

NoSQL Using MongoDB

An Introduction to NoSQL Databases
and MongoDB

What is NoSQL?

- NoSQL = Not Only SQL
- Designed for semi-structured and unstructured data
- Enables horizontal scalability and high availability
- Types: Document (e.g., MongoDB), Key-Value (e.g., Redis), Column-family (e.g., Cassandra), Graph (e.g., Neo4j)

Purpose and Benefits of NoSQL

- Handle high volumes of diverse and dynamic data
- Provide schema flexibility for rapid development
- Support distributed architectures and scaling
- Enable real-time analytics and low-latency responses

NoSQL vs. Relational DBMS

- Schema: Flexible (NoSQL) vs. Fixed (SQL)
- Data Model: JSON-like documents vs. Tables and rows
- Scaling: Horizontal (NoSQL) vs. Vertical (SQL)
- Transactions: Limited in NoSQL, ACID in SQL
- Use Cases: Big Data, IoT (NoSQL) vs. Structured apps (SQL)

Scaling: Horizontal (NoSQL) vs. Vertical (SQL)

- ◆ Horizontal Scaling (NoSQL):
 - Add more servers (nodes) to distribute data and load
 - Enables high availability and fault tolerance
 - Common in cloud-based and distributed systems
- ◆ Vertical Scaling (SQL):
 - Increase capacity of a single server (CPU, RAM, storage)
 - Simpler but can become costly and has physical limits
 - Often involves downtime during scaling

Transactions: Limited in NoSQL, ACID in SQL

- ◆ ACID Transactions (SQL):
 - Atomicity: All steps succeed or fail together
 - Consistency: Keeps data in a valid state
 - Isolation: Transactions don't interfere with each other
 - Durability: Once committed, changes persist
- ◆ Limited Transactions (NoSQL):
 - Not all NoSQL DBs support full ACID transactions
 - Prioritizes performance and availability over strict consistency
 - MongoDB supports multi-document transactions (since v4.0) but with trade-offs

Introduction to MongoDB

- Document-based NoSQL database
- Stores data in BSON (Binary JSON) format
- Supports replication, sharding, and indexing
- Open-source and cross-platform

Core Concepts in MongoDB

- **Database:** Container for collections (eg school, ecommerce, hospital)
- **Collection:** Similar to SQL table
 - Collections do not enforce schemas, allowing different document structures in the same collection. (eg students, products, patients)
- **Document:** JSON-like data structure
 - A document is a single record stored in BSON (Binary JSON) format. It's analogous to a **row** in a relational table but more flexible. Contains key-value pairs and can nest arrays or subdocuments.
- **Field:** Key-value pair inside documents

In Summary

MongoDB Server

└ Database

└ Collection

└ Document

- A MongoDB database contains multiple collections
- Each collection contains multiple documents
- A document is the actual unit of data

Example MongoDB (Document)

```
{
  "_id": ObjectId("123"),
  name: "Alice",
  age: 25,
  grades: [85, 90, 92],
  address: {
    street: "123 Maple St",
    city: "Springfield"
  }
}
```

Example MongoDB (Database & Collection)

You create or switch to a database with:

```
use school;
```

Creating a collection (optional—MongoDB auto-creates it on insert):

```
db.createCollection("students");
```

Insert a document into the students collection:

```
db.students.insertOne({  
  "_id": ObjectId("123"),  
  name: "Alice",  
  age: 25,  
  grades: [85, 90, 92],  
  address: {  
    street: "123 Maple St",  
    city: "Springfield"  
  }  
});
```

MongoDB CRUD Operations

- Create: `db.users.insertOne({...})`
- Read: `db.users.find({...})`
- Update: `db.users.updateOne({...})`
- Delete: `db.users.deleteOne({...})`

Use Cases for MongoDB

- Content Management Systems
- Real-time analytics and dashboards
- IoT and sensor data processing
- Mobile and web app backends
- Catalog and inventory management

Conclusion

- NoSQL databases like MongoDB provide flexibility and scalability
- Ideal for unstructured data, high availability, and modern app needs
- Understand trade-offs: consistency, schema, and security models

References

- <https://www.mongodb.com/>
- <https://university.mongodb.com/>
- <https://en.wikipedia.org/wiki/NoSQL>