### **Summary: Introduction to NoSQL Databases**

**NoSQL databases** emerged to address the need for scalability, flexibility, and efficiency in managing large volumes of unstructured or semi-structured data. They are schema-flexible, making them ideal for applications requiring fast access to varied data types.

#### **Types of NoSQL Databases and CAP Theorem Classification**

1. **Key-Value Stores (e.g., Redis, Riak):**
   * **Data Model:** Simple key-value pairs, similar to a hash table.
   * **Use Cases:** Caching, session management, fast data retrieval.
   * **CAP Classification:** Generally AP (Availability and Partition Tolerance).
2. **Wide Column Stores (e.g., Cassandra, HBase):**
   * **Data Model:** Data stored in columns grouped by row keys.
   * **Use Cases:** High write/read speeds, time-series data, logs.
   * **CAP Classification:** Often AP, ensuring data availability with eventual consistency.
3. **Document Databases (e.g., MongoDB):**
   * **Data Model:** JSON-like documents with nested structures.
   * **Use Cases:** Flexible schema, web applications, content management.
   * **CAP Classification:** Typically CP (Consistency and Partition Tolerance).
4. **Graph Databases (e.g., Neo4j, HyperGraphDB):**
   * **Data Model:** Nodes and edges representing data relationships.
   * **Use Cases:** Complex relationships, social networks, recommendation engines.
   * **CAP Classification:** Typically CP, prioritizing data consistency.

#### **CAP Theorem and MongoDB as a CP Database**

* **CAP Theorem:** A distributed database can only guarantee two out of three properties: Consistency, Availability, and Partition Tolerance.
  + **Consistency:** All clients see the same data simultaneously.
  + **Availability:** Every request receives a response.
  + **Partition Tolerance:** The system functions despite communication failures.
* **MongoDB as CP:**
  + **Consistency:** Achieved by directing all write operations to a single primary node.
  + **Partition Tolerance:** Maintained through replica sets, which may temporarily sacrifice availability to preserve consistency.

#### **Comparisons: Relational vs. Non-Relational Databases**

| **Feature** | **RDBMS (ACID)** | **NoSQL (BASE)** |
| --- | --- | --- |
| Data Structure | Rows and columns (fixed schema) | Document, key-value, column, graph |
| Consistency | Strong consistency | Eventual consistency |
| Scalability | Vertical | Horizontal |
| Schema Evolution | Rigid, predefined schema | Flexible schema |
| Transaction Support | Complex multi-table transactions | Simple transactions |

#### **Benefits of NoSQL Databases in Web Applications**

* **Flexibility in Schema Design:** Handles diverse data structures without a strict schema.
* **High Scalability:** Scales horizontally to manage increasing data volumes and traffic.
* **Reduced Need for ETL:** Stores data in native formats, reducing data transformation needs.
* **Efficient Handling of Unstructured Data:** Supports indexing and querying within unstructured text data.
* **High Performance and Faster Data Retrieval:** Suitable for real-time applications and high-traffic environments.
* **Cloud and Distributed System Compatibility:** Designed for distributed environments, with built-in features for data sharding and replication.
* **Enhanced Developer Productivity:** Simplified query model allows focus on application functionality.
* **Vendor and Open-Source Options:** Variety of options to match specific needs.

#### **Practical Use Case: Building MVC Web Applications with NoSQL**

Students will develop MVC-based web applications using MongoDB, learning to implement CRUD operations, optimize data structures, and leverage MongoDB’s scaling and replication features.

#### **Summary**

NoSQL databases, including MongoDB, offer scalability, flexibility, and efficient data management. By following the BASE model, they ensure high availability and adaptability for modern applications. MongoDB’s CP classification ensures consistency across distributed nodes, supporting diverse web applications requiring real-time data handling.