DATABASE SYSTEMS

FACULTY OF AI & MMG

NoSQL / MongoDB

Content

- Introduction to NoSQL Databases
- Types of NoSQL Databases
- Overview of MongoDB
- CRUD Operations in MongoDB

1. Introduction to NoSQL Databases

NoSQL (Not Only SQL) databases are non-relational data storage systems designed for:

- **Scalability:** Handle massive data volumes across distributed systems.
- **Flexibility:** Schema-less models (unlike rigid SQL tables).
- **Performance:** Optimized for high-speed read/write operations.
- **Variety of Data Types:** Supports JSON, key-value, graphs, etc.

Why Use NoSQL?

- When data is unstructured or semi-structured (e.g., social media posts, sensor data).
- To avoid complex joins (common in relational databases).

Types of NoSQL Databases

NoSQL databases are categorized into four main types:

A. Document Stores

- **Structure**: Store data in **JSON/BSON** documents.
- **Examples**: MongoDB, CouchDB.
- **Use Case**: CMS, user profiles, catalogs.

B. Key-Value Stores

- **Structure**: Simple **key** → **value** pairs.
- **Examples**: Redis, DynamoDB.
- **Use Case**: Caching, session management.

C. Column-Family Stores

- **Structure**: Data stored in **columns** (not rows).
- **Examples**: Cassandra, HBase.
- **Use Case**: Time-series data, big data analytics.

D. Graph Databases

- **Structure**: Nodes + edges to represent relationships.
- **Examples**: Neo4j, ArangoDB.
- **Use Case**: Social networks, fraud detection.

Overview of MongoDB

- MongoDB is the most popular **document-oriented NoSQL database**:
  - **Data Format**: BSON (Binary JSON).
  - **Scalability**: Auto-sharding for distributed data.
  - **Query Language**: JavaScript-like syntax.

- **Key Features**
  - **Indexing**: Speeds up queries.
  - **Aggregation**: For complex analytics.
  - **Replication**: High availability via replica sets.

CRUD Operations in MongoDB

- A. Create (insert): Add new documents to a collection.

```
db.users.insertOne({
  name: "Alice",
  age: 25,
  email: "alice@example.com"
});

// Insert multiple
db.users.insertMany([
  { name: "Bob", age: 30 },
  { name: "Charlie", age: 22 }
]);
```

- B. Read (find): Query documents.

```
// Find all
db.users.find();

// Find with filter
db.users.find({ age: { $gt: 25 } });

// Projection (select fields)
db.users.find({}, { name: 1, email: 1 });
```

CRUD Operations in MongoDB

- C. Update (update): Modify existing documents.

```
// Update one
db.users.updateOne(
  { name: "Alice" },
  { $set: { age: 26 } }
);

// Update many
db.users.updateMany(
  { age: { $lt: 30 } },
  { $inc: { age: 1 } } // Increment age
);
```

- D. Delete (remove): Remove documents.

```
// Delete one
db.users.deleteOne({ name: "Bob" });

// Delete many
db.users.deleteMany({ age: { $lt: 25 } });
```

When to Choose MongoDB?

Scenario	Why MongoDB?	Example
Unstructured/Semi-Structured Data	Handles JSON-like documents with varying fields.	User profiles, IoT sensor data
Rapid Prototyping	No fixed schema → Add/remove fields without migrations.	Startups, MVPs
High Write Throughput	Optimized for fast inserts (e.g., logs, real-time analytics).	Clickstream data, social media posts
Scalability	Horizontal scaling via **sharding** (distributes data across servers).	Big Data applications
Complex Hierarchical Data	Nested documents (e.g., orders with items arrays).	E-commerce catalogs
Geospatial Data	Built-in geospatial indexing and queries.	Location-based apps (Uber, maps)

When to Avoid MongoDB

Scenario	Why Not MongoDB?	Better Alternative
Complex Transactions	Limited multi-document ACID support (though newer versions improve this).	PostgreSQL, MySQL
Strict Schema Requirements	Schema-less design can lead to inconsistencies if not managed properly.	SQL databases
Heavy Joins	No native joins (must use $lookup, which is less efficient than SQL joins).	Relational databases

MongoDB vs. SQL Comparison

Factor	MongoDB (NoSQL)	SQL (e.g., MySQL, PostgreSQL)
Data Model	Document-based (JSON/BSON)	Table-based (rows/columns)
Schema	Dynamic (schema-less)	Fixed (requires migrations)
Scaling	Horizontal (sharding)	Vertical (hardware upgrades)
Transactions	Multi-document (limited)	ACID-compliant
Joins	Manual ($lookup)	Native joins (INNER JOIN, etc.)
Best For	Unstructured data, fast iterations	Structured data, complex queries