BUDDHA INSTITUTE OF TECHNOLOGY
Gida Gorakhpur

Department- Computer Science and Allied
(CSE AND DS)

Program & Semester- B.Tech 3rd Year (6th Semester)

Course and Code- BIG Data Analytics BCDS 061 & BCS 061

Course Outcome

CO No.	Course Outcome	Bloom's Knowledge Level (KL)
CO 1	Demonstrate knowledge of Big Data Analytics concepts and its applications in business.	K1, K2
CO 2	Demonstrate functions and components of Map Reduce Framework and HDFS.	K1, K2
CO 3	Discuss Data Management concepts in NoSQL environment.	K6
CO 4	Explain process of developing Map Reduce based distributed processing applications.	K2, K5
CO 5	Explain process of developing applications using HBASE, Hive, Pig etc.	K2, K5

UNIT-4
YARN & NOSQL

1. Define NoSQL databases. What are the key characteristics and benefits of NoSQL databases compared to traditional relational databases? (2021–22, 2023–24)

NoSQL databases are non-relational databases designed to handle large volumes of structured, semi-structured, and unstructured data. Unlike traditional relational databases (RDBMS), NoSQL databases do not use fixed table schemas and are built to support distributed data storage and high scalability.

NoSQL databases emerged to address the limitations of relational databases in handling Big Data. They are widely used in applications requiring high performance, flexibility, and horizontal scalability.

Key characteristics of NoSQL databases include:

• Schema-less design – Data can be stored without predefined structure

• Horizontal scalability – Easily scalable by adding more nodes

• High availability – Supports distributed architecture

• Flexible data models – Supports key-value, document, column, and graph models

• High performance – Optimized for fast read/write operations

Types of NoSQL databases:

• Key-value stores (e.g., Redis)

• Document databases (e.g., MongoDB)

• Column-family databases (e.g., HBase)

• Graph databases (e.g., Neo4j)

Comparison with relational databases:

• RDBMS uses fixed schema, while NoSQL is schema-less

• RDBMS ensures ACID properties, while NoSQL often follows BASE model

• RDBMS scales vertically, NoSQL scales horizontally

• RDBMS is suitable for structured data, NoSQL supports all data types

Benefits of NoSQL databases:

• Scalability – Can handle large-scale distributed data

• Flexibility – Suitable for dynamic and evolving data

• High performance – Faster processing for Big Data applications

• Fault tolerance – Data is distributed across multiple nodes

• Cost-effective – Uses commodity hardware

Another important advantage is the ability to handle unstructured data such as social media content, images, and logs. This makes NoSQL highly suitable for modern applications like e-commerce, IoT, and real-time analytics.

However, NoSQL databases may lack strong consistency compared to relational databases. They also require careful design for data modeling.

In conclusion, NoSQL databases provide a flexible and scalable alternative to relational databases. Their ability to handle large and diverse datasets makes them essential for Big Data environments.

2. Explain CRUD operations in MongoDB with suitable examples. (2022–23, 2023–24)

MongoDB is a document-oriented NoSQL database that stores data in the form of JSON-like documents. CRUD operations represent the basic operations performed on data: Create, Read, Update, and Delete.

CRUD operations are fundamental for managing data in MongoDB collections.

Create operation is used to insert data into a collection. Documents are stored in BSON format.

• Example:

db.students.insertOne({name: "Anurag", age: 22})

This command inserts a new document into the students collection.

Read operation is used to retrieve data from the database.

• Example:

db.students.find({name: "Anurag"})

This command retrieves documents matching the condition.

Update operation modifies existing data.

• Example:

db.students.updateOne({name: "Anurag"}, {$set: {age: 23}})

This updates the age field of the specified document.

Delete operation removes data from the collection.

• Example:

db.students.deleteOne({name: "Anurag"})

This deletes the document matching the condition.

Key features of CRUD operations in MongoDB include:

• Flexible schema allows dynamic data insertion

• Supports complex queries using JSON-like syntax

• Efficient indexing improves query performance

• Supports bulk operations

MongoDB also supports advanced operations such as:

• findOne() for retrieving a single document

• updateMany() for updating multiple documents

• deleteMany() for removing multiple documents

Advantages of MongoDB CRUD operations:

• Easy to use and flexible

• Suitable for large-scale applications

• High performance due to indexing

• Supports nested data structures

MongoDB operations are atomic at the document level, ensuring consistency within a single document.

In conclusion, CRUD operations in MongoDB provide a simple yet powerful way to manage data. Their flexibility and performance make MongoDB suitable for modern Big Data applications.

3. Discuss the architecture and components of YARN. (2024–25)

YARN (Yet Another Resource Negotiator) is a resource management layer in Hadoop that manages cluster resources and schedules applications. It separates resource management from data processing, improving scalability and efficiency.

YARN architecture consists of the following components:

• ResourceManager – global authority that manages cluster resources

• NodeManager – runs on each node and manages resources locally

• ApplicationMaster – manages execution of an application

• Container – unit of resource allocation (CPU, memory)

The ResourceManager is the master component responsible for allocating resources to applications. It has two main parts:

• Scheduler – allocates resources based on policies

• ApplicationManager – manages application lifecycle

NodeManager runs on each node and is responsible for:

• Monitoring resource usage

• Managing containers

• Reporting status to ResourceManager

ApplicationMaster is launched for each application and performs the following tasks:

• Negotiates resources with ResourceManager

• Coordinates execution of tasks

• Monitors progress

Working of YARN:

• Client submits application to ResourceManager

• ResourceManager allocates resources

• ApplicationMaster is launched

• ApplicationMaster requests containers

• Tasks are executed in containers

• Results are returned to client

Key features of YARN include:

• Scalability – supports large clusters
• Efficient resource utilization
• Supports multiple processing frameworks
• Improves cluster performance

YARN supports different schedulers such as:

• FIFO Scheduler
• Fair Scheduler
• Capacity Scheduler

Advantages of YARN:

• Better resource management
• Separation of concerns
• Supports real-time and batch processing
• Improved system utilization

YARN allows multiple applications to run simultaneously on the same cluster, improving efficiency.

In conclusion, YARN is a powerful resource management system that enhances Hadoop's performance by efficiently allocating resources and managing applications.

4. Explain the role of YARN in the Hadoop ecosystem. (2023–24, 2024–25)

YARN (Yet Another Resource Negotiator) is a core component of the Hadoop ecosystem responsible for resource management and job scheduling. It was introduced in Hadoop 2.0 to overcome the limitations of the earlier MapReduce framework, where resource management and processing were tightly coupled. YARN separates these functions, making the system more scalable and efficient.

The primary role of YARN is to manage cluster resources and allocate them to various applications running on the Hadoop cluster. It ensures that resources such as CPU, memory, and storage are used efficiently and fairly among different users and applications.

Key roles of YARN include:

• Resource management – Allocates cluster resources dynamically based on application requirements
• Job scheduling – Determines the order in which jobs are executed
• Multi-tenancy – Allows multiple applications to run simultaneously on the same cluster
• Scalability – Supports large-scale clusters with thousands of nodes
• Fault tolerance – Handles failures and restarts tasks automatically

YARN operates through its main components:

• ResourceManager – central authority managing resources
• NodeManager – manages resources at each node
• ApplicationMaster – handles execution of a specific application

• Containers – units of resource allocation

Working of YARN:

• Client submits application to ResourceManager

• ResourceManager allocates resources for ApplicationMaster

• ApplicationMaster negotiates resources for tasks

• Tasks are executed in containers on NodeManagers

• Results are returned to the client

YARN improves Hadoop by allowing different processing frameworks such as MapReduce, Spark, and Tez to run on the same cluster. This flexibility makes Hadoop more versatile.

Advantages of YARN:

• Efficient utilization of cluster resources

• Supports diverse workloads (batch, interactive, real-time)

• Reduces bottlenecks in resource allocation

• Enhances system performance

YARN also supports different scheduling policies like FIFO, Fair Scheduler, and Capacity Scheduler, which ensure proper distribution of resources among users.

In conclusion, YARN plays a crucial role in the Hadoop ecosystem by managing resources, scheduling jobs, and enabling multiple applications to run efficiently, thereby improving scalability and performance.

5. Explain schedulers in Hadoop such as Fair Scheduler and Capacity Scheduler. (2023–24)

Schedulers in Hadoop are responsible for allocating resources among various applications running in a cluster. They are part of the YARN ResourceManager and ensure efficient and fair utilization of cluster resources.

Schedulers determine how resources are distributed among competing jobs and users. Two commonly used schedulers in Hadoop are Fair Scheduler and Capacity Scheduler.

The Fair Scheduler allocates resources such that all applications get an equal share over time. If one application uses fewer resources, others can utilize the remaining resources.

Features of Fair Scheduler:

• Provides equal share of resources to all applications

• Dynamically reallocates resources based on demand

• Supports priority and preemption

• Suitable for environments with multiple users

Working of Fair Scheduler:

• Resources are divided equally among running jobs

• If a job finishes, resources are redistributed

• Idle resources are assigned to other applications

The Capacity Scheduler is designed for large organizations where resources are divided into queues. Each queue is assigned a fixed capacity, and users submit jobs to these queues.

Features of Capacity Scheduler:

• Resources are divided into multiple queues

• Each queue has a guaranteed capacity

• Supports hierarchical queues

• Allows sharing of unused resources

Working of Capacity Scheduler:

• Resources are allocated based on queue capacity

• If a queue is idle, others can use its resources

• Ensures minimum resource guarantee for each queue

Comparison of schedulers:

• Fair Scheduler focuses on equal distribution

• Capacity Scheduler focuses on guaranteed resource allocation

• Fair Scheduler is flexible, Capacity Scheduler is structured

Advantages of schedulers:

• Efficient resource utilization

• Improved performance of cluster

• Fair distribution among users

• Support for multi-user environments

Schedulers play a crucial role in optimizing cluster performance by balancing resource usage and ensuring fairness. In conclusion, Hadoop schedulers like Fair Scheduler and Capacity Scheduler help in effective resource allocation and job scheduling, improving overall system efficiency.

6. Explain Resilient Distributed Datasets (RDDs) in Apache Spark and their role in fault tolerance. (2023–24)

Resilient Distributed Datasets (RDDs) are the fundamental data structure in Apache Spark. They represent a distributed collection of data that is processed in parallel across multiple nodes in a cluster. RDDs are designed to provide fault tolerance, scalability, and high performance.

RDDs are immutable, meaning once created, they cannot be modified. Instead, transformations create new RDDs from existing ones. This immutability ensures consistency and reliability.

Key characteristics of RDDs include:

• Distributed – Data is partitioned across multiple nodes

• Immutable – Cannot be changed once created

• Fault tolerant – Automatically recovers from failures

• Lazy evaluation – Operations are executed only when needed

• In-memory processing – Improves speed compared to disk-based systems

RDDs support two types of operations:

• Transformations – create new RDDs (e.g., map, filter)

• Actions – return results (e.g., collect, count)

Fault tolerance in RDDs is achieved through lineage. Lineage is a record of transformations used to create an RDD. If a partition is lost due to node failure, Spark can recompute it using lineage instead of replicating data.

Fault tolerance mechanism:

• RDD tracks sequence of transformations

• If data is lost, it is recomputed

• No need to store multiple replicas

• Efficient recovery mechanism

Advantages of RDDs:

• High performance due to in-memory computation

• Efficient handling of large datasets

• Fault tolerance without heavy replication

• Supports parallel processing

RDDs also support persistence, where data can be cached in memory for faster access. This improves performance in iterative algorithms.

Compared to MapReduce, RDDs reduce disk I/O and provide faster execution.

In conclusion, RDDs are a powerful abstraction in Spark that enable efficient distributed data processing. Their lineage-based fault tolerance mechanism ensures reliability while maintaining high performance.

7. Explain the components and features of Apache Spark. (2024–25)

Apache Spark is a fast, distributed data processing framework designed for large-scale data analytics. It provides in-memory processing capabilities, making it significantly faster than traditional MapReduce. Spark supports various workloads such as batch processing, real-time processing, machine learning, and graph processing.

The architecture of Spark consists of several core components:

• Driver Program – controls the execution of the application and creates SparkContext

• Cluster Manager – manages resources across the cluster (YARN, Mesos, Standalone)

• Executors – worker nodes that execute tasks and store data

• Tasks – smallest unit of work assigned to executors

Spark ecosystem includes the following components:

• Spark Core – provides basic functionalities such as task scheduling and memory management

• Spark SQL – used for structured data processing using SQL queries

• Spark Streaming – processes real-time data streams

• MLlib – machine learning library

• GraphX – used for graph processing

Key features of Apache Spark include:

• In-memory processing – stores data in memory for faster computation

• Lazy evaluation – operations are executed only when required

• Fault tolerance – achieved through lineage

• Scalability – supports large clusters

• Multi-language support – supports Java, Scala, Python, and R

Working of Spark:

• User submits application to Driver Program

• Driver divides job into stages and tasks

• Tasks are distributed to executors

• Executors process data and return results

Advantages of Spark:

• Faster than MapReduce due to in-memory computation

• Supports iterative algorithms efficiently

• Handles both batch and real-time processing

• Easy integration with Hadoop ecosystem

Spark improves performance by reducing disk I/O and enabling parallel processing. It is widely used in applications such as data analytics, machine learning, and real-time processing.

In conclusion, Apache Spark is a powerful data processing framework with multiple components and features that enable efficient and scalable Big Data processing.

8. Explain the Hadoop ecosystem components and how they work together to enable distributed data processing. (2022–23, 2023–24)

The Hadoop ecosystem consists of various tools and frameworks that work together to store, process, and analyze large datasets in a distributed environment. These components provide different functionalities such as storage, processing, data ingestion, and querying.

The core components of Hadoop ecosystem include:

• HDFS – distributed storage system

• MapReduce – processing framework

• YARN – resource management system

In addition to these, several other tools enhance the capabilities of Hadoop:

• Hive – data warehouse for querying data using SQL-like language

• Pig – high-level scripting language for data processing
• HBase – NoSQL database for real-time data access
• Sqoop – transfers data between Hadoop and relational databases
• Flume – collects and transfers log data into Hadoop
• Oozie – workflow scheduler for managing jobs
• ZooKeeper – coordination service for distributed systems

Working of Hadoop ecosystem:

• Data is collected using tools like Flume and Sqoop
• Data is stored in HDFS
• Processing is done using MapReduce or Spark
• Resource management is handled by YARN
• Data is queried using Hive or Pig
• Results are analyzed and visualized

Key features of Hadoop ecosystem:

• Distributed storage and processing
• Scalability and fault tolerance
• Support for structured and unstructured data
• Integration of multiple tools

Advantages of Hadoop ecosystem:

• Handles large volumes of data efficiently
• Flexible and extensible framework
• Cost-effective due to commodity hardware
• Supports real-time and batch processing

The integration of various components allows Hadoop to perform complex data processing tasks efficiently. Each component plays a specific role and works in coordination with others.

In conclusion, the Hadoop ecosystem provides a complete platform for Big Data processing by combining multiple tools that work together to enable distributed storage, processing, and analysis.

9. Evaluate the performance benefits and limitations of Spark over MapReduce. (2024–25)

Apache Spark and MapReduce are both distributed data processing frameworks, but Spark offers several performance advantages over MapReduce due to its in-memory processing capabilities.

Spark processes data in memory, while MapReduce relies on disk-based processing. This difference significantly impacts performance.

Performance benefits of Spark:

• In-memory computation – reduces disk I/O and speeds up processing

• Faster execution – up to 100 times faster than MapReduce for certain tasks

• Efficient for iterative algorithms – reuses data in memory

• Supports real-time processing – handles streaming data

• Reduced latency – faster data access and processing

Spark also uses Directed Acyclic Graph (DAG) execution, which optimizes task execution and reduces unnecessary operations.

Limitations of Spark compared to MapReduce:

• High memory usage – requires more RAM

• Complex setup – more complex than MapReduce

• Not suitable for very simple tasks – MapReduce may be sufficient

• Resource management challenges – requires proper tuning

Comparison between Spark and MapReduce:

• Spark is faster due to in-memory processing

• MapReduce is more reliable for batch processing

• Spark supports multiple workloads, MapReduce is limited

• Spark reduces execution time significantly

Advantages of Spark:

• Better performance for large datasets

• Supports machine learning and graph processing

• Flexible and easy to use

Advantages of MapReduce:

• Simple and stable

• Suitable for large-scale batch processing

• Requires less memory

Spark is widely used in modern Big Data applications where speed and real-time processing are important. However, MapReduce is still used in scenarios where memory resources are limited.

In conclusion, Spark provides significant performance improvements over MapReduce, especially for iterative and real-time processing. However, its higher resource requirements must be considered when choosing between the two.

10. Discuss the advantages and challenges of deploying Hadoop in a cloud environment. (2024–25)

Deploying Hadoop in a cloud environment has become increasingly popular due to the need for scalable and flexible infrastructure for Big Data processing. Cloud platforms such as AWS, Azure, and Google Cloud provide on-demand resources that enhance Hadoop's capabilities.

Cloud-based Hadoop systems allow organizations to process large datasets without investing in physical

infrastructure. The cloud provides virtual machines, storage, and networking services that can be easily configured.

Advantages of deploying Hadoop in cloud include:

• Scalability – Resources can be scaled up or down based on demand

• Cost efficiency – Pay-as-you-go model reduces capital expenditure

• Flexibility – Easy to deploy and configure clusters

• High availability – Cloud providers ensure uptime and reliability

• Reduced maintenance – No need to manage physical hardware

Another important advantage is elasticity. Organizations can dynamically allocate resources based on workload requirements. This is particularly useful for applications with varying data processing needs.

Cloud platforms also provide integration with various tools such as data analytics, machine learning, and storage services, enhancing the overall ecosystem.

Despite these advantages, there are certain challenges:

• Data security concerns – Sensitive data stored in cloud may be vulnerable

• Network latency – Data transfer between cloud and local systems may cause delays

• Dependency on internet connectivity – Requires stable network access

• Cost management – Improper usage can lead to higher costs

• Limited control – Less control over infrastructure compared to on-premise systems

Another challenge is data transfer cost. Moving large datasets to and from the cloud can be expensive and time-consuming.

Cloud environments also require proper configuration of Hadoop clusters to ensure optimal performance. Misconfiguration can lead to inefficiencies.

In conclusion, deploying Hadoop in the cloud provides scalability, flexibility, and cost benefits. However, challenges such as security and cost management must be addressed for effective implementation.

11. Discuss different types of NoSQL databases and their use cases in real-time applications. (2024–25)

NoSQL databases are designed to handle large volumes of diverse data and provide flexible storage solutions. They are widely used in real-time applications where scalability and performance are critical.

There are four main types of NoSQL databases:

• Key-value stores – store data as key-value pairs

• Document databases – store data in JSON-like documents

• Column-family databases – store data in column-oriented format

• Graph databases – store relationships between data

Key-value stores are simple and efficient. They are used in caching and session management.

Examples and use cases:

• Redis – used for caching and real-time analytics

• DynamoDB – used in e-commerce applications

Document databases store semi-structured data and are flexible.

Use cases include:

• MongoDB – used in content management systems

• CouchDB – used in web applications

Column-family databases are optimized for large-scale data processing.

Use cases include:

• HBase – used for real-time analytics in Hadoop

• Cassandra – used in distributed systems

Graph databases focus on relationships between data.

Use cases include:

• Social networks

• Recommendation systems

• Fraud detection

Advantages of NoSQL databases:

• High scalability

• Flexible schema

• Fast performance

• Suitable for unstructured data

In real-time applications, NoSQL databases are used for:

• Social media analytics

• Online gaming systems

• IoT data processing

• Real-time recommendation engines

NoSQL databases support distributed architecture, allowing data to be processed quickly and efficiently.

In conclusion, different types of NoSQL databases serve various purposes in real-time applications. Their flexibility and scalability make them ideal for modern data-driven systems.

12. Explain how YARN manages resources and scheduling in a distributed Hadoop environment. (2024–25)

YARN is responsible for managing resources and scheduling tasks in a Hadoop cluster. It ensures efficient utilization of resources and supports multiple applications running simultaneously.

YARN separates resource management from data processing, allowing better scalability and performance.

The main components involved in resource management are:

• ResourceManager – manages cluster resources

• NodeManager – monitors resources on each node

• ApplicationMaster – manages application execution

• Containers – units of resource allocation

Resource management process:

• Client submits application to ResourceManager

• ResourceManager allocates resources for ApplicationMaster

• ApplicationMaster requests containers for tasks

• NodeManagers execute tasks in containers

Scheduling in YARN is handled by different schedulers:

• FIFO Scheduler – processes jobs in order

• Fair Scheduler – allocates equal resources

• Capacity Scheduler – allocates resources based on queues

Features of YARN resource management:

• Dynamic allocation of resources

• Support for multiple applications

• Efficient utilization of cluster resources

• Fault tolerance and recovery

YARN improves performance by allowing parallel execution of tasks. It also ensures that resources are not wasted by reallocating unused resources.

Advantages of YARN:

• Better scalability compared to MapReduce

• Supports multiple frameworks like Spark and Tez

• Efficient scheduling and resource allocation

• Improved cluster utilization

YARN also supports monitoring and tracking of applications, providing better control over resource usage.

In conclusion, YARN plays a critical role in managing resources and scheduling tasks in Hadoop. Its efficient resource allocation and scheduling mechanisms ensure optimal performance in distributed environments.

13. Compare and contrast NoSQL databases with relational databases. (2021–22, 2022–23)

NoSQL and relational databases are two major types of database systems used for data storage and management. Relational databases (RDBMS) follow a structured approach using tables, whereas NoSQL databases provide a flexible and scalable approach suitable for Big Data applications.

Relational databases store data in tables with predefined schemas. Each table consists of rows and columns, and relationships between tables are maintained using keys. SQL is used for querying data. NoSQL databases, on the other hand, use flexible data models such as key-value, document, column-family, and graph formats.

Key differences between NoSQL and relational databases include:

• Schema – RDBMS uses fixed schema, NoSQL uses dynamic schema
• Data model – RDBMS uses tables, NoSQL supports multiple data models
• Scalability – RDBMS scales vertically, NoSQL scales horizontally
• Consistency – RDBMS follows ACID properties, NoSQL often follows BASE model
• Performance – NoSQL provides faster performance for large datasets

Relational databases ensure strong consistency and are suitable for applications requiring transactional integrity, such as banking systems. NoSQL databases are designed for distributed systems and are suitable for applications requiring high scalability and flexibility.

Advantages of relational databases:
• Strong consistency and reliability
• Structured data storage
• Mature technology with widespread use

Advantages of NoSQL databases:
• High scalability and flexibility
• Efficient handling of unstructured data
• Better performance for Big Data applications

Limitations of relational databases include difficulty in scaling and handling large volumes of unstructured data. NoSQL databases may lack strong consistency and require careful design.

In conclusion, relational databases are suitable for structured data and transactional systems, while NoSQL databases are ideal for Big Data applications requiring scalability and flexibility.

14. Explain the role of indexing in MongoDB with an example. (2021–22)

Indexing in MongoDB is a technique used to improve the performance of data retrieval operations. It allows the database to quickly locate and access data without scanning the entire collection.

An index is a data structure that stores a small portion of the data along with pointers to the actual documents. MongoDB supports various types of indexes, such as single-field, compound, text, and geospatial indexes.

The role of indexing includes:
• Faster query execution
• Efficient data retrieval
• Reduced search time
• Improved performance for large datasets

Without indexing, MongoDB performs a collection scan, which means it checks every document in the collection. This is inefficient for large datasets.

Example of indexing in MongoDB:
• Create index:

db.students.createIndex({name: 1})

This creates an index on the name field in ascending order.

• Query using index:

db.students.find({name: "Anurag"})

With indexing, MongoDB quickly locates the document without scanning the entire collection.

Types of indexes in MongoDB:

• Single-field index – index on a single field

• Compound index – index on multiple fields

• Text index – used for text search

• Hashed index – used for sharding

Advantages of indexing:

• Improves query performance

• Reduces resource usage

• Enhances scalability

Limitations of indexing:

• Additional storage space required

• Slower write operations due to index updates

Indexes must be carefully designed to balance performance and storage requirements.

In conclusion, indexing plays a crucial role in MongoDB by improving query performance and enabling efficient data retrieval. Proper use of indexes enhances the overall performance of the database system.

15. Does MongoDB support ACID properties? Justify your answer. (2021–22)

MongoDB is a NoSQL database that traditionally follows the BASE model, which prioritizes availability and scalability over strict consistency. However, modern versions of MongoDB provide support for ACID properties at the document level and through multi-document transactions.

ACID properties include:

• Atomicity – ensures that operations are completed fully or not at all

• Consistency – ensures data remains valid after transactions

• Isolation – ensures transactions do not interfere with each other

• Durability – ensures committed data is permanently stored

MongoDB supports atomicity at the document level. This means that all operations on a single document are atomic. If an operation fails, the document remains unchanged.

With the introduction of multi-document transactions in newer versions, MongoDB now supports ACID properties across multiple documents as well.

ACID support in MongoDB includes:

• Atomic operations on single documents

• Multi-document transactions with full ACID support

• Write concern for durability

• Read concern for consistency

Example of transaction in MongoDB:

• Start transaction

• Perform multiple operations

• Commit transaction

If any operation fails, the entire transaction is rolled back.

Advantages of ACID support in MongoDB:

• Ensures data consistency

• Improves reliability of applications

• Supports complex operations

Limitations:

• Performance overhead for transactions

• Increased complexity in implementation

MongoDB balances between ACID and BASE models. It provides strong consistency when needed while maintaining scalability.

In conclusion, MongoDB supports ACID properties, especially in modern versions with multi-document transactions. This makes it suitable for applications requiring both scalability and data consistency.