# SQL and NOSQL

## SQL:

**SQL (Structured Query Language)** is used to manage **relational databases** where data is stored in **tables with rows and columns**. It’s ideal for structured data and supports complex queries and transactions.

**Example: Airline Reservation System**

* **Use Case**: Booking flights, managing passenger records, and handling payments.
* **Why SQL**: Requires structured data, complex joins (e.g., flights ↔ passengers ↔ payments), and strong consistency.
* **Typical Database**: PostgreSQL or Oracle.

**Advantages**

* **Structured Data**: Ideal for well-defined schemas and relationships.
* **ACID Compliance**: Ensures reliable transactions and data integrity.
* **Powerful Queries**: Supports complex joins and aggregations.
* **Mature Ecosystem**: Long-standing tools, documentation, and community support.

**Disadvantages**

* **Rigid Schema**: Difficult to adapt to changing data structures.
* **Vertical Scalability**: Limited by server capacity unless cloud-optimized.
* **Less Ideal for Big Data**: Struggles with high-volume, unstructured data.
* **Setup Overhead**: Requires careful schema design and normalization.

## NoSQL

**NoSQL (Not Only SQL)** refers to **non-relational databases** that store data in formats like **documents, key-value pairs, graphs, or wide-columns**. It’s designed for flexibility, scalability, and handling unstructured or semi-structured data.

**Example: Netflix Streaming Platform**

* **Use Case**: Storing user preferences, viewing history, and personalized recommendations.
* **Why NoSQL**: Handles massive, varied, and fast-changing data across millions of users.
* **Typical Database**: Cassandra or Amazon DynamoDB.

**Advantages**

* **Flexible Schema**: Easily handles dynamic and varied data formats.
* **Horizontal Scalability**: Designed for distributed systems and big data.
* **High Performance**: Optimized for fast read/write operations.
* **Variety of Models**: Supports document, key-value, graph, and column stores.

**Disadvantages**

* **Limited Query Capabilities**: Complex queries can be harder to implement.
* **Eventual Consistency**: May sacrifice immediate accuracy for speed.
* **Less Mature**: Newer tools may lack robust documentation or standards.
* **Data Integrity Risks**: Weaker transactional guarantees compared to SQL.

### Quick Summary

* Use **SQL** when your data is structured, relationships matter, and consistency is key.
* Use **NoSQL** when your data is varied, scalability is critical, and flexibility is needed.

|  |  |  |
| --- | --- | --- |
| **Feature** | **SQL (Relational)** | **NoSQL (Non-relational)** |
| **Data Structure** | Tables with rows and columns | Document, key-value, graph, or column-based |
| **Schema** | Fixed schema (predefined structure) | Flexible schema (dynamic and adaptable) |
| **Scalability** | Vertically scalable (upgrade hardware) | Horizontally scalable (add more servers) |
| **Query Language** | SQL | Varies (e.g., MongoDB uses its own syntax) |
| **Transactions** | ACID-compliant (strong consistency) | BASE-compliant (eventual consistency) |
| **Performance** | Best for complex queries and joins | Optimized for large-scale, fast read/write ops |
| **Use Cases** | Banking, ERP, CRM, legacy systems | Big data, real-time apps, IoT, content storage |
| **Examples** | MySQL, PostgreSQL, Oracle, MS SQL Server | MongoDB, Cassandra, CouchDB, Neo4j |

### Scenarios:

**Scenario 1: Banking System — Choose SQL**

* Why: You need strong consistency, complex transactions, and structured data (e.g., account balances, transfers).
* SQL Advantage: ACID compliance ensures data integrity, which is critical in financial applications.

**Scenario 2: Social Media App — Choose NoSQL**

* Why: You’re dealing with massive volumes of user-generated content, likes, comments, and media—all semi-structured or unstructured.
* NoSQL Advantage: Scales horizontally and handles flexible data formats like JSON, perfect for evolving schemas.

**Scenario 3: E-commerce Product Catalogue — Choose NoSQL**

* Why: Products vary widely in attributes (e.g., electronics vs. clothing), and you need fast reads and writes.
* NoSQL Advantage: Document-based databases like MongoDB allow dynamic fields and quick updates without schema changes.