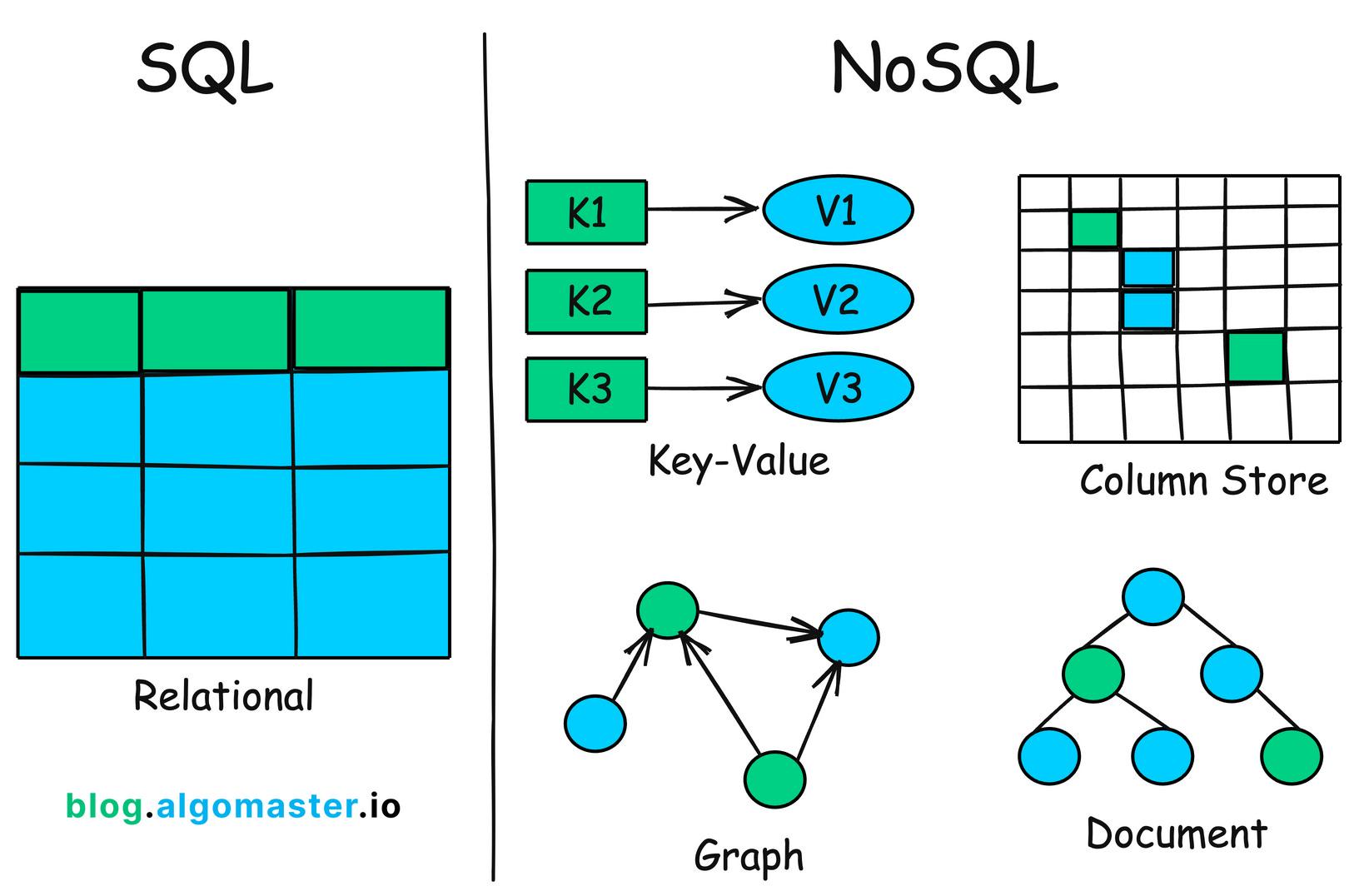
SQL & NOSQL



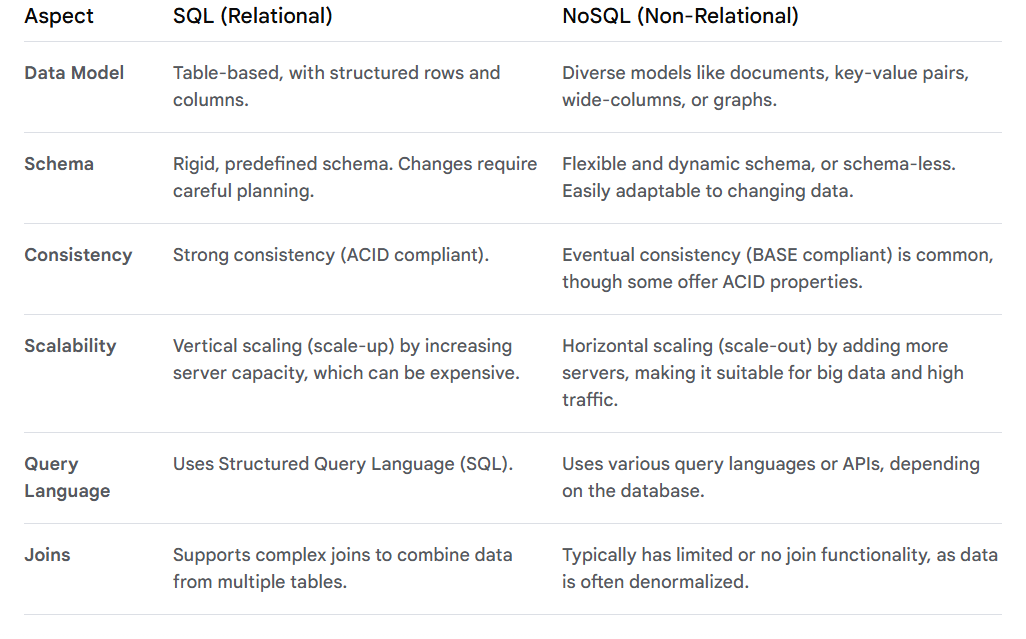
## What is SQL?

* SQL (Structured Query Language) is the standard language used for interacting with relational databases (RDBMS). It organizes data into tables with predefined schemas, where information is stored in rows and columns. SQL allows you to perform operations such as creating, retrieving, updating, and deleting data.
* **Key features of SQL:**
  + Structured schema: A fixed, predefined schema is required before data can be stored. This ensures a high level of data integrity.
  + ACID compliance: Transactions adhere to the Atomicity, Consistency, Isolation, and Durability (ACID) properties, which ensures reliable data management.
  + Standardized language: SQL is an ANSI/ISO standard, providing consistency across different RDBMS platforms like MySQL, Oracle, and PostgreSQL.
  + Complex queries: It can handle complex queries and joins across multiple tables efficiently.
  + Vertical scalability: Scaling is typically done by adding more resources (CPU, RAM) to a single server.

## What is NoSQL?

* NoSQL (Not only SQL) databases are non-relational database management systems designed for unstructured or semi-structured data. Unlike SQL databases, NoSQL systems do not require a predefined schema and can handle large volumes of diverse data.
* **Key features of NoSQL:**
  + Flexible schema: NoSQL databases have a dynamic schema, allowing for faster development and easier adaptation to changing data requirements.
  + BASE model: Many NoSQL databases follow the Basically Available, Soft state, and Eventually consistent (BASE) model, prioritizing availability and partition tolerance over strong consistency.
  + Horizontal scalability: NoSQL databases are designed to scale out horizontally by adding more servers to a distributed system, making them cost-effective for handling massive data volumes.
  + Diverse data models: There are several types of NoSQL databases, each with a different approach to data storage:
    - Document: Stores data in JSON-like documents (e.g., MongoDB).
    - Key-value: Stores data as simple key-value pairs (e.g., Redis).
    - Columnar: Stores data in columns instead of rows, optimized for analytical queries (e.g., Cassandra).
    - Graph: Represents data as nodes and edges, ideal for mapping relationships (e.g., Neo4j).

## Comparison of SQL and NoSQL



## Scenarios to choose SQL vs. NoSQL

* Choose SQL when:
  + **Structured data and fixed schema are required:** Ideal for traditional business applications like financial systems, inventory management, or customer relationship management (CRM) where data is well-defined and unlikely to change frequently.
  + **Data integrity and ACID compliance are essential:** When strong consistency and reliability are non-negotiable, such as in banking and e-commerce transactions, SQL's ACID properties (Atomicity, Consistency, Isolation, Durability) ensure data accuracy.
  + **Complex queries and reporting are common:** Relational databases excel at performing complex queries involving joins, aggregations, and subqueries, making them suitable for business intelligence and data analytics.
  + **A mature ecosystem and broad support are needed:** With decades of development, SQL databases have a vast ecosystem of tools, documentation, and a large community of experts, which can simplify management and troubleshooting.
  + **Vertical scaling is sufficient:** For applications with a predictable workload, scaling vertically (upgrading a single server with more power) is a straightforward way to handle increased demand.
* Choose NoSQL when:
  + **Handling large volumes of semi-structured or unstructured data:** Perfect for big data, IoT devices, and social media, where data comes in various formats and doesn't fit a rigid table structure.
  + **Extreme scalability and high performance are needed:** NoSQL databases are designed for horizontal scaling (adding more servers) to handle massive datasets and high traffic with high availability and speed, which is crucial for real-time applications.
  + **Rapid development and flexible schema are prioritized:** In agile development environments, the dynamic schema of NoSQL allows developers to iterate quickly without the overhead of complex database migrations.
  + **The data model is specialized:** Different types of NoSQL databases (document, key-value, graph) are optimized for specific use cases. For example, a graph database is better for modeling relationships in a social network.
  + **Availability is more critical than strong consistency:** Many NoSQL databases follow the BASE model (Basically Available, Soft state, Eventually consistent), which provides high availability even if data is not immediately consistent across all nodes.

## Advantages of SQL

* **Data integrity:** SQL databases enforce data integrity and consistency through a predefined schema, ensuring data is valid and reliable.
* **Standardized language:** As an ANSI/ISO standard, SQL offers a common language for interacting with relational databases, providing consistency and broad tooling support across platforms.
* **Powerful querying:** The declarative nature of SQL makes it easy to write complex queries, perform powerful aggregations, and combine data from multiple tables using joins.
* **Security features:** SQL databases often come with robust security features, including user authentication, encryption, and access control to protect data.
* **Mature ecosystem:** The long history of SQL has led to a mature ecosystem with abundant resources, experienced developers, and a wide array of tools for management, reporting, and analysis.

## Advantages of NoSQL

* **Flexibility and agile development:** The flexible schema allows developers to store diverse data types and adapt data models quickly, accelerating the development cycle.
* **Horizontal scalability:** NoSQL databases scale out by adding more servers, offering a cost-effective way to handle massive datasets and high traffic loads common in modern web applications.
* **High performance:** Optimized for specific data models and access patterns, NoSQL can deliver ultra-low latency for high-speed data ingestion and retrieval.
* **Cost efficiency:** Horizontal scaling on commodity hardware can be more cost-effective for managing large datasets compared to the expensive, powerful hardware required for vertical scaling in some SQL databases.
* **High availability:** With distributed architectures and data replication across nodes, NoSQL databases are resistant to single points of failure, ensuring high availability and resilience.

## 