# Von relationalen Datenbanken zu NoSQL

Die von Don Chamberlin und Ray Boyce bei IBM erfundene Structured Query Language (SQL) ist die Standardprogrammiersprache für die Interaktion mit relationalen Datenbankverwaltungssystemen. Sie ermöglicht das Ausführen von Transaktionen wie beispielsweise dem Hinzufügen, Aktualisieren oder Löschen von Datenzeilen sowie dem Zusammenfügen von mehreren Tabellen über Primärschlüssel auf relationalen Datenbanken [1]. Eine relationale Datenbank speichert Daten dabei grundsätzlich immer in tabellarischer Form.

Ein großer Vorteil von relationalen Datenbanken ist es, dass diese die sogenannten ACID-Eigenschaften einhalten können. ACID steht für die englischen Begriffe atomicity, consistency, isolation und durability: Datentransaktionen sollen entweder ganz oder gar nicht ausgeführt werden und sind für andere Benutzer nach Abschluss der Transaktion sichtbar (Atomarität / atomicity). Vor und nach dem Ausführen der Transaktion sind alle Integritätsbedingungen erfüllt, die Daten sind also weiterhin nach zuvor definierten Regeln gültig (Konsistenz / consistency). Wenn mehrere Transaktionen gleichzeitig stattfinden, muss der Endzustand derselbe sein, als wenn die Transaktionen getrennt voneinander ausgeführt worden wären (Isolation / isolation). Und die Daten dürfen sich nur durch Transaktionen verändern, nicht aber durch äußere Einflüsse wie beispielsweise einem Stromausfall (Dauerhaftigkeit / durability). Die ACID-Eigenschaften gelten als Voraussetzung für die Verlässlichkeit von Systemen und sind ein Grund für den Erfolg relationaler Datenbanken.

Dem entgegen steht das Skalierungsproblem relationaler Datenbanken: Dem Scheitern des relationalen Systems, auch bei sehr großen Datenmengen Transaktionen effizient und schnell auszuführen:

Grundsätzlich gibt es drei Ansätze, um dem Skalierungsproblem entgegenzuwirken: Durch Effizienzsteigerungen der Datenbank, sowie durch senkrechte und horizontale Skalierbarkeit. Beispielhafte Ansätze der Effizienzsteigerung der Datenbank sind das Speichern häufig abgefragter Daten im Zwischenspeicher (Caching) oder das Auslagern selten genutzter Daten aus der aktiven Datenbank (Archivierung). Senkrechte Skalierbarkeit bezieht sich auf die Steigerung der Leistungsfähigkeit eines einzelnen Servers, beispielsweise durch CPU-, RAM- oder Speichererweiterungen. Dem entgegen steht die horizontale Skalierbarkeit: Hierbei wird die Leistung eines Systems durch das Hinzufügen von mehreren Knoten (Servern) zu einem Cluster erweitert. Jeder Knoten im Cluster kann Anfragen verarbeiten und die Last wird auf alle Knoten im Cluster verteilt. Wenn die Anforderungen an das System steigen, können zusätzliche Knoten zum Cluster hinzugefügt werden, um die Leistung zu verbessern und die Last weiter zu verteilen.

Relationale Datenbanken sind dabei nicht auf die horizontale Skalierbarkeit als einzig endlose Skalierungsmöglichkeit ausgelegt: Beim Kopieren auf Knoten muss hier die gesamte Datenstruktur übernommen werden. Dies ist notwendig, um die Konsistenz der Daten über alle Knoten hinweg sicherzustellen und sicherzustellen, dass Anfragen an den neuen Knoten dieselben Ergebnisse liefern wie Anfragen an die anderen Knoten im Cluster. Transaktionen wie das Hinzufügen einer neuen Tabellenspalte zu der relationalen Datenbank werden mit steigender Anzahl der Knoten also aufwendiger, da alle Knoten die daraus resultierende neue Struktur übernehmen müssen.

Große Unternehmen wie Facebook, Amazon und Twitter stoßen mit ihren Datenmengen an solche Skalierungsgrenzen relationaler Datenbanken und weichen deshalb auf die Alternative der NoSQL-Datenbanken, also nichtrelationalen Datenbanken aus (not only SQL).

NoSQL: Horizontale Skalierbarkeit, aber kein ACID.

Arten von NoSQL-Datenbanken

CAP-Theorem: relational=CA, NoSQL=AP

Bei Injektionsangriffen fügt ein Angreifer bösartigen Code in die Anweisungen ein, die von der Anwendung an die Datenbankschicht weitergegeben werden. Dies kann beispielsweise über Suchfelder, Formularfelder oder API-Eingaben erfolgen. Wenn die Anwendung solche Eingaben nicht bereinigt, kommt es zur Akzeptanz und Ausführung des Codes. Der Angreifer hat dann fast unbegrenzte Kontrolle über die zu Grunde liegende Datenbank: Er kann Daten verändern, einfügen oder löschen und hat Zugriff auf nichtautorisierte Daten [2]. Je nach Datenbank wird hierbei zwischen SQL- und NoSQL-Injektionsangriffen unterschieden. Die NoSQL-Injektion ist ein spezifischer Typ von Injektionsangriff, der nur auf NoSQL-Datenbanken ausgeführt werden kann.

[1] Chamberlin, D. D. (2012). *Early History of SQL. IEEE Annals of the History of Computing, 34(4), 78–82.* doi:10.1109/mahc.2012.61

[2] Ron, A., Shulman-Peleg, A., & Puzanov, A. (2016). *Analysis and Mitigation of NoSQL Injections. IEEE Security & Privacy, 14(2), 30–39*. doi:10.1109/msp.2016.36

[3] Schwarze Consulting GmbH. (2015). *Übersicht relational und/oder noSQL (I/II).* https://databasecamp.de/daten/acid