No-SQL 101

Monday, October 27, 2025 11:57 PM

Issues with traditional SQL DB's.

1. No in-built Scalability
   a. Lower scalability and not real-time
   b. Can have latency at petabytes scale of data
2. Flexibility
   a. Schema once created is harder to edit in production running apps
   b. Not fully agile, and in a fast paced world, may require constant changes / new requirements
3. Performance
   a. More tables and relations -> harder to fetch the data
   b. Latency increases with the data

No-SQL databases

1. Documents and collections in JSON like document
   a. Highly scalable, no document restrictions, fully customizable.
   b. Fully flexible to edit anytime
   c. Significantly higher performance compared to SQL
2. Where to be used / preferred
   a. Thumb rule: I don't need data table restrictions / validations but need flexibility and performance.
      i. Big-data apps like social media / video sharing apps
      ii. Content management high availability websites or apps
      iii. User data management / data hubs

Analogy between SQL and No-SQL

SQL Database	No SQL Database
Database	Schema
Table	Collection
Row	Document
Column	Field
Primary Key	_id
Foreign Key	Embedded key / Reference document
Index	Index
Join	Aggregation look-up pipelines

Feature	SQL (Relational)	NoSQL (Non-relational)
Data Model	Tables with rows and columns	Key-Value, Document, Column-Family, Graph
Schema	Fixed, predefined schema	Flexible, dynamic schema
Query Language	SQL (Structured Query Language)	Varies: MongoDB uses JSON-like queries, Cassandra uses CQL, etc.
Transactions	ACID (Atomicity, Consistency, Isolation, Durability)	BASE (Basically Available, Soft-state, Eventually consistent)
Scalability	Vertical scaling (scale-up)	Horizontal scaling (scale-out)
Performance	Excellent for complex joins & structured queries	Optimized for large-scale read/write workloads
Use Cases	Banking, ERP, CRM, structured data	Big data, real-time apps, content management, IoT
Examples	MySQL, PostgreSQL, Oracle, SQL Server	MongoDB, Cassandra, DynamoDB, Redis, Couchbase
Relationships	Strong, enforced via foreign keys	Often denormalized; relationships handled at application level
Consistency	Strong consistency	Eventual consistency (varies by system)
Flexibility	Low (schema changes require migrations)	High (schema can evolve easily)
Best For	Structured, relational data	Unstructured or semi-structured data, high velocity & volume