Topic:

"Relational vs. NoSQL Databases – Impact on Modern Applications".

1. Introduction

Databases play a critical role in modern applications by storing, organizing, and managing data efficiently. Almost every system we use today—such as mobile apps, websites, banking systems, and social media platforms—depends on databases to function properly. With the rapid growth of technology, the amount of data being generated has increased significantly, making database management more important than ever.

There are two major categories of databases used in modern systems: Relational (SQL) databases and NoSQL databases. Relational databases have been used for decades and are known for their structured design and strong data consistency. NoSQL databases emerged later to handle large volumes of unstructured and semi-structured data and to support modern needs such as scalability and high performance.

Different types of databases exist because no single database solution can meet all application requirements. Some applications require strict accuracy and consistency, while others prioritize speed, flexibility, and scalability. Therefore, understanding relational and NoSQL databases is essential for building efficient modern applications.

2. Relational & NoSQL Databases

Relational Databases (SQL)

Relational databases store data in the form of **tables**, which consist of rows and columns. Each table represents an entity, and relationships between tables are established using keys. These databases use Structured Query Language (SQL) to insert, update, delete, and retrieve data.

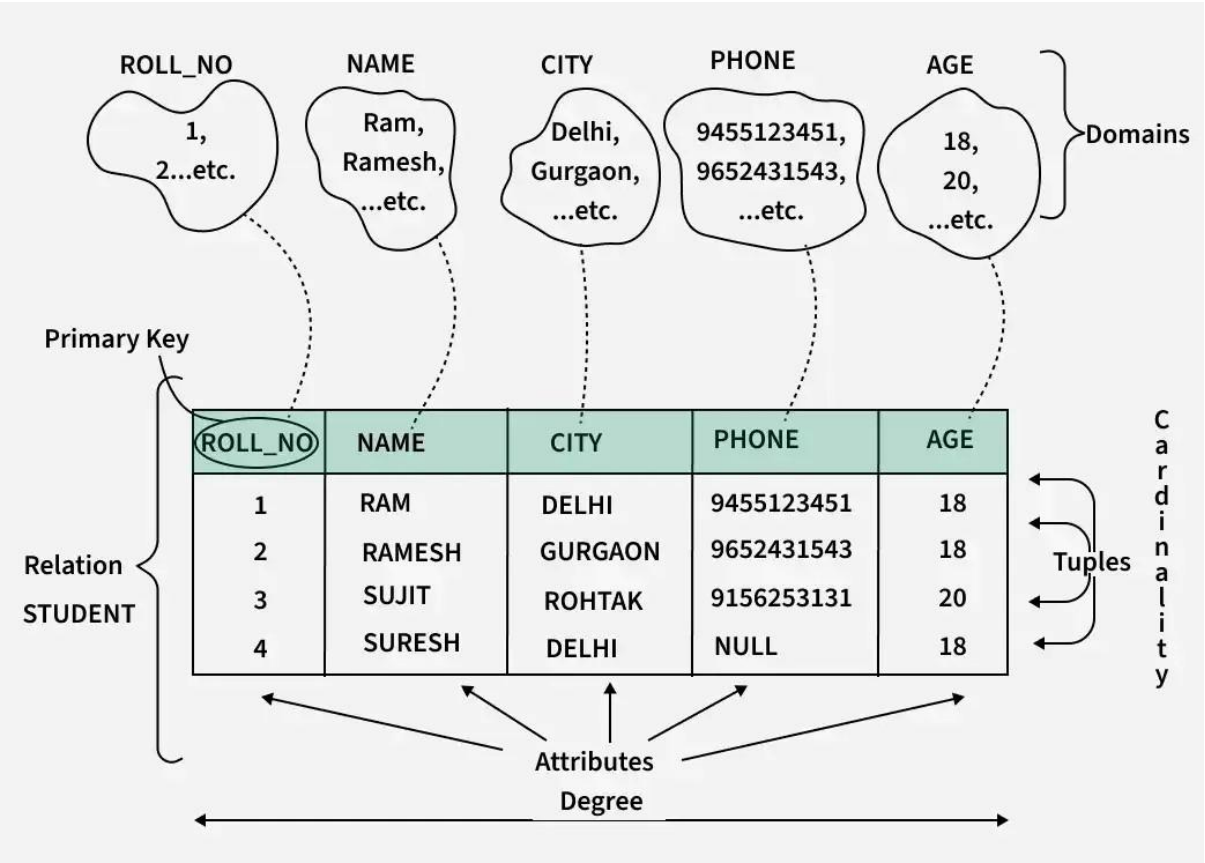
Examples of relational databases include:

- MySQL
- PostgreSQL
- Oracle Database

Characteristics of Relational Databases:

- Fixed schema (predefined structure)
- Data is highly organized
- Strong consistency and accuracy
- Supports ACID properties
- Best suited for structured data

Relational databases are widely used in systems where data accuracy and reliability are extremely important, such as banking and financial applications.



NoSQL Databases

NoSQL databases are designed to store and manage large volumes of data that may not follow a fixed structure. Unlike relational databases, NoSQL databases allow flexible schemas and can easily handle unstructured or semi-structured data.

Types of NoSQL Databases:

- Key-Value Databases (e.g., Redis)
- Document Databases (e.g., MongoDB)
- Column-Family Databases (e.g., Cassandra)
- Graph Databases (e.g., Neo4j)

Characteristics of NoSQL Databases:

- Flexible or schema-less design
- High scalability

- Faster read/write performance
- Suitable for big data and real-time applications

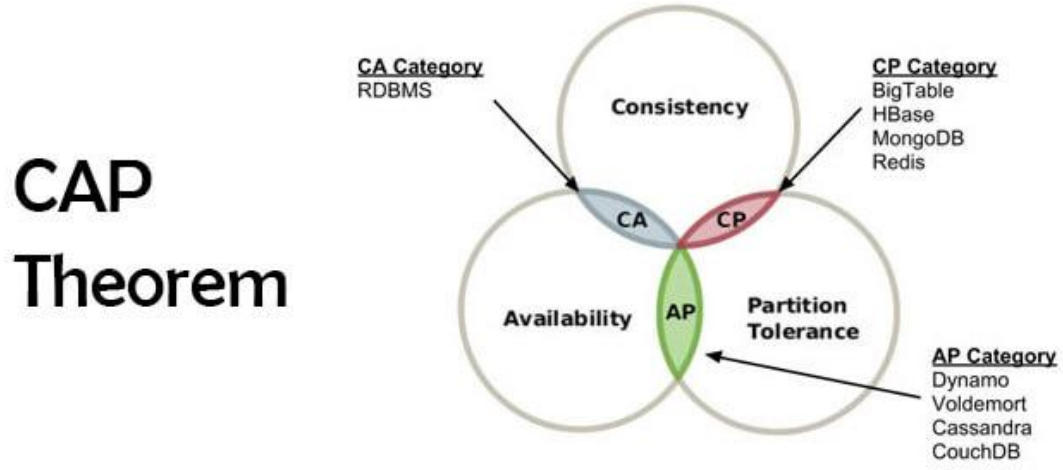
NoSQL databases are commonly used in modern web applications, social networks, and Internet of Things (IoT) systems.

CAP Theorem

The CAP theorem states that a distributed database system can provide only two out of three guarantees at the same time:

- **Consistency (C):** All users see the same data at the same time
- **Availability (A):** The system always responds to user requests
- **Partition Tolerance (P):** The system continues to work even if network failures occur

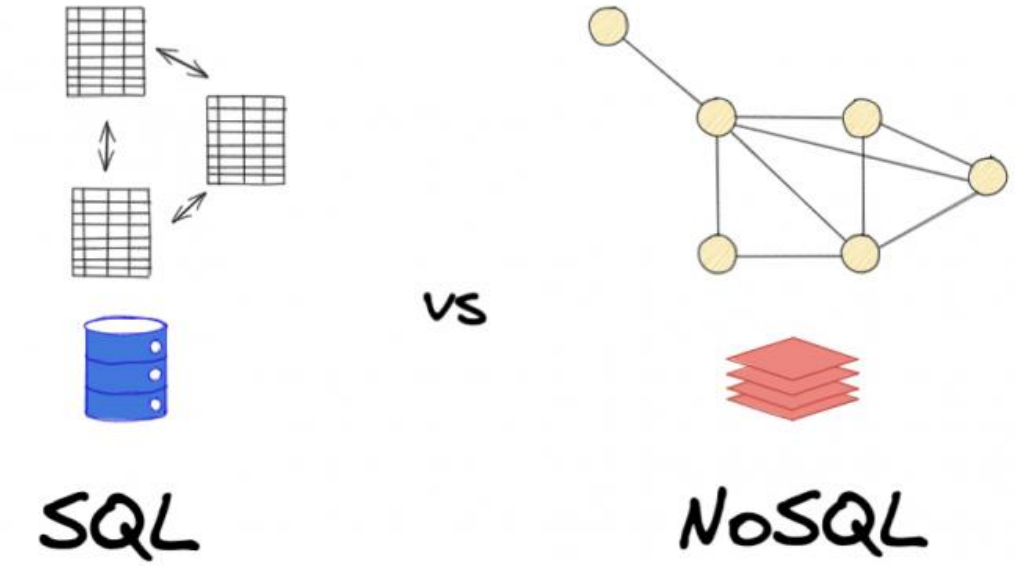
Most NoSQL databases prioritize Availability and Partition Tolerance, which allows them to perform well in distributed environments, even though they may sacrifice strict consistency.



3. Key Differences & Comparisons

SQL vs. NoSQL Comparison

Feature	Relational (SQL)	NoSQL
Data Structure	Tables (rows & columns)	Documents, Key-Value, Graph
Schema	Fixed	Flexible
Scalability	Vertical (scale up)	Horizontal (scale out)
Consistency	Strong	Eventual
Performance	Moderate	High
Best Use	Structured data	Big & unstructured data

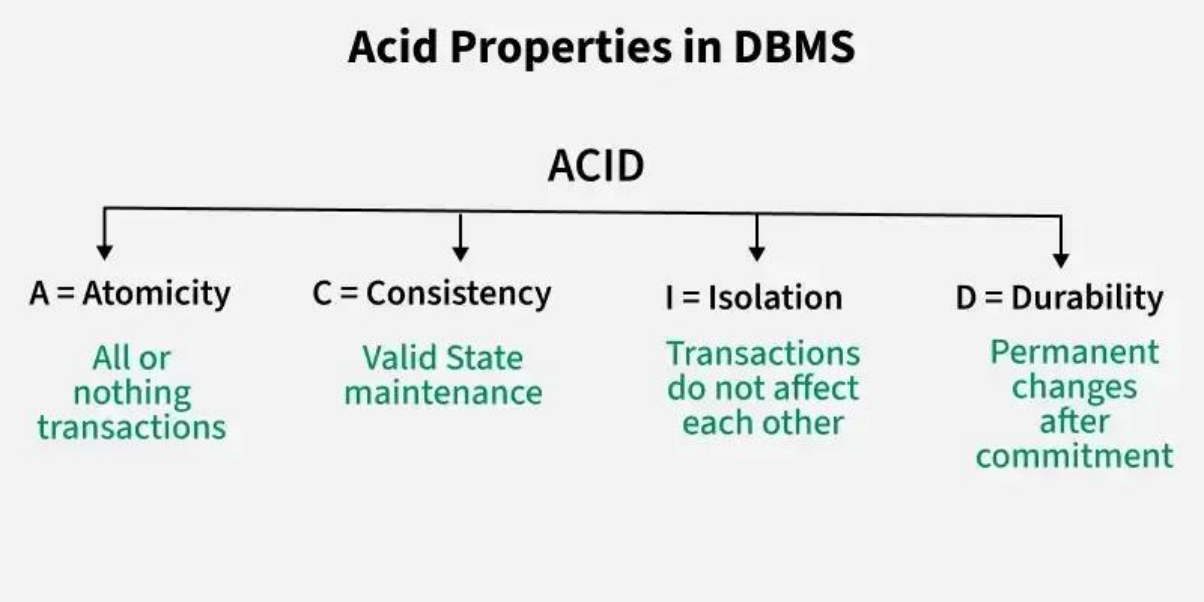


ACID vs. BASE Models

ACID (Used by SQL databases):

- **Atomicity:** Transaction is fully completed or not at all
- **Consistency:** Data remains correct
- **Isolation:** Transactions do not interfere
- **Durability:** Data is permanently saved

ACID ensures strong reliability but may reduce performance.



BASE (Used by NoSQL databases):

- **Basically Available**
- **Soft State**
- **Eventually Consistent**

BASE improves performance and scalability but allows temporary inconsistency.

4. Impact on Modern Applications

Applications Using Relational Databases

- **Banking systems:** secure transactions
- **E-commerce platforms:** order and payment processing
- **ERP systems:** company operations management

These systems require high accuracy and consistency.

Applications Using NoSQL Databases

- **Social media platforms:** user posts and interactions
- **Real-time analytics systems**
- **IoT applications:** sensor data storage

These applications need fast processing and scalability.

Database Selection by Businesses

Businesses select databases based on:

- Type of data
- Performance requirements
- Scalability needs
- Budget and infrastructure

Many organizations now use hybrid approaches, combining SQL and NoSQL databases.

5. Future Trends & Challenges

NewSQL Databases

NewSQL databases aim to combine:

- SQL's consistency
- NoSQL's scalability

They are becoming popular in cloud-based systems.

Challenges

- Data security
- Maintaining consistency
- Managing large-scale data

Future Trends

- Cloud-native databases
- AI-driven database optimization
- Multi-model databases

6. Conclusion

Relational and NoSQL databases both play vital roles in modern applications. Relational databases are ideal for applications requiring strong consistency and structured data, while NoSQL databases are suitable for scalable and flexible systems handling large data volumes. Understanding their differences helps developers and organizations choose the right database for their application needs.

How Data Travelled in Distributed Database?

Data Flow in Distributed Database

User sends request (Query) to the system

Request goes to a **Coordinator Node**

Coordinator checks:

- Where data is stored
- Which server (node) has the required data

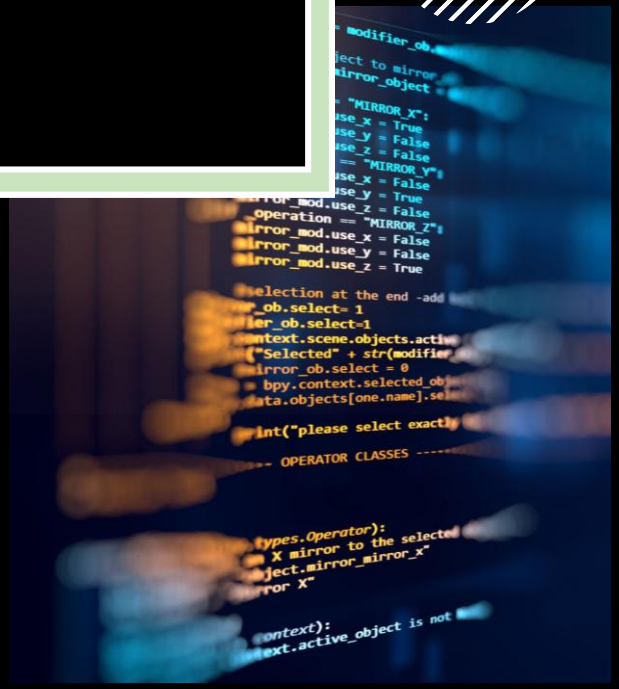
Query is divided into smaller parts (if needed)

Sub-queries are sent to different database servers

Simple Flow:
User → Coordinator → Multiple Database Nodes

UnStructured To Structured Database

- **Definition:**
  Converting unstructured data into structured data means extracting useful information and organizing it into tables (rows and columns).
- **Why Conversion is Needed:**
- Easy data analysis
- Faster searching and reporting
- Better decision making
- Can be stored in databases like MySQL or Oracle
- **Common Unstructured Data:**
- Text documents (PDF, Word)
- Emails
- Images
- Audio and video
- Social media posts



- **Steps to Convert Unstructured Data to Structured Data**
- **1. Data Collection**
- Gather unstructured data from files, emails, websites, etc.
- **2. Data Processing**
- Text: use keyword extraction or NLP
- Images: use OCR to extract
- Audio/Video: use speech-to-text
- **3. Data Cleaning**
- Remove noise, errors, duplicates
- Standardize formats
- **4. Data Structuring**
- Organize extracted data into tables
- Define columns (Name, Date, Amount, etc.)
- **5. Store in Database**
- Save structured data in SQL databases