Big Data Analytics

VIT Vidyalankar Institute of Technology
Accredited A+ by NAAC

Academic Year 2022-23

Index

1. NoSQL
2. NoSQL v/s RDBMS
3. NoSQL Database Type

Subject Teacher: **Prof. Prakash Parmar, Assistant Professor, CMPN**

eCommerce scenario

The number of eCommerce users worldwide and the number of products offered by the online retailers are increasing at an exponential pace crossing hundreds of millions. One of the leading online retailers has over two million sellers worldwide.

As the volume of users, products and sellers increase, database for the eCommerce system are required to do the following:

- Manage increasing loads of product, customer and seller data
- Support heterogeneous forms of data including images, videos and text
- Handle concurrent transactions done by millions of online customers and vendor operations
- Process millions of user requests within milliseconds
- Reduce overall operational cost

database implementations for solving the given requirements:

- **SQL databases**
  - Relational (tabular) databases with a rigid structure and limited storage.

- **NoSQL (Not only SQL) databases**
  - Very flexible non-relational (non-tabular) databases with unlimited storage.

Which is suitable ?

Suitable database – NoSQL

VIT Vidyalankar Institute of Technology
Accredited A+ by NAAC

Flexible Schema:
In an eCommerce product catalog, the attributes of products can vary greatly. Some products may have additional attributes (e.g., color, size, weight) while others may not. With a NoSQL database like MongoDB or Cassandra, you can easily accommodate this variability without needing to modify the schema every time a new product category is added. This flexibility is crucial in an ever-expanding catalog.

Scalability:
eCommerce platforms experience fluctuating traffic, especially during holidays or promotions. NoSQL databases are designed to scale horizontally, allowing you to distribute the database across multiple servers or clusters. This ensures that the platform can handle both peak and regular traffic without compromising performance.

High Read and Write Throughput:
In an eCommerce scenario, thousands of users are simultaneously browsing, searching, and making purchases. NoSQL databases are optimized for high read and write throughput, making it possible to serve product information to users quickly and update product details in real-time.

Document Stores:
Many NoSQL databases, like MongoDB, are document stores that store data in a JSON-like format. This makes it easy to represent complex product data with nested attributes, such as product variants, reviews, and recommendations. Retrieving entire product documents in one query can improve query performance.

Suitable database - NoSQL

Vidyalankar Institute of Technology
Accredited A+ by NAAC

High Availability:
Downtime in eCommerce can lead to lost sales and customer dissatisfaction. NoSQL databases can be configured for high availability and fault tolerance. They can handle server failures and network issues without significant disruptions in service.

Geo-distribution:
Large eCommerce companies often serve customers worldwide. NoSQL databases can be set up with multi-region or global distribution to minimize latency for users in different geographic locations. This ensures that product information loads quickly for customers, regardless of their location.

Analytics and Personalization:
NoSQL databases can store vast amounts of historical data, including user behavior, purchase history, and product interactions. This data can be leveraged for analytics and personalization, helping eCommerce companies recommend products to users based on their preferences and behavior.

Schema Evolution:
As an eCommerce business evolves, the data model may change. NoSQL databases can accommodate schema changes more easily than traditional relational databases, which require complex migrations.

In conclusion, NoSQL databases are well-suited for eCommerce product catalogs due to their flexibility, scalability, high throughput, support for complex data structures, and ability to handle global distribution and high availability. These features enable eCommerce companies to provide a seamless and responsive shopping experience for their customers while efficiently managing a vast and dynamic product catalog.

NoSQL (Not Only SQL)

VIT Vidyalankar Institute of Technology
Accredited A+ by NAAC

- A NoSQL database (also known as “no SQL” or “not only SQL”) is a distributed, non-relational database designed approach for large-scale data storage and massively parallel, high-performance data processing across many commodity systems.
- It is a modern data storage paradigm that provides data persistence for environments where high performance is the primary requirement. Within a NoSQL database, data is stored so that both writing and reading are fast, even under heavy load.
- NoSQL Database does not require a fixed schema. It avoids joins, and is easy to scale. The major purpose of using a NoSQL database is for distributed data stores with humongous data storage needs.
- NoSQL is used for Big data and real-time web apps. For example, companies like Twitter, Facebook and Google collect terabytes of user data every single day.

Feature / Characteristics of NoSQL

Schema Flexibility:

NoSQL databases offer flexible schema design, allowing data to be inserted without a predefined schema or with a schema that can evolve over time.

Data structures can vary from one record to another within the same database or collection, making NoSQL databases ideal for handling unstructured or semi-structured data.

Scalability:

NoSQL databases are designed for horizontal scalability, meaning you can distribute data across multiple servers or nodes to accommodate growing data volumes and traffic.

They can handle massive datasets and high concurrent read and write operations by adding more hardware resources.

Data Types and Models:

NoSQL databases support various data types and models, including key-value stores, document-oriented databases, column-family stores, and graph databases.

You can choose the appropriate data model based on the nature of your data and the requirements of your application.

Feature / Characteristics of NoSQL

Vidyalankar Institute of Technology
Accredited A+ by NAAC

Distributed and Fault-Tolerant:

NoSQL databases are often designed to be distributed, with data distributed across multiple servers or nodes.

They are built to withstand hardware failures, maintain data availability, and ensure data consistency in distributed environments.

High Performance:

Many NoSQL databases are optimized for high read and write throughput, enabling rapid data retrieval and updates.

They are well-suited for applications that require low-latency access to data, such as real-time analytics and content delivery.

Support for Big Data:

NoSQL databases are commonly used in big data applications, where traditional relational databases may struggle to handle large volumes of data.

They play a crucial role in big data processing frameworks and analytics tools.

Low Latency and High Throughput:

NoSQL databases are designed for real-time and near-real-time applications, making them suitable for use cases like real-time analytics, IoT, and streaming data processing.

They provide low-latency access to data and support high throughput for concurrent operations.

Characteristics or Feature of NoSQL

Vidyalankar Institute of Technology
Accredited A+ by NAAC

CAP Theorem Considerations:

NoSQL databases are designed with the CAP theorem in mind, which states that in a distributed system, you can achieve at most two out of three guarantees: consistency, availability, and partition tolerance.

Depending on the database type, NoSQL databases may prioritize different aspects of the CAP theorem to meet specific application requirements.

Rich Querying Capabilities:

Some NoSQL databases offer powerful querying capabilities that enable complex data retrieval and analysis, even in schema-less or semi-structured data models.

Geo-distribution:

Many NoSQL databases can be configured for geo-distribution, enabling data to be replicated and served from multiple geographic regions to reduce latency and improve global availability.

SQL Vs NoSQL

VIT Vidyalankar Institute of Technology
Accredited A+ by NAAC

Parameter	SQL databases	NoSQL databases
Storage	Better for relational data (small dataset).	Better for Big data (large dataset).
Architecture	Centralized - single node dependency	Distributed - set up multiple machines
Scalability	Scale up vertically - hardware upgraded as load increases	Scale out horizontally - new machines are added as load increases
Data model	Fixed schema	Flexible schema - can store images, audio, video and text
Language	Well known language- SQL	Every NoSQL database have its own lang.
ACID properties	Support ACID properties.	Not supported ACID properties.
Data Integrity	Support Database constraint.	Not support Database constraint .
High reliability	Requests cannot be processed if the central server is down	Data is copied to multiple nodes overcoming single node failure
Latency	On-disk processing slows down query performance	In-memory (caching) processing delivers sub-millisecond response
Example	MySQL, Oracle, PostgreSQL, IBM DB2, Microsoft SQL Server	MongoDB, Apache Cassandra, Apache Hbase, Neo4j, Redis, ScyllaDB

NoSQL Databases / NoSQL Data Architecture Patterns

Vidyalankar Institute of Technology
Accredited A+ by NAAC

NoSQL data architecture patterns refer to common design approaches and strategies for organizing and storing data in NoSQL databases.

These patterns help developers and architects make informed decisions on how to structure data in NoSQL databases based on the specific needs of their applications.

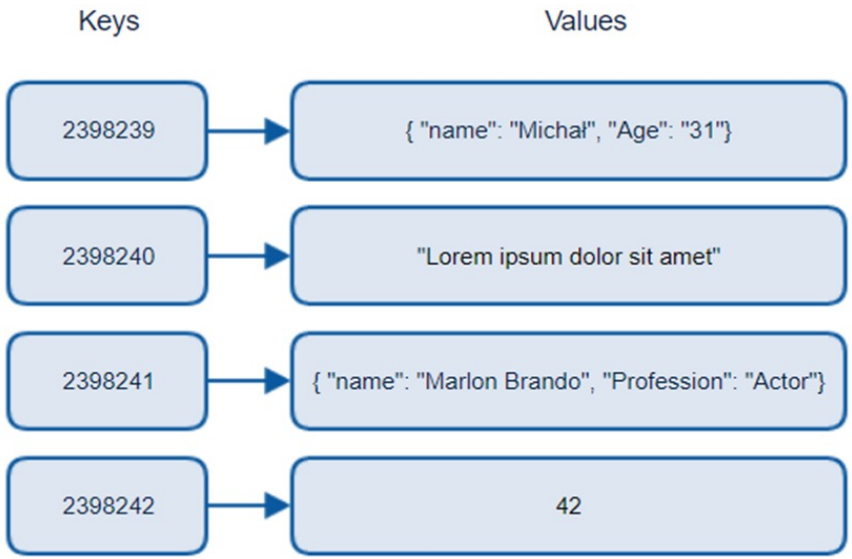
NoSQL databases offer different data models and storage mechanisms, and these patterns provide guidance on how to leverage these features effectively.

The data is stored in NoSQL in any of the following four data architecture patterns.

1. Key-Value Store Database
2. Column Store Database
3. Document Database
4. Graph Database

Key-Value database

- The Key-Value Store NoSQL pattern is one of the simplest and most straightforward data storage patterns where data is stored as simple key-value pairs.
- In this pattern, each piece of data is associated with a unique key, which is used to access or retrieve the data.
- This pattern is highly efficient for read and write operations, making them suitable for scenarios where quick and efficient lookup by a known key is essential.



Key-Value database: Use case & Example

- **Use Case**: Caching, session management, and high-speed data access where simple lookups by key are sufficient.

- **Example Database**: Redis, Amazon DynamoDB

Key-Value database

Vidyalankar Institute of Technology
Accredited A+ by NAAC

Example: Redis to store product information and manage a user's shopping cart.

Storing Product Information:
We can store product information as key-value pairs in Redis, with each product having a unique product ID as the key and product details (e.g., name, price) as the value.

```
# Example product data in Redis
SET product:101 "{\"name\": \"Laptop\", \"price\": 999.99}"
SET product:102 "{\"name\": \"Smartphone\", \"price\": 599.99}"
SET product:103 "{\"name\": \"Tablet\", \"price\": 299.99}“
```

Managing a User's Shopping Cart:

Redis is well-suited for managing user shopping carts efficiently. We can use the user's session ID as the key, and the shopping cart contents as the value (typically a list or set).

```
# Example shopping cart for user with session ID "user123"
RPUSH cart: user123 101 102 103
```

In this example, we use RPUSH to add product IDs (101, 102, 103) to the user's shopping cart. RPUSH is used to push items to the end of a list in Redis.

Key-Value database

Retrieving Product Information:

When a user wants to view their shopping cart or checkout, we can efficiently retrieve product information from Redis using the stored product IDs.

```
# Get product details for the items in the user's cart
GET product:101
GET product:102
GET product:103
```

These commands retrieve product information for the items in the user's shopping cart.

Updating the Shopping Cart:

When the user adds or removes items from their cart, Redis allows for easy updates:

```
# Add a new item (e.g., product ID 104) to the user's cart
RPUSH cart:user123 104

# Remove an item (e.g., product ID 102) from the user's cart
LREM cart:user123 1 102
```

The RPUSH command appends an item to the end of the list, and LREM removes a specified number of instances of an item from the list.

Key-Value database

Vidyalankar Institute of Technology
Accredited A+ by NAAC

WHEN TO USE KEY VALUE STORES ?

Key-value stores handle size well and are good at processing a constant stream of read/write operations with low latency

making them perfect for,

1. Session management at high scale
2. User preference and profile stores
3. Can effectively work as a cache for heavily accessed but rarely updated data

ADVANTAGES OF KEY VALUE STORES,

1. Quick look-ups using the key.
2. The relationship between data does not have to be calculated by a query language, there is no optimization performed.
3. Can store in multiple machine on distributed systems and don't need to worry about where to store indexes, how much data exists on each system or the speed of a network with a distributed system they just work.

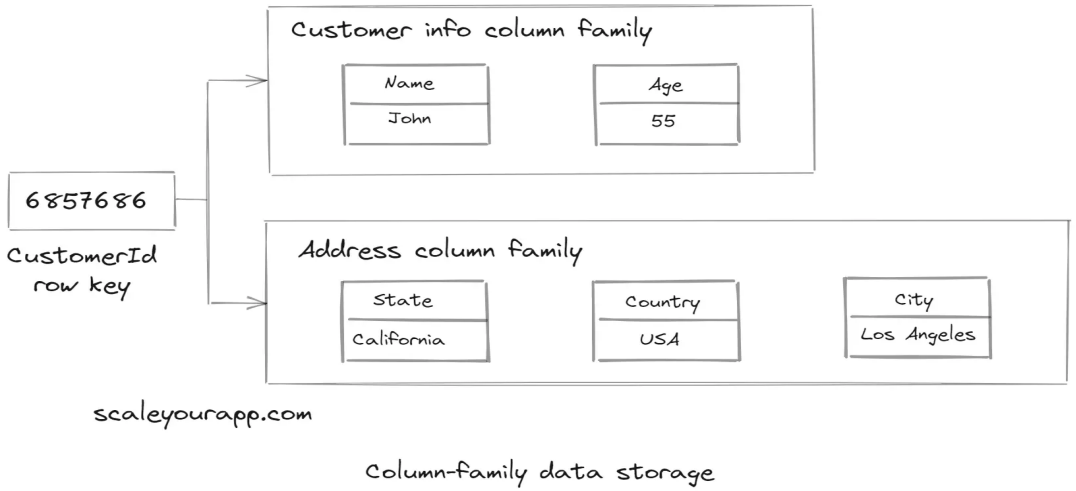
DISADVANTAGES OF KEY VALUE STORES,

1. No complex query filters
2. All joins must be done in code
3. No foreign key constraints
4. No trigger

Column Family database

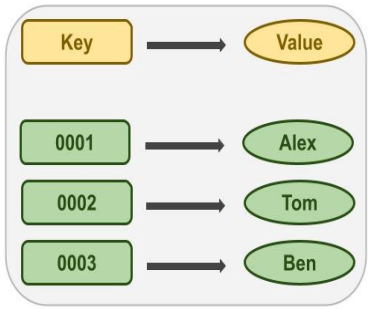
Vidyalankar Institute of Technology
Accredited A+ by NAAC

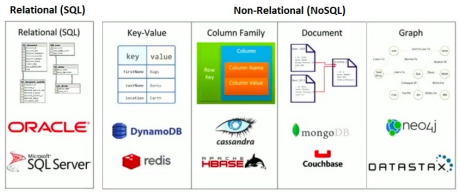
- A key-value database (also called key-value store) uses a simple key-value method to store data.
- These databases contain a simple string (the key) that is always unique and an arbitrary large data field (the value).
- They are easy to design and implement.



Column-family data storage

NoSQL Databases



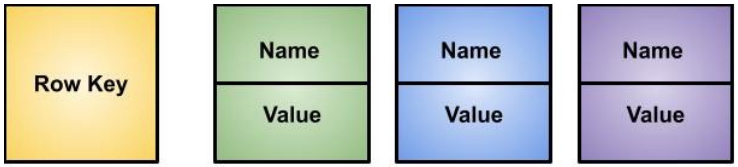


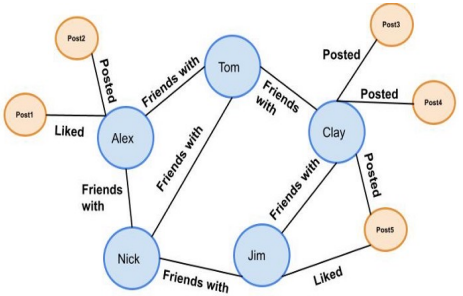
Document 1

Id: "001",
Name: "Alex Bill",
Phone: "+001 234 5341",
Department: "Finance"

Document 2

Id: "003",
Name: "Alex Bill",
Phone: {
Home: "+001 234 5341",
Office: "+001 111 2759"
}
Department: "Finance"





NoSQL database

VIT Vidyalankar Institute of Technology
Accredited A+ by NAAC

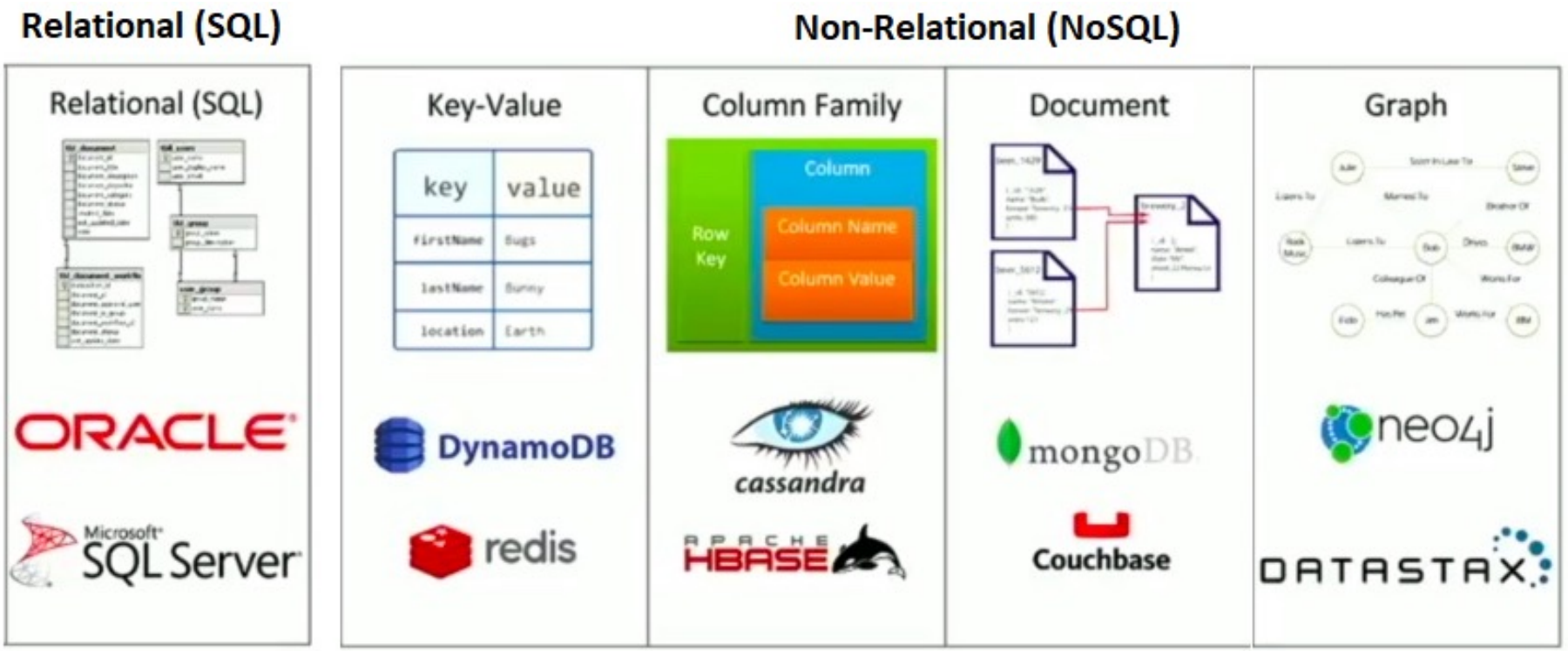
	Key-Value	Column-Family	Document-Oriented	Graph-Based
Databases	• Redis; • Riak • …	• Cassandra • HBase • …	• MongoDB • Couchbase • …	• Neo4j • OrientDB • …
Features	• Simple design • Fast Read/Write • No indexes on non-key fields	• Storage not wasted on NULL values • Very useful for wide rows • Best suitable for Big Data Operations	• Flexible schema • Secondary Indexes • Rich query Langauge • Allows nested data structures	• Perfect for highly interrelated data • Data is stored in nodes and edges • Quick graph traversals
Use when	• You are just dealing with bytes/string • Your data is not highly related • All you need is basic CRUD	• You need very fast column operations including aggregation • Need compression or versioning	• You don't know much about the schema • The schema is likely to change often	• Your data looks more like a graph • Shortest path traversal

NoSQL database

VIT Vidyalankar Institute of Technology
Accredited A+ by NAAC

	Key-Value	Column-Family	Document-Oriented	Graph-Based
Suitable use cases	•Session information •User profiles, Preferences •Shopping cart	•Event logging •Content Management System (CMS), Blogging platforms •Expiring data	•Event logging •CMS, Blogging platforms •Web Analytics or real-time analytics •eCommerce (product catalog, orders)	•Routing, dispatch •Location-based services •Recommendation engines
Challenges	•Relationships among data •Query by value	•Need to know access patterns •Keys design is complex	•You need complex join-like queries •Circular dependencies	•Does not scale well horizontally
When not to use	•Not suitable for complex querying	•Column family design changes	•Data is highly related •If efficiency is preferred over consistency	•Updating all or subset of entities •Operations involving whole graph

NoSQL Databases / NoSQL Data Architecture Patterns



SQL Vs NoSQL

Vidyalankar Institute of Technology
Accredited A+ by NAAC

Problem Statement:
Assume an organization needs to create an online platform for its employees to create posts and add various images, videos, audio, and comments. Any employee can comment on these posts and rate them. Employees can join different groups as per their interests. The landing page will have a feed of posts that employees can share and interact with.

For the given scenario, suggest which type of database implementation (SQL / NoSQL) would be most suitable and specify appropriate reasons for your choice of the database implementation.

SQL Vs NoSQL

Vidyalankar Institute of Technology
Accredited A+ by NAAC

Best suitable database implementation would be NoSQL databases.

Following are the reasons:

For the given scenario, there are several entities such as employee, post, comments, images, audios, videos, rating and so on.

You have several relationships that link these entities as shown in the table below:

Entity	Relationship	Entity
post	*written by*	employee
post	*with many*	comments
comments	*created by*	employee
rating	*assigned by*	employee
post	*with many*	rating
post	*with many*	images

SQL Vs NoSQL

VIT Vidyalankar Institute of Technology
Accredited A+ by NAAC

Use NoSQL database implementation because,

NoSQL databases are very flexible in structure and can store all types of related data in one place

User can retrieve the whole post with a single query avoiding joins thus increasing the performance

Data on NoSQL databases scale out naturally and hence able to deal with the continuous streaming of posts

Using SQL databases is not suggested as,

Several joins are used to display the post containing various forms of data which is very time consuming

Data is existing in heterogeneous forms that SQL does not support

Continuous streaming of posts that are dynamically loaded onto the screen require thousands of queries to be performed

Learn Fundamentals & Enjoy Engineering

Prof. Prakash Parmar
Assistant Professor
Computer Engineering Department
Vidyalankar Classes CSE GATE Faculty