SQL and NoSQL

What is SQL?

Structured Query Language

Databases are relational databases that store data in tables with predefined schemas. They use SQL for querying and managing data. Examples include MySQL, PostgreSQL, and Microsoft SQL Server.

What is NoSQL?

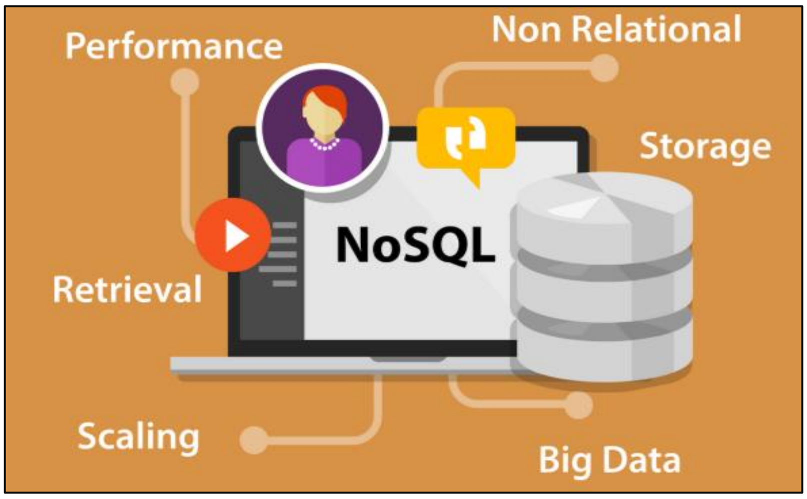
Not Only SQL

Databases are non-relational and store data in various formats like key-value pairs, documents, graphs, or wide-columns. They are designed for flexibility, scalability, and handling large volumes of unstructured or semi-structured data. Examples include MongoDB, Cassandra, and Redis.

When to Choose SQL?

- Your data is structured and relationships between entities are important.
- You need ACID compliance (Atomicity, Consistency, Isolation, Durability).
- Complex queries and joins are required.
- You're working with legacy systems or enterprise-grade applications.

When to Choose NoSQL?



- You're dealing with large volumes of unstructured or semi-structured data.
- You need high scalability and performance across distributed systems.
- Your application requires flexible schema design.
- You're building real-time applications like chat apps, recommendation engines, or IoT platforms.

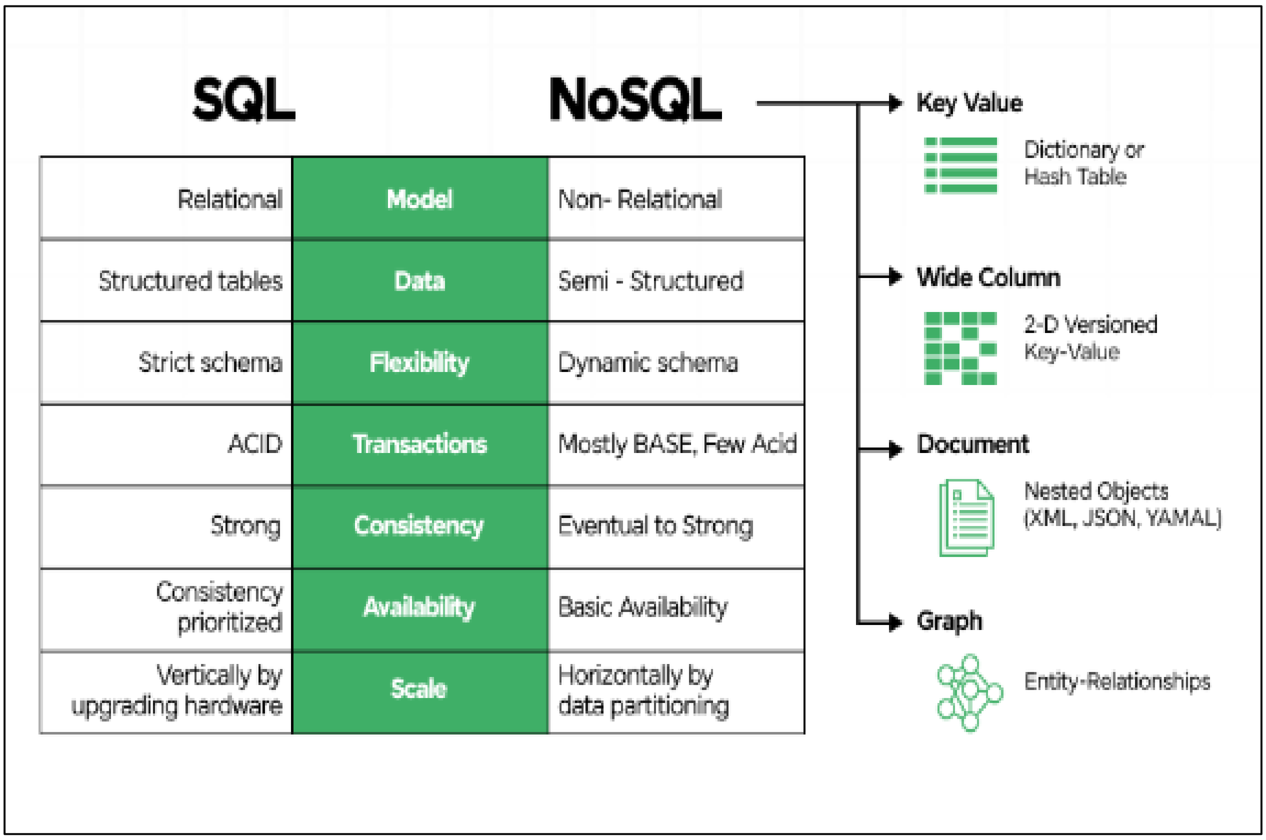
Advantages of SQL

- **Structured Data Storage**: Ideal for structured data with clear relationships.
- **ACID Compliance**: Ensures data integrity and reliability.
- **Powerful Query Language**: SQL is standardized and widely used.
- **Mature Ecosystem**: Well-supported with tools, documentation, and community.

Advantages of NoSQL

- **Scalability**: Easily handles massive data across distributed systems.
- **Flexibility**: Schema-less design allows quick changes and iterations.
- **High Performance**: Optimized for specific use cases like caching, real-time analytics.
- **Variety of Models**: Supports document, key-value, graph, and column-family models.

Comparison between SQL and NoSQL



SQL		NoSQL
Relational	**Model**	Non- Relational
Structured tables	**Data**	Semi - Structured
Strict schema	**Flexibility**	Dynamic schema
ACID	**Transactions**	Mostly BASE, Few Acid
Strong	**Consistency**	Eventual to Strong
Consistency prioritized	**Availability**	Basic Availability
Vertically by upgrading hardware	**Scale**	Horizontally by data partitioning

Key Value	Dictionary or Hash Table
Wide Column	2-D Versioned Key-Value
Document	Nested Objects (XML, JSON, YAMAL)
Graph	Entity-Relationships