UNIT -I

Introducing MongoDB

Introducing MongoDB: History, MongoDB Design Philosophy, Speed, Scalability, and Agility, Non-Relational Approach, JSON-Based Document Store, Performance vs. Features, Running the Database Anywhere, SQL Comparison

PYQ : (Previous Year Mumbai University Question)

Nov – 18

Apr – 19

1. What is MongoDB Philosophy? Explain
2. Write a short note on Non-Relational Approach

Nov – 19

1. Briefly explain how MongoDB Is different from SQL

Nov – 22

Dec – 23

1. Explain Design Decisions considered for MongoDB

Nov – 24

1. Explain MongoDB Design Philosophy in detail

Introducing MongoDB :

"MongoDB is one of the leading NoSQL document store databases. Itenables organizations to handle and gain meaningful insights from Big Data."

Question 1 : Explain "MongoDB is one of the leading NoSQL document store databases. Itenables organizations to handle and gain meaningful insights from Big Data."

MongoDB is a popular NoSQL database that stores data in a flexible, document-like format. Unlike traditional databases that use tables and rows, MongoDB uses collections of documents, which are similar to JSON objects. This makes it easier to work with large amounts of varied and complex data, often referred to as Big Data.

Example:

Imagine you have a database for a library. In a traditional database, you would have tables for books, authors, and publishers. In MongoDB, you can store all related information in a single document.

Here's how a book might be stored in MongoDB:

json

```
{
  "title": "The Great Gatsby",
  "author": "F. Scott Fitzgerald",
  "published_year": 1925,
  "genres": ["Fiction", "Classic"],
  "available": true
}
```

This single document contains all the necessary details about the book, making it easier to manage and retrieve data quickly.

Benefits:

Flexibility: You can add, remove, or update fields in documents without affecting the entire database structure.

Scalability: MongoDB can handle large volumes of data across many servers, making it ideal for Big Data applications.

Performance: It provides fast data access and can perform complex queries efficiently.

History

History In the later part of 2007, Dwight Merriman, Eliot Horowitz, and their team decided to develop an online service. The intent of the service was to provide a platform for developing, hosting, and auto-scaling web applications, much in line with products such as the Google App Engine or Microsoft Azure. Soon they realized that no open source database platform suited the requirements of the service. “We felt like a lot of existing databases didn’t really have the ‘cloud computing’ principles you want them to have: elasticity, scalability, and … easy administration, but also ease of use for developers and operators,” Merriman said. “[MySQL] doesn’t have all those properties.” 1 So they decided to build a database that would not comply with the RDBMS model. A year later, the database for the service was ready to use. The service itself was never released but the team decided in 2009 to open source the database as MongoDB. In March of 2010,the release of MongoDB 1.4.0 was considered production-ready. The latest production release is 3.0and it was released in March 2015.MongoDB was built under the sponsorship of 10gen, a New York–based startup.

MongoDB Design Philosophy :

Question 1 : What is MongoDB Philosophy? Explain.
Question 2 : Explain MongoDB Design Philosophy in detail
Question 3 : Explain Design Decisions considered for MongoDB

1. Speed, Scalability, and Agility:

Speed: MongoDB is designed to provide fast access to data. It can handle large volumes of data and perform complex queries quickly.

Scalability: MongoDB can easily grow with your data. It can distribute data across multiple servers, ensuring that the database can handle increasing loads without slowing down.

Agility: (Quickness) MongoDB's flexible schema allows you to make changes to your data structure without affecting the entire database. This makes it easier to adapt to changing requirements.

Example: If an e-commerce website adds a new feature like customer reviews, MongoDB allows adding this new data without redesigning the entire database.

Question 1 : Write a short note on Non-Relational Approach

2. Non-Relational Approach:

MongoDB is a NoSQL database, which means it does not use tables and rows like traditional relational databases. Instead, it uses collections of documents. This non-relational approach allows for more flexible and dynamic data models.

Example: Instead of having separate tables for users, orders, and products, MongoDB can store all related information in a single document, reducing complexity and improving performance.

3. JSON-Based Document Store:

MongoDB stores data in a format similar to JSON (JavaScript Object Notation). Each document in MongoDB is a key-value pair, making it easy to read and write data.

Example: A user profile in MongoDB might look like this:

json

```
{
  "username": "john_doe",
  "email": "john@example.com",
  "orders": [
    {"order_id": 1, "product": "Laptop", "quantity": 1},
    {"order_id": 2, "product": "Mouse", "quantity": 2}
  ]
}
```

4. Performance vs. Features:

MongoDB balances performance and features. It is designed to be fast and efficient, but also provides rich features like indexing, aggregation, and replication.

Example: MongoDB's indexing improves query performance, while its aggregation framework allows for complex data processing and analysis directly within the database.

5. Running the Database Anywhere:

MongoDB can run on various platforms, including on-premises servers, cloud services, and hybrid environments. This flexibility allows organizations to deploy MongoDB in a way that best fits their needs.

Example: A company might run MongoDB on their own servers for sensitive data, while using a cloud service for less critical data, ensuring both security and cost efficiency.

Summary

- **Speed, Scalability, and Agility:** Fast access, easy to scale, and flexible schema.

- **Non-Relational Approach:** Uses documents instead of tables, allowing for more flexible data models.

- **JSON-Based Document Store:** Stores data in an easy-to-read JSON-like format.

- **Performance vs. Features:** Balances fast performance with rich features.

- **Running the Database Anywhere:** Can be deployed on-premises, in the cloud, or in hybrid environments.

Example Recap: MongoDB allows an e-commerce website to store all user data, orders, and products in flexible, easy-to-manage documents, enabling fast access and easy scaling as the business grows.

SQL Comparison:

Question 1: Briefly explain how MongoDB Is different from SQL

The following are the ways in which MongoDB is different from SQL.

SQL vs. MongoDB Comparison

1. **Data Model:**

   - **SQL:** Uses a table-based structure (rows and columns).

   - **MongoDB:** Uses a document-based structure (JSON-like documents).

Example:

- SQL: A table for books with columns like `title`, `author`, `published_year`.

- MongoDB: A document for each book with fields like `"title": "The Great Gatsby", "author": "F. Scott Fitzgerald", "published_year": 1925`.

2. **Schema:**

- **SQL:** Schema is fixed and defined by tables. Each row must follow the same structure.

- **MongoDB:** Schema is flexible and dynamic. Each document can have a different structure.

Example:

- SQL: All rows in a table must have the same columns.

- MongoDB: One document can have `"publisher": "Scribner"` while another might not have a publisher field.

3. **Query Language:**

- **SQL:** Uses Structured Query Language (SQL) for querying data.

- **MongoDB:** Uses MongoDB Query Language (MQL), which is more like JSON.

Example:

- SQL: `SELECT * FROM books WHERE author = 'F. Scott Fitzgerald';`

- MongoDB: `db.books.find({ "author": "F. Scott Fitzgerald" });`

4. **Transactions:**

- **SQL:** Supports multi-row transactions, ensuring atomicity.

- **MongoDB:** Supports multi-document transactions since version 4.0, but it's less common.

Example:

- SQL: Updating multiple rows in a transaction.

- MongoDB: Updating multiple documents in a transaction.

5. **Normalization:**

- **SQL:** Data is normalized to reduce redundancy (using multiple related tables).

- **MongoDB:** Data is often denormalized for performance (embedding related data in documents).

Example:

- SQL: Separate tables for authors and books.

- MongoDB: Author information embedded within the book document.

6. **Joins:**

- **SQL:** Supports joins to combine data from multiple tables.

- **MongoDB:** Does not support joins traditionally, but uses embedded documents or `$lookup` for similar functionality.

Example:

- SQL: `SELECT books.title, authors.name FROM books JOIN authors ON books.author_id = authors.id;`

- MongoDB: Embedding author details directly in the book document.

7. **Scalability:**

- **SQL:** Scales vertically by adding more resources to a single server.

- **MongoDB:** Scales horizontally by adding more servers (sharding).

Example:

- SQL: Upgrading server hardware.

- MongoDB: Adding more servers to distribute the load.

8. **Performance:**

- **SQL:** Can be slower with complex queries involving multiple joins.

- **MongoDB:** Generally faster for read and write operations on large datasets due to denormalization.

Example:

- SQL: Slow performance with multiple joins.

- MongoDB: Faster retrieval of documents with embedded data.

9. **Data Relationships:**

- **SQL:** Well-suited for complex relationships between entities.

- **MongoDB:** Better for hierarchical data structures.

Example:

- SQL: Relating customers, orders, and products with foreign keys.

- MongoDB: Storing a blog post with comments embedded within it.

10. **Community and Ecosystem:**

- **SQL:** Long-established with a vast ecosystem of tools and support.

- **MongoDB:** Newer but rapidly growing with a strong community and modern tooling.

Example:

- SQL: Mature databases like MySQL, PostgreSQL.

- MongoDB: Modern database with features for contemporary applications.

==========