**Free AWS Training** | Advance your career with AWS Cloud Practitioner Essentials—a free, four-hour, foundational course »

# What is NoSQL?

Get started with DynamoDB

Get started with MemoryDB

Get started with ElastiCache

Get started with Neptune

Get started with DocumentDB

NoSQL databases are purpose-built for specific data models and stores data in flexible schemas that scale easily for modern applications. NoSQL databases are widely recognized for their ease of development, functionality, and performance at scale. This page includes resources to help you better understand NoSQL databases and to get started.

## What are the advantages of NoSQL databases

Modern applications face several challenges that can be solved by NoSQL databases. For instance, applications process a large data volume from disparate sources like social media, smart sensors, and third-party databases. All of this disparate data doesn't fit neatly into the relational model. Enforcing tabular structures can lead to redundancy, data duplication, and performance issues at scale.

NoSQL databases are purpose-built for non-relational data models and have flexible schemas for building modern applications. They are widely recognized for their ease of development, functionality, and performance at scale. Benefits of NoSQL databases are listed below.

### Flexibility

NoSQL databases generally provide flexible schemas that enable faster and more iterative development. The flexible data model makes NoSQL databases ideal for semi-structured and unstructured data.

### Scalability

NoSQL databases are typically designed to scale out by using distributed clusters of hardware, as opposed to scaling up by adding expensive and robust servers. Some cloud providers handle these operations behind the scenes as a fully managed service.

### High performance

NoSQL databases are optimized for specific data models and access patterns. These enable higher performance than if you were trying to accomplish similar functionality with relational databases.

### Highly functional

## What are the use cases of NoSQL databases

You can use NoSQL databases to build a wide variety of high-performance mobile, Internet of Things (IoT), gaming, and web applications that provide great user experiences at scale. The range of NoSQL databases and their respective uses cases are wide-ranging. While it is challenging to present a representative set of use cases, below we provide a few illustrative examples as thought-starters and encourage you to learn more about each NoSQL database and their respective uses cases.

### Real-time data management

You can deliver real-time recommendations, personalization, and improved user experiences with NoSQL databases. For example, Disney+ delivers its extensive digital content library to over 150 million+ subscribers using NoSQL database technology. It can scale and deliver popular features such as Continue Watching, Watchlist, and Personalized Recommendations with Amazon DynamoDB.

### Cloud security

You can use graph databases to quickly discover complex relationships within your data. For instance, Wiz re-imagined cloud security as a graph using Amazon Neptune. Wiz helps their customers improve their security posture by quickly identifying and fixing the most critical risks. They use graph model stored in Amazon Neptune to uncover the toxic combination of risk factors that represent critical risks. The Wiz risk engines traverse the graph and within seconds, weave together a series of interconnected risks factors in a security graph.

### High-availability applications

Distributed NoSQL databases are excellent for building high-availability, low-latency applications for messaging, social media, file sharing, and more. For example, Snapchat has more than 290 million users sending billions of pictures and video messages daily. It uses NoSQL database systems to reduce the median latency of sending messages by 20%.

## How do NoSQL databases work

NoSQL databases use a variety of data models for accessing and managing data. These types of databases are optimized specifically for applications that require flexible data models, large data volume, and low latency, which are achieved by relaxing some of the data consistency restrictions of relational databases. There are differences in implementation based on the data model. However, many NoSQL databases use Javascript Object Notation (JSON), an open data interchange format that represents data as a collection of name-value pairs.

tables, and relationships are defined by primary and foreign key constraints. In this example, the **Books** table has columns for **ISBN**, **Book Title**, and **Edition Number**; the **Authors** table has columns for **AuthorID** and **Author Name**; and finally, the **Author-ISBN** table has columns for **AuthorID** and **ISBN**. The relational model is designed to enable the database to enforce referential integrity between tables in the database, normalized to reduce the redundancy, and generally optimized for storage.

- In a NoSQL database, a book record is usually stored as a document. For each book, the item, **ISBN**, **Book Title**, **Edition Number**, **Author Name**, and **AuthorID** are stored as attributes in a single document. In this model, data is optimized for intuitive development and horizontal scalability.

### SQL vs. NoSQL terminology

The following table compares terminology used by select NoSQL databases with terminology used by SQL databases.

| SQL | MongoDB | DynamoDB | Cassandra | Couchbase |
|---|---|---|---|---|
| Table | Collection | Table | Table | Data bucket |
| Row | Document | Item | Row | Document |
| Column | Field | Attribute | Column | Field |
| Primary key | ObjectId | Primary key | Primary key | Document ID |
| Index | Index | Secondary index | Index | Index |
| View | View | Global secondary index | Materialized view | View |
| Nested table or object | Embedded document | Map | Map | Map |
| Array | Array | List | List | List |

## What are the types of NoSQL databases

There are several different NoSQL database systems due to variations in the way they manage and store schema-less data. We explain some of the common types below.

### Key-value databases

millisecond latency for any scale of workloads.

## Document databases

Document databases have the same document model format that developers use in their application code. They store data as JSON objects that are flexible, semi-structured, and hierarchical in nature.. The flexible, semistructured, and hierarchical nature of documents and document databases allows them to evolve with applications' needs. The document database model works well with catalogs, user profiles, and content management systems, where each document is unique and evolves over time. Amazon DocumentDB (with MongoDB compatibility) and MongoDB are popular document databases that provide powerful and intuitive APIs for flexible and iterative development.

## Graph databases

Graph databases are purpose-built to make it easy to build and run applications that work with highly connected datasets. They use nodes to store data entities and edges to store relationships between entities. An edge always has a start node, end node, type, and direction. It can describe parent-child relationships, actions, ownership, and the like. There is no limit to the number and kind of relationships a node can have. You can use a graph database to build and run applications that work with highly connected datasets. Typical use cases for a graph database include social networking, recommendation engines, fraud detection, and knowledge graphs. Amazon Neptune is a fully-managed graph database service supporting both the Property Graph model and the Resource Description Framework (RDF) with the choice of two graph APIs (TinkerPop and RDF/SPARQL).

## In-memory databases

While other non-relational databases store data on disk or SSDs, in-memory data stores are designed to eliminate the need to access disks. They are ideal for applications that require microsecond response times or have large spikes in traffic. You can use them in gaming and ad-tech applications for features like leaderboards, session stores, and real-time analytics. Amazon MemoryDB for Redis is a Redis-compatible, durable, in-memory database service that delivers microsecond read latency, single-digit millisecond write latency, and Multi-AZ durability. Amazon ElastiCache is a fully managed, in-memory caching service compatible with both Redis and Memcached, to serve low-latency, high-throughput workloads. Amazon DynamoDB Accelerator (DAX) is another example of a purpose-built data store that makes DynamoDB reads an order of magnitude faster.

## Search databases

A search-engine database is a type of non-relational database that is dedicated to the search of data content, such as application output logs used by developers to troubleshoot issues. They use indexes to categorize similar characteristics among data and facilitate search capability. Search-engine databases are optimized for

# What are the differences between NoSQL and SQL databases

For decades, the predominant data model in application development was the relational data model that stored data in tables made of rows and columns. Structured Query Language (SQL) was used to create and edit these relational tables. SQL databases model data relationships as tables. The rows in the table represent a collection of related values of one object or entity. Each column in the table represents a data attribute, and a field (or table cell) stores the actual value of the attribute. You can use a relational database management system (RDBMS) to access the data in many different ways without reorganizing the database tables themselves.

It wasn't until the mid to late 2000s that other flexible data models began to gain significant adoption and usage. To differentiate and categorize these new classes of databases and data models, the term NoSQL was coined. NoSQL stands for not only SQL or non-SQL. Often the term NoSQL is used interchangeably with the term non-relational. Key differences between relational and non-relational databases are given in the table below.

| | Relational databases | NoSQL databases |
|---|---|---|
| Optimal workloads | Relational databases are designed for transactional and strongly consistent online transaction processing (OLTP) applications. They are also good for online analytical processing (OLAP). | NoSQL databases are designed for a number of data access patterns that include low-latency applications. NoSQL search databases are designed for analytics over semi-structured data. |
| Data model | The relational model normalizes data into tables that are composed of rows and columns. A schema strictly defines the tables, rows, columns, indexes, relationships between tables, and other database elements. The database enforces referential integrity in relationships between tables. | NoSQL databases provide a variety of data models, such as key-value, document, graph, and column, which are optimized for performance and scale. |
| ACID properties | Relational databases provide atomicity, consistency, isolation, and durability (ACID) properties: | Most NoSQL databases offer trade-offs by relaxing some of the ACID properties of relational databases in favor of a more flexible data model that can scale |

| | | |
|---|---|---|
| | committed.<br>• Isolation requires that concurrent transactions execute separately from each other.<br>• Durability requires the ability to recover from an unexpected system failure or power outage to the last known state. | |
| Performance | Performance is generally dependent on the disk subsystem. The optimization of queries, indexes, and table structure is often required to achieve peak performance. | Performance is generally a function of the underlying hardware cluster size, network latency, and the calling application. |
| Scale | Relational databases typically scale up by increasing the compute capabilities of hardware or scale out by adding replicas for read-only workloads. | NoSQL databases are typically partitionable. This is because access patterns can scale out by using distributed architecture to increase throughput that provides consistent performance at near-boundless scale. |
| APIs | Requests to store and retrieve data are communicated using queries that conform to a structured query language (SQL). These queries are parsed and executed by the relational database. | Object-based APIs allow app developers to easily store and retrieve data structures. Partition keys let apps look up key-value pairs, column sets, or semi-structured documents that contain serialized app objects and attributes. |

## When should you choose NoSQL databases over SQL databases

A NoSQL database is best for handling indeterminate, unrelated, or rapidly changing data. It is intuitive to use for developers when the application dictates the database schema. You can use it for applications that:

- Need flexible schemas that enable faster and more iterative development.

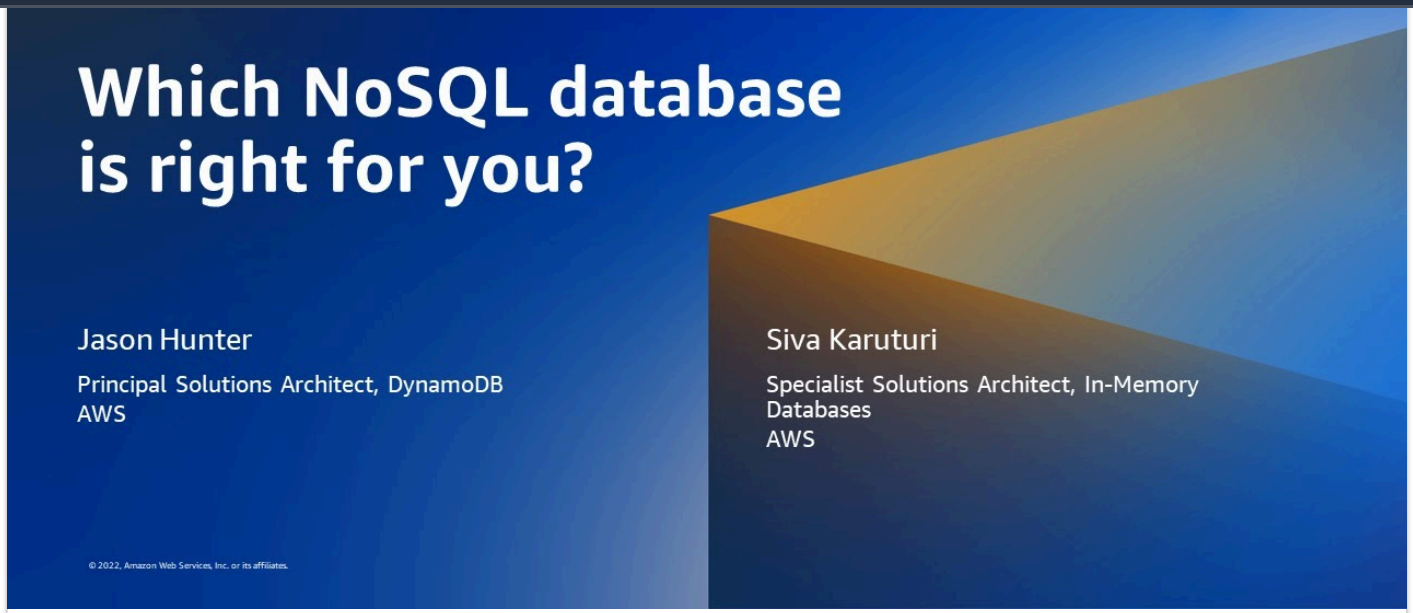
You don't always have to choose between a non-relational and relational database schema. You can employ a combination of SQL and NoSQL databases in your applications. This hybrid approach is quite common and ensures each workload is mapped to the right database for optimal price performance.

## How can AWS support your NoSQL database requirements

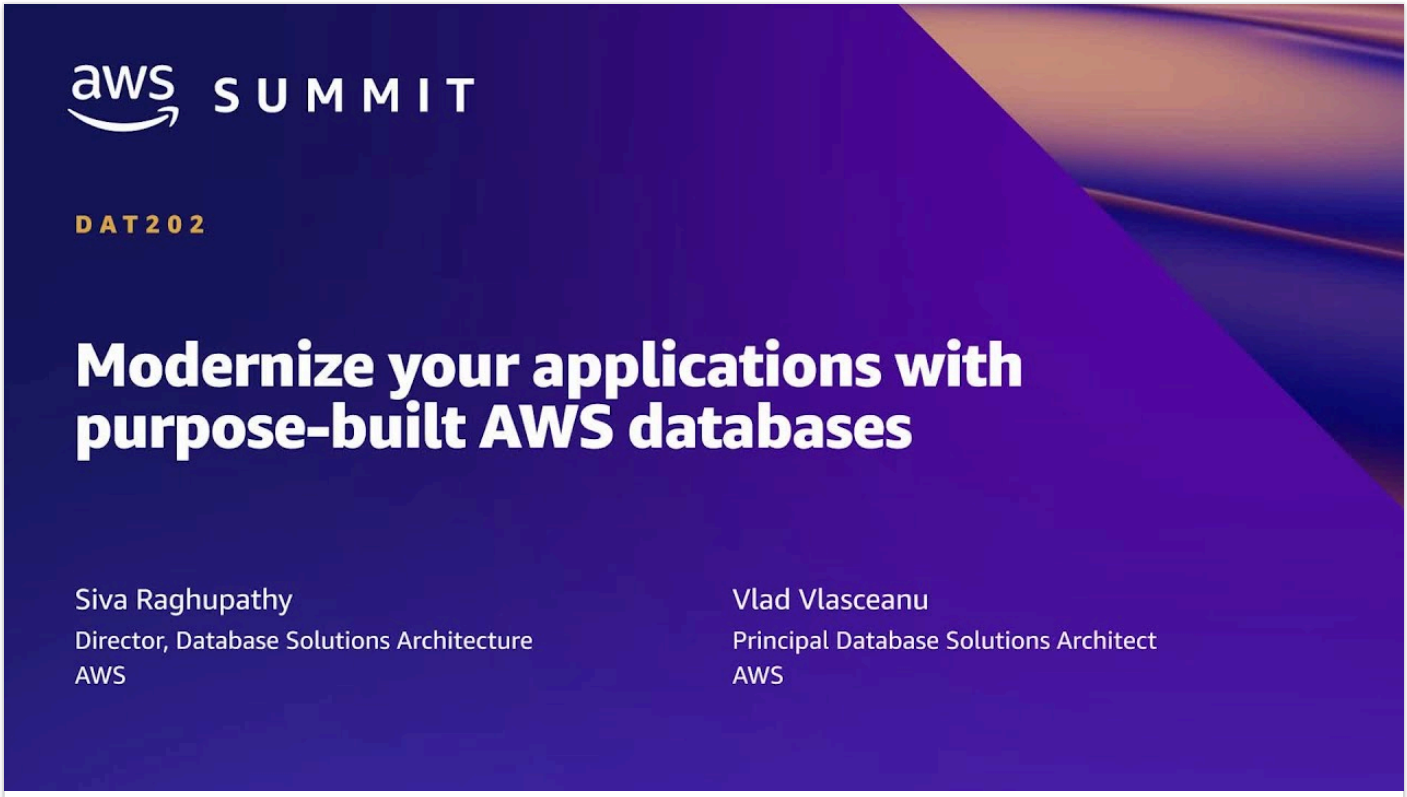
AWS has several NoSQL database services to meet all your NoSQL requirements. For example:

- Amazon DynamoDB is a serverless, fully managed, key-value database service that provides consistent, single-digit-millisecond performance with limitless scalability.
- Amazon DocumentDB (with MongoDB compatibility) is a fully managed, native JSON document database that makes it easy and cost effective to operate critical document workloads at virtually any scale without managing infrastructure.
- Amazon Neptune is a serverless, fully managed graph database service designed for superior scalability and availability with ability to query billions of relationships in seconds.
- Amazon MemoryDB for Redis is a durable, in-memory database service that delivers microsecond read and write response times for ultra-fast performance.
- Amazon ElastiCache is a fully managed, Redis- and Memcached-compatible, in-memory data store and cache service that delivers real-time, cost-optimized performance.
- Amazon Keyspaces (for Apache Cassandra) is a serverless, fully managed wide-column database designed for up to 99.999% availability with multi-Region replication.
- Amazon Timestream is a serverless, fully managed time-series database that makes it easier to store and analyze trillions of events per day up to 1,000 times faster than relational databases.
- Amazon OpenSearch Service is fully managed distributed search and analytics suite that enables real-time search, monitoring, and analysis of business and operational data.

Get started with NoSQL on AWS by creating a free account today!




Which NoSQL database is right for you?




Modernize your applications with purpose-built databases



Gain free, hands-on experience with AWS for 12 months

**Get Your AWS Certification**

Grow in-demand skills, validate your expertise, and advance your career. Learn about AWS Certification and the resources to help you succeed »

Get Started for Free

Contact Us

Create an AWS Account

Amazon is an Equal Opportunity Employer: *Minority / Women / Disability / Veteran / Gender Identity / Sexual Orientation / Age.*

**Language**

عربي |
Bahasa Indonesia |
Deutsch |
English |
Español |
Français |
Italiano |
Português |
Tiếng Việt |
Türkçe |
Русский |
ไทย |
日本語 |
한국어 |
中文 (简体) |
中文 (繁體)

Privacy
|
Site Terms
|
Cookie Preferences
|
© 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.