Working with NoSQL Databases Using Python

Pinal Dave

Data Expert

@pinaldave | blog.sqlauthority.com

Why Do We Need NoSQL?

Traditional relational databases struggle with scalability and flexibility

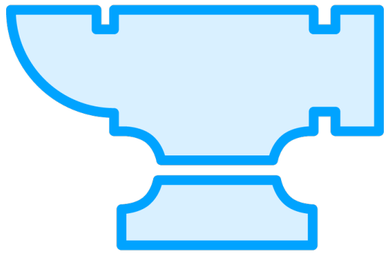
NoSQL is designed for high-volume, distributed, and unstructured data

Ideal for real-time applications, IoT, social media, and big data analytics

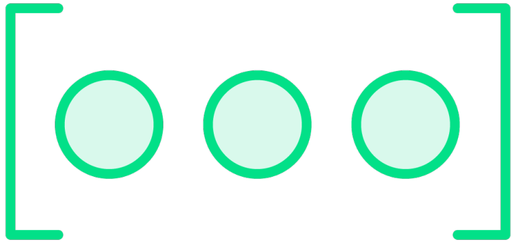
NoSQL vs. SQL - Key Differences

Feature	SQL (Relational DBs)	NoSQL (Non-relational DBs)
Schema	Fixed, predefined	Flexible, schema-less
Scalability	Vertical (scale-up)	Horizontal (scale-out)
Data storage	Tables (rows & columns)	JSON, key-value, columnar, graph
Transactions	ACID	BASE (eventual consistency)
Use cases	Structured data, financial apps	Big data, real-time apps

Scaling NoSQL Databases



Large-scale applications demand distributed data storage

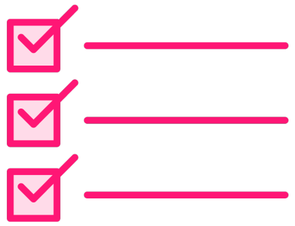


NoSQL databases scale horizontally (adding more nodes)

Proper indexing and partitioning are crucial for performance

Balancing Reads and Writes

Depending on the workload, different strategies can enhance performance and scalability while maintaining data consistency.

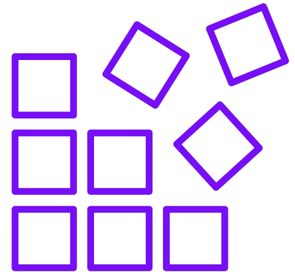


Read-heavy workloads

Use caching (Redis, Memcached)

Write-heavy workloads

Use batch writes and optimize schema



Eventual consistency

Improves performance but may lead to stale data