

BLOG | COMPANY

SQL im Vergleich zu NoSQL: Vorteile und Nachteile

Updated: May 29, 2025 · 9 min read



Was ist die beste Methode, um Ihre Daten zu speichern, zu schützen und darauf zuzugreifen? Das ist eine grundlegende, aber überaus wichtige Frage. Schließlich sind Daten für nahezu jedes Unternehmen von heute der Eckpfeiler des Erfolgs. Bei den meisten Unternehmen fällt die Wahl auf SQL- und NoSQL-Datenbanken. Jeder der beiden Typen hat seine eigenen Stärken und Schwächen.

SQL-Datenbanken haben sich seit den 1970er Jahren bewährt. Sie bestehen aus stark strukturierten Tabellen, die Zeilen und Spalten enthalten und durch gemeinsame Attribute miteinander verknüpft sind. Jede Spalte muss

einen Wert für die ihr entsprechende Zeile haben. NoSQL- ▲ ▲ ▲ Datenbanken („Not only SQL“ oder „Non-SQL“) wurden später entwickelt, um das starre Konzept der relationalen Tabellen aufzubrechen, mit der Fähigkeit, alle Datentypen, strukturiert und unstrukturiert, zusammen zu speichern und aufzurufen. Sie sind äußerst flexibel und für Entwickler einfach zu bearbeiten und zu ändern. Erfahren Sie mehr über [SQL- und NoSQL-Datenbanken und ihre grundlegenden Unterschiede](#).

Was ist nun die optimale Wahl? Das hängt von einer ganzen Reihe von Faktoren ab, einschließlich Ihrer Abfrage-, Verfügbarkeits- und Konformitätsanforderungen sowie der Vielfalt Ihrer Datentypen und des erwarteten Datenwachstums.

Vergleichsdiagramm SQL vs. NoSQL



Schauen wir uns die Vor- und Nachteile von SQL und NoSQL einmal genauer an, um Sie bei der richtigen Wahl zu unterstützen.

Vorteile von SQL

Standardisiertes Schema



Obwohl das standardisierte Schema von SQL-Datenbanken sie unflexibel macht und Änderungen erschwert, hat es durchaus einige Vorteile. Alle Daten, die der Datenbank hinzugefügt werden, müssen dem bekannten Schema der verknüpften Tabellen aus Zeilen und Spalten entsprechen. Dies mögen manche als einschränkend empfinden, aber es ist hilfreich, wenn Konsistenz, Integrität, Sicherheit und Konformität der Daten an erster Stelle stehen.

Große Benutzer-Community

Mit einem Alter von fast 50 Jahren ist die Programmiersprache SQL extrem ausgereift und immer noch weit verbreitet. Hinter SQL steht eine starke Community mit unzähligen Experten, die bereit sind, Tipps und etablierte Best Practices auszutauschen. Es gibt viele Möglichkeiten, Fähigkeiten zu erweitern und zusammenzuarbeiten. Bei Bedarf können Berater und SQL-Anbieter zusätzliche Unterstützung bieten. Mit SQL finden Ihre Entwickler die Antworten, die sie brauchen.

Kein Code erforderlich

SQL ist eine benutzerfreundliche Sprache. Das Verwalten und Abfragen der Datenbank ist mittels einfacher Schlüsselwörter möglich, wobei wenig bis kein Coding erforderlich ist. Die meisten Entwickler lernen SQL schon während ihres Studiums.

ACID-Konformität

Dank der extrem strukturierten Natur relationaler Datenbanktabellen können SQL-Datenbanken ACID-kompatibel sein. Dieses Maß an Konformität sorgt dafür, dass die Tabellen synchron bleiben, und garantiert die Gültigkeit von Transaktionen. SQL ist wahrscheinlich die richtige Wahl, wenn Sie Anwendungen ausführen, die unbedingt fehlerfrei bleiben müssen und ein Höchstmaß an Datenintegrität benötigen.

Hier sind die ACID-Eigenschaften:

- Atomarität: alle Änderungen an Daten und Transaktionen werden vollständig und in einem Vorgang durchgeführt. Ist dies nicht möglich, wird keine der Änderungen durchgeführt. Es ist ein Alles-oder-Nichts-Konzept.
- Konsistenz: Die Daten müssen zu Beginn und am Ende einer Transaktion gültig und konsistent sein.
- Isolation: Transaktionen laufen gleichzeitig, ohne miteinander zu konkurrieren. Stattdessen verhalten sie sich so, als würden sie nacheinander auftreten.
- Dauerhaftigkeit: Wenn eine Transaktion abgeschlossen ist, sind die zugehörigen Daten dauerhaft und können nicht geändert werden.

Nachteile von SQL

Hardware

Relationale Datenbanken werden in der Regel vertikal skaliert. Die Kapazität lässt sich dabei nur durch eine Erhöhung der Hardware-Leistung, z. B. bei RAM, CPU und SSD, auf dem vorhandenen Server oder durch Migration auf einen größeren, teureren Server erweitern. Sie müssen den Festplattenspeicher kontinuierlich erweitern, wenn Ihr Datenbestand wächst, und Sie benötigen schnellere Maschinen, um neue und leistungsfähigere Technologien zu nutzen. Ihr Datenbankanbieter wird wahrscheinlich verlangen, dass Sie Ihre Hardware regelmäßig aufrüsten, nur um die neuesten Versionen ausführen zu können. In diesem Umfeld kann die Hardware schnell veralten. Jedes Upgrade ist mit hoher Wahrscheinlichkeit teuer und ressourcenintensiv. Der Hardwarebedarf von SQL umfasst auch laufende, tägliche Wartungs- und Betriebskosten. Es ist praktisch eine unendliche Geschichte.

Datennormalisierung

Da sie zu einer Zeit entwickelt wurden, als die Kosten für die Datenspeicherung hoch waren, versuchen relationale Datenbanken, Datenduplizierung zu vermeiden. Jede Tabelle enthält unterschiedliche Informationen und sie können über gemeinsame Werte verknüpft und abgefragt werden. Wenn SQL-Datenbanken jedoch größer werden, können die zwischen zahlreichen Tabellen erforderlichen Suchvorgänge und Verknüpfungen die Prozesse verlangsamen.



Starrheit

Das Schema einer SQL-Datenbank muss vor dem Einsatz definiert werden. Nach erfolgter Einrichtung sind sie unflexibel und Änderungen sind in der Regel schwierig und ressourcenintensiv. Aus diesem Grund muss am Anfang viel Zeit in die Planung investiert werden, bevor die Datenbank überhaupt in Betrieb genommen wird. SQL-Datenbanken sind somit nur geeignet, wenn alle Ihre Daten auch strukturiert sind und Sie keine großen Änderungen erwarten, weder was das Volumen noch die Datentypen betrifft.

Ressourcenintensive Skalierung

Wie bereits erwähnt, werden SQL-Datenbanken in der Regel vertikal skaliert, indem die Hardwareinvestitionen erweitert werden. Dies ist teuer und zeitaufwendig. In manchen Fällen versuchen Organisationen, ihre SQL-Datenbank durch Partitionierung horizontal zu skalieren. Die zusätzliche Komplexität erhöht hier noch den Zeit- und Ressourcenaufwand. Dabei wird wahrscheinlich auch Coding anfallen, was hochqualifizierte, hochbezahlte Entwickler erfordert. Wenn das Datenvolumen immer weiter wächst, ist die Skalierung Ihrer SQL-Datenbank ein endloses Unterfangen, bei dem das perfekte Setup immer gerade außer Reichweite ist. Dagegen lassen sich NoSQL-Datenbanken horizontal skalieren, wodurch die Kapazität einfacher und kostengünstiger erweitert werden kann. Sie sind bestens geeignet für Cloud Computing und die Handhabung extrem großer und schnell wachsender Datenbestände.

Kontinuierliche Verfügbarkeit

Bei NoSQL werden Daten über mehrere Server und Regionen hinweg verteilt, sodass es keinen Single Point of Failure gibt. Infolgedessen sind NoSQL-Datenbanken stabiler und resilenter und bieten kontinuierliche Verfügbarkeit sowie null Ausfallzeiten.

Abfragegeschwindigkeit

Da NoSQL-Datenbanken denormalisiert sind und Datenduplizierung kein Problem ist, werden alle Informationen, die für eine bestimmte Abfrage benötigt werden, oft bereits zusammen gespeichert – es sind keine Verknüpfungen erforderlich. Dies kann Suchen erleichtern, insbesondere bei der Arbeit mit großen Datenmengen, und es bedeutet auch, dass NoSQL bei einfachen Abfragen sehr schnell sein kann. Dabei ist es keineswegs so, dass SQL-Datenbanken nicht auch sehr schnelle Abfragen durchführen können. Sie unterstützen zudem hochkomplexe Abfragen für strukturierte Daten. Die Abfragegeschwindigkeit kann jedoch schnell nachlassen, wenn SQL-Datenbanken wachsen und komplexe Verknüpfungsanforderungen zunehmen.

Agilität

NoSQL-Datenbanken wurden entwickelt, als die Kosten für die Datenspeicherung drastisch zu sinken begannen und die Entwicklerkosten stiegen. Datenduplizierung stellte kein Problem mehr dar. Stattdessen sollte ihr Design Entwicklern möglichst viel Flexibilität bieten, um die Kreativität und Produktivität steigern zu können. Da NoSQL-Datenbankschemata nicht durch Zeilen und Spalten eingeschränkt sind, müssen sie nicht vordefiniert werden. Stattdessen sind sie dynamisch und können alle Arten von Daten handhaben, einschließlich strukturierter, halbstrukturierter, unstrukturierter und polymorpher Daten. Sie können mit NoSQL-Datenbanken loslegen,

ohne Zeit für das Definieren ihrer Struktur aufzuwenden, und Datentypen und Felder einfach und ohne Ausfallzeiten im Handumdrehen hinzufügen. All dies macht NoSQL zu einer großartigen Lösung für moderne, agile Entwicklungsteams. Entwickler können einsteigen und mit dem Aufbau einer Datenbank beginnen, ohne Zeit und Mühe in die Planung investieren zu müssen. Wenn sich die Anforderungen ändern und neue Datentypen hinzugefügt werden, können sie schnell Änderungen vornehmen. Die Flexibilität und Anpassungsfähigkeit von NoSQL-Datenbanken machen sie zu einer großartigen Lösung für Organisationen, die über eine Vielzahl von Datentypen verfügen und fortlaufend neue Features und Funktionen hinzufügen.

NoSQL-Datenbanken sind nicht universell, sondern individuell. Im Gegensatz zu SQL-Datenbanken sind sie nicht auf ein starres, zentralisiertes Datenmodell beschränkt, das wahrscheinlich auf einem einzelnen Server untergebracht ist. Stattdessen bietet NoSQL die Flexibilität, unterschiedliche Datenbankmodelltypen zu verbinden, die auf viele Server verteilt sein können. NoSQL umfasst mehrere Datenbanktypen, sodass Entwickler die für ihre Daten und Anwendungsfälle am besten geeignete Mischung finden können. Die Haupttypen von NoSQL-Datenbanken sind Key/Value, dokumentenorientiert, tabellarisch (oder Wide-Column), Graph oder Multi-Modell.

Niedrige Kosten

NoSQL-Datenbanken können horizontal skaliert werden, sodass sich ihre Kapazität kostengünstig erweitern lässt. Anstatt teure Hardware aufrüsten zu müssen, können einfach handelsübliche Server oder Cloud-Instanzen hinzugefügt werden. Und Open-Source-Versionen von NoSQL-Datenbanken bieten vielen Unternehmen erschwingliche Optionen. Sie sind bestens geeignet für Cloud Computing und die Handhabung extrem großer und schnell wachsender Datenbestände.

Keine standardisierte Sprache

Es gibt keine Standardsprache zum Durchführen von NoSQL-Abfragen. Die Syntax zum Abfragen von Daten variiert je nach dem Typ der NoSQL-Datenbank. Im Gegensatz zu SQL, wo man nur eine einfach zu erlernende Sprache beherrschen muss, hat NoSQL eine steilere Lernkurve. Beispielsweise kann es für einen Entwickler schwierig sein, sich schnell in eine Wide-Column-Datenbank einzuarbeiten, wenn er sich bisher nur mit dem Erstellen und Verwalten von Graphdatenbanken befasst hat.

Kleine Benutzer-Community

Entwickler arbeiten seit mehr als einem Jahrzehnt mit NoSQL-Datenbanken und die Community wächst schnell. Da sie aber jünger als die SQL-Community ist, könnte es schwieriger sein, nicht dokumentierte Probleme zu lösen. Außerdem gibt es auf der NoSQL-Seite weniger Berater und Experten.

Ineffizienz bei komplexen Abfragen

Die Flexibilität hat ihren Preis. Aufgrund der Vielzahl der Datenstrukturen, die in NoSQL-Datenbanken zu finden sind, ist die Durchführung von Abfragen weniger effizient. Anders als bei SQL-Datenbanken gibt es keine Standardschnittstelle zur Durchführung komplexer Abfragen. Selbst für einfache NoSQL-Abfragen ist wahrscheinlich Programmiererfahrung erforderlich. Die Abfragen müssen daher von technisch versierteren und teureren Mitarbeitern wie Entwicklern oder Datenexperten durchgeführt werden.

Inkonsistenz beim Datenabruf

Die verteilte Natur von NoSQL-Datenbanken ermöglicht eine schnellere Verfügbarkeit der Daten. Allerdings kann es dadurch auch schwieriger sein, sicherzustellen, dass die Daten immer konsistent sind. Abfragen geben möglicherweise nicht immer aktualisierte Daten zurück und es besteht das Risiko, ungenaue Informationen zu erhalten. Wegen ihres verteilten Ansatzes könnte die Datenbank gleichzeitig unterschiedliche Werte zurückgeben, je nachdem, welcher Server gerade abgefragt wird. Dies ist einer der Gründe, warum NoSQL keine ACID-Konformität erreicht. Das „C“ in ACID steht für Consistency bzw. Konsistenz. Diese Anforderung besagt, dass Daten zu Beginn und am Ende einer Transaktion gültig und konsistent sein müssen. Stattdessen richten sich die meisten NoSQL-Datenbanken nach dem BASE-Konsistenzmodell, wobei das „E“ für Eventual Consistency (letztendliche Konsistenz) steht. Mit anderen Worten, die Daten werden zu einem späteren Zeitpunkt konsistent sein. In der Praxis ist dies häufig eine kleine Verzögerung von nur wenigen Millisekunden. Für viele Anwendungen spielt das wahrscheinlich keine Rolle, z. B. wenn Social-Media-Beiträge veröffentlicht werden oder ein Online-Warenkorb aktualisiert wird. In diesen Situationen wiegt eine schnellere Verfügbarkeit für den größten Teil des Netzwerks den Nutzen auf, dass allen Benutzern zur gleichen Zeit exakt dieselben Daten zur Verfügung gestellt werden. In einigen Fällen kann dies jedoch durchaus von Bedeutung sein, beispielsweise, wenn Sie online einen Aktienkauf tätigen. Bei NoSQL haben Geschwindigkeit und Verfügbarkeit Vorrang vor der Konsistenz. Jedes Unternehmen muss selbst entscheiden, ob dies mit seinen Zielen konform geht.



Optionen abwägen

Sowohl SQL- als auch NoSQL-Datenbanken sind für spezielle Anforderungen und Anwendungsfälle bestens geeignet. Abhängig von der Datenumgebung und den Zielen Ihres Unternehmens können sich die jeweiligen Vor- und Nachteile verstärkt auswirken. Vielleicht stellen Sie auch fest, dass die beste Lösung darin besteht, beide einzusetzen und jeden Datenbanktyp seine Stärken

ausspielen zu lassen. Viele Unternehmen verwenden sowohl SQL- als auch NoSQL-Datenbanken in ihrer Cloud-Architektur, manchmal sogar innerhalb derselben Anwendung. Auch hier könnte die beste Option darin bestehen, eine Lösung wie DataStax Astra DB zu finden, die die inhärenten Vorteile von NoSQL wie Flexibilität, kontinuierliche Verfügbarkeit und Skalierbarkeit nutzt und gleichzeitig seine Nachteile minimiert.

Astra ist eine Multi-Cloud-DBaaS-Lösung („Database-as-a-Service“) auf der Basis von Apache Cassandra und Kubernetes mit einer Microservices-Architektur. Sie können nach nur wenigen Klicks in der Cloud Ihrer Wahl (Azure, Google Cloud Platform oder AWS) einsatzbereit sein. Einmal dort eingerichtet, vereinfacht die Lösung die Anwendungsentwicklung drastisch. Astra integriert Stargate, eine Open-Source-Daten-API-Schicht, die Treiber aus der Gleichung entfernt und über die Sie Ihre Daten abfragen oder Tabellen und Schemata erstellen können, ohne die Cassandra Query Language (CQL) erlernen zu müssen. Stargate ermöglicht Ihnen die Interaktion mit Daten mittels moderner APIs, einschließlich schemlosem JSON, REST und GraphQL, und mit Storage-Attached-Indexing (SAI) von Astra können Sie jede Spalte in der Tabelle abfragen, ohne auf den Primärschlüssel beschränkt zu sein.

[Schauen Sie sich Astra im Detail an.](#)

DataStax AI Platform:

The Fastest Way to Build and Deploy AI Apps

TRY FOR FREE



Rich Edwards



More Company

[View All](#)

The card has a dark background with a subtle grid pattern. In the top left corner, the DataStax logo is displayed with the text "an IBM Company" underneath. To the right of the logo is a screenshot of a mobile application interface. The interface includes a "watsonx" header, a text input field with placeholder text "Define the assistant's role and capabilities", a "Temperature" slider set to "Precise", and a "Top P" button. Below this is another card labeled "Astra DB" with the subtext "Search, retrieve, and manage embeddings." and a "Astra DB Application Token" input field.

DATASTAX
an IBM Company

IBM Acquires DataStax,
Accelerating Production AI
& NoSQL Data at Scale

COMPANY

MAY 28, 2025

DataStax Officially Joins IBM

Chet Kapoor



COMPANY

APRIL 30, 2025

DataStax Joins the OpenSearch Software Foundation

Kiyu Gabriel



COMPANY

APRIL 17, 2025

DataStax and Google Cloud: Partners in AI, Data, and Innovation

Tina Brown

COMPANY

[About Us](#)
[Leadership](#)
[Board of Directors](#)
[Contact Us](#)
[Partners](#)
[Careers](#)
[Newsroom](#)

[Apache Cassandra® vs DataStax](#)

RESOURCES

[Docs](#)
[Blog](#)
[Events](#)
[Meet Bodi](#)
[Brand Resources](#)
[Contact Support](#)
[Security](#)
[What is a Vector Database?](#)
[What is Retrieval-Augmented Generation \(RAG\)?](#)

CLOUD PARTNERS

[Amazon Web Services](#)
[Google Cloud](#)
[Microsoft Azure](#)
[NVIDIA](#)

Subscribe to AI++

A newsletter for devs building apps using AI

Work Email



© 2025 DataStax

[Privacy Policy](#) | [Terms of Use](#) | [Trademark Notice](#) | [Legal](#) | [Manage Privacy Choices](#)

Germany