoSQL databases is ideal for managing data generated by IoT devices. They can efficiently handle high volumes of device-generated data, facilitating real-time analytics, predictive maintenance, and device monitoring.

- Content Management: NoSQL databases are well-suited for organizing various content types, from articles and images to videos and metadata. They underpin content management systems, ensuring efficient storage, retrieval, and presentation of diverse content.

6. Challenges: While NoSQL databases offer numerous benefits, they demand thoughtful consideration. Selecting the appropriate database type and structure for a specific application is pivotal. Additionally, maintaining data consistency and security in NoSQL environments can pose complex challenges.

In summation, NoSQL databases mark a transformative shift in data management, delivering agility, scalability, and adaptability. Their burgeoning role in contemporary data ecosystems underscores their significance in addressing the evolving demands of data-driven applications and analytical processes.

b) A table (chart) about comparison among different types of No-SQL databases.

Database Type	Data Model	Query Language	Storage Model
Document Store	JSON/BSON	Query Language (e.g. SQL)	Collection Based
Key-Value Store	Key-Value Pair	NoSQL (operations)	Raw
Column Family	Columnar	Query Language (e.g. CQL)	Column Based
Graph Database	Graph	Graph Query Language	Property Graph (Vertices, Edges)

Database Type	Consistency Model	Use Cases
Document Store	Eventual Consistency	Content Management, User Profiles, Catalogs, Blogs, E-commerce
Key-Value Store	Eventual Consistency	Caching, Session Management, User Preferences, Personalization Engines, Distributed Storage
Column Family	Strong Consistency	Time-Series Data, Sensor Data, Sensor Data, Logging, Financial Data
Graph Database	Strong Consistency	Social Networks, Recommendation Systems, Fraud Detection, Knowledge Graphs

c) Create a test database and sample data in Mongo DB

```
testdb> use ys5250_testdb
switched to db ys5250_testdb
ys5250_testdb> db.sample.insertOne({ name: "John", age: 30, city: "New York" })
{
  acknowledged: true,
  insertedId: ObjectId("6541d28cfecaf2cec748b498")
}
ys5250_testdb> db.sample.insertOne({ name: "Alice", age: 25, city: "Los Angeles" })
{
  acknowledged: true,
  insertedId: ObjectId("6541d291fecaf2cec748b499")
}
ys5250_testdb> db.sample.insertOne({ name: "Michael", age: 35, city: "Chicago" })
{
  acknowledged: true,
  insertedId: ObjectId("6541d297fecaf2cec748b49a")
}
ys5250_testdb> db.sample.insertOne({ name: "Emma", age: 28, city: "Houston" })
{
  acknowledged: true,
  insertedId: ObjectId("6541d29cfecaf2cec748b49b")
}
ys5250_testdb> db.sample.insertOne({ name: "William", age: 22, city: "San Francisco" })
{
  acknowledged: true,
  insertedId: ObjectId("6541d2a1fecaf2cec748b49c")
}
ys5250_testdb> db.sample.insertOne({ name: "Olivia", age: 32, city: "Miami" })
{
  acknowledged: true,
  insertedId: ObjectId("6541d2a8fecaf2cec748b49d")
}
ys5250_testdb> db.sample.insertOne({ name: "James", age: 29, city: "Boston" })
{
  acknowledged: true,
  insertedId: ObjectId("6541d2aefecaf2cec748b49e")
}
ys5250_testdb> db.sample.insertOne({ name: "Sophia", age: 27, city: "Seattle" })
{
  acknowledged: true,
  insertedId: ObjectId("6541d2b4fecaf2cec748b49f")
}
ys5250_testdb> db.sample.insertOne({ name: "Robert", age: 31, city: "Dallas" })
{
  acknowledged: true,
  insertedId: ObjectId("6541d2b9fecaf2cec748b4a0")
}
ys5250_testdb> db.sample.insertOne({ name: "Charlotte", age: 26, city: "Denver" })
{
  acknowledged: true,
  insertedId: ObjectId("6541d2c0fecaf2cec748b4a1")
```

```
ys5250_testdb> db.sample.find().pretty()
[
  {
    _id: ObjectId("6541d28cfecaf2cec748b498"),
    name: 'John',
    age: 31,
    city: 'New York'
  },
  {
    _id: ObjectId("6541d297fecaf2cec748b49a"),
    name: 'Michael',
    age: 35,
    city: 'Chicago'
  },
  {
    _id: ObjectId("6541d29cfecaf2cec748b49b"),
    name: 'Emma',
    age: 28,
    city: 'Houston'
  },
  {
    _id: ObjectId("6541d2a1fecaf2cec748b49c"),
    name: 'William',
    age: 22,
    city: 'San Francisco'
  },
  {
    _id: ObjectId("6541d2a8fecaf2cec748b49d"),
    name: 'Olivia',
    age: 32,
    city: 'Miami'
  },
  {
    _id: ObjectId("6541d2aefecaf2cec748b49e"),
    name: 'James',
    age: 29,
    city: 'Boston'
  },
  {
    _id: ObjectId("6541d2b4fecaf2cec748b49f"),
    name: 'Sophia',
    age: 27,
    city: 'Seattle'
  },
  {
    _id: ObjectId("6541d2b9fecaf2cec748b4a0"),
    name: 'Robert',
    age: 31,
    city: 'Dallas'
  },
  {
    _id: ObjectId("6541d2c0fecaf2cec748b4a1"),
    name: 'Charlotte',
    age: 26,
    city: 'Denver'
  },
  {
    _id: ObjectId("6541d3a5fecaf2cec748b4a2"),
    name: 'Alice',
    age: 25,
    city: 'Los Angeles'
  }
]
```

d) Show screenshots of CURD operation on test database

Create:

```
ys5250_testdb> db.sample.insertOne({ name: "Charlotte", age: 26, city: "Denver" })
{
  acknowledged: true,
  insertedId: ObjectId("6541d2c0fecaf2cec748b4a1")
}
```

Read:

```
ys5250_testdb> db.sample.find({ city: "New York" }, { name: 1, age: 1, _id: 0 })
[ { name: 'John', age: 30 } ]
ys5250_testdb>
```

Update:

```
ys5250_testdb> db.sample.find({ city: "New York" }, { name: 1, age: 1, _id: 0 })
[ { name: 'John', age: 30 } ]
ys5250_testdb>

ys5250_testdb> db.sample.updateOne({ name: "John" }, { $set: { age: 31 } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
ys5250_testdb> db.sample.find({ city: "New York" }, { name: 1, age: 1, _id: 0 })
[ { name: 'John', age: 31 } ]
```

Delete:

Initial db:

```
[
  {
    _id: ObjectId("6541d28cfecaf2cec748b498"),
    name: 'John',
    age: 31,
    city: 'New York'
  },
  {
    _id: ObjectId("6541d297fecaf2cec748b49a"),
    name: 'Michael',
    age: 35,
    city: 'Chicago'
  },
  {
    _id: ObjectId("6541d29cfecaf2cec748b49b"),
    name: 'Emma',
    age: 28,
    city: 'Houston'
  },
  {
    _id: ObjectId("6541d2a1fecaf2cec748b49c"),
    name: 'William',
    age: 22,
    city: 'San Francisco'
  },
  {
    _id: ObjectId("6541d2a8fecaf2cec748b49d"),
    name: 'Olivia',
    age: 32,
    city: 'Miami'
  },
  {
    _id: ObjectId("6541d2aefecaf2cec748b49e"),
    name: 'James',
    age: 29,
    city: 'Boston'
  },
  {
    _id: ObjectId("6541d2b4fecaf2cec748b49f"),
    name: 'Sophia',
    age: 27,
    city: 'Seattle'
  },
  {
    _id: ObjectId("6541d2b9fecaf2cec748b4a0"),
    name: 'Robert',
    age: 31,
    city: 'Dallas'
  },
  {
    _id: ObjectId("6541d2c0fecaf2cec748b4a1"),
    name: 'Charlotte',
    age: 26,
    city: 'Denver'
  },
  {
    _id: ObjectId("6541d3a5fecaf2cec748b4a2"),
    name: 'Alice',
    age: 25,
    city: 'Los Angeles'
  }
]
```

Delete command:

```
ys5250_testdb> db.sample.deleteOne({ name: "Alice" })
{ acknowledged: true, deletedCount: 1 }
```

Updated db:

```
ys5250_testdb> db.sample.find().pretty()
[
  {
    _id: ObjectId("6541d28cfecaf2cec748b498"),
    name: 'John',
    age: 31,
    city: 'New York'
  },
  {
    _id: ObjectId("6541d297fecaf2cec748b49a"),
    name: 'Michael',
    age: 35,
    city: 'Chicago'
  },
  {
    _id: ObjectId("6541d29cfecaf2cec748b49b"),
    name: 'Emma',
    age: 28,
    city: 'Houston'
  },
  {
    _id: ObjectId("6541d2a1fecaf2cec748b49c"),
    name: 'William',
    age: 22,
    city: 'San Francisco'
  },
  {
    _id: ObjectId("6541d2a8fecaf2cec748b49d"),
    name: 'Olivia',
    age: 32,
    city: 'Miami'
  },
  {
    _id: ObjectId("6541d2aefecaf2cec748b49e"),
    name: 'James',
    age: 29,
    city: 'Boston'
  },
  {
    _id: ObjectId("6541d2b4fecaf2cec748b49f"),
    name: 'Sophia',
    age: 27,
    city: 'Seattle'
  },
  {
    _id: ObjectId("6541d2b9fecaf2cec748b4a0"),
    name: 'Robert',
    age: 31,
    city: 'Dallas'
  },
  {
    _id: ObjectId("6541d2c0fecaf2cec748b4a1"),
    name: 'Charlotte',
    age: 26,
    city: 'Denver'
  }
]
```