entwickler.press
shortcuts

NoSQL-Überblick

Elasticsearch und Redis

Eberhard Wolff, Christian Meder,
Bernhard Pflugfelder

Eberhard Wolff, Christian Meder, Bernhard Pflugfelder

NoSQL-Überblick

Elasticsearch und Redis

ISBN: 978-3-86802-503-3

Ein Imprint der Software & Support Media GmbH

1 Wie können Datenbanken verglichen werden?

NoSQL führt eine Vielzahl neuer Technologien in die Datenbankszene ein. Dieser shortcut wird einen genauen Blick darauf werfen, was es alles gibt und wie sich die Datenbanken unterscheiden.

Ein technischer Überblick über die Konzepte und Modelle von NoSQL-Datenbanken findet sich schon unter [1]. Wir fassen die Kernpunkte des Kapitels zusammen:

- Als Gründe für das Entstehen der NoSQL-Bewegung werden oft das Wachstum der Datenmengen und die höheren Anforderungen an Performance vor allem aus dem Webbereich angeführt. Das ist sicher richtig, aber nur einer der Treiber. Bei der ersten Generation von Datenbanken haben sich nämlich die Internetriesen genau dem Problem der großen Datenmengen angenommen: Google mit BigTable, Amazon mit Dynamo und Facebook mit Cassandra.
- Ein weiterer Treiber ist die Struktur der Daten: User-generated Content aus Social Networks ist nur wenig strukturiert und kann daher nur schwer in einer relationalen Datenbank mit festem Schema gespeichert werden. Aber auch Datenmodelle aus dem klassischen IT-Bereich beispielsweise für Finanzprodukte, Versicherungsverträge usw. sind sehr komplex – und können mit bestimmten NoSQL-Datenbanken einfacher bearbeitet werden. Einige vernetzte Strukturen, z. B. die Beziehungen in sozialen Netzwerken wie Facebook oder auch Verkehrs- oder Telekommunikationsnetze, sind mit relationalen Ansätzen nur mit sehr schlechter Performance abbildbar. Auch hier können NoSQL-Datenbanken helfen, und zwar vor allem die Graphendatenbanken.

- Ein weiteres Problem ist die Flexibilität: Schemaänderungen bei relationalen Datenbanken sind insbesondere dann schwierig, wenn bereits große Datenmengen vorhanden sind. Durch Continuous Delivery werden viel öfter neue Softwareversionen mit ihren dazugehörigen Schemaänderungen deployt. So kann die Datenbank zu einem Flaschenhals für Änderungen an der Software werden. Um mit diesem Problem umzugehen, gibt es zahlreiche Workarounds – oft sind das interessante, manchmal aber auch einfach nur lustige Lösungen. Jeder hat wohl schon Schüssel-Wert-Tabellen oder Tabellen mit in XML serialisierten Java-Objekten gesehen. Außerdem gibt es Ansätze zum strukturierten Refactoring von Datenbanken [2].

NoSQL-Spielarten

Weil es so viele unterschiedliche Gründe für neue Persistenztechnologien gibt, ist NoSQL auch ein Sammelbegriff. Wesentliche Spielarten von NoSQL sind:

- Key-Value Stores speichern unter einem Schlüssel einen Wert (beides binäre Blobs). Sie sind sehr einfach aufgebaut, aber das Datenmodell kann eine Herausforderung darstellen – Zugriff lediglich über einen Schlüssel ist eben für einige Anwendungen doch zu wenig. Einige Lösungen aus diesem Bereich skalieren sehr gut. Beispiele sind Redis als schnelle In-Memory-Datenbank mit eher begrenzter Skalierung bezüglich der Datenmenge und Riak als sehr skalierbare Lösung.
- Wide Column Stores verwalten die Werte in Tabellen, die sowohl in der Anzahl Spalten als auch in der Anzahl Zeilen sehr groß werden können. Im Gegensatz zu relationalen Datenbanken dürfen die Tabellen aber keine Beziehungen zueinander haben. Beispiele sind HBase aus dem Apache-Hadoop-Projekt und Cassandra, ebenfalls ein Apache-Projekt.

- Dokumentenorientierte Datenbanken speichern hierarchisch strukturierte Dokumente ab, meistens als JSON. Dadurch können auch sehr komplexe Datenstrukturen abgebildet werden. Typische Vertreter sind MongoDB und CouchDB.
- Graphendatenbanken sind vor allem auf die effiziente Speicherung von komplexen, vernetzten Domänen ausgerichtet. In den Graphen sind die Entitäten als Knoten durch getypte Kanten miteinander verbunden. Beide können beliebige Attribute enthalten. Dieses Datenmodell unterscheidet sich fundamental von den anderen drei NoSQL-Varianten, die Beziehungen zwischen Datensätzen weitestgehend vermeiden. Wenn Datensätze stark miteinander in Beziehung stehen, ist es schwer, die Daten auf mehrere Server zu verteilen. Um eine bessere Skalierung über die Nutzung von mehr Servern zu erreichen, gehen daher die anderen NoSQL-Datenbanken bewusst den Trade-off ein, Beziehungen zwischen Daten weniger gut zu unterstützen, um so Skalierbarkeit zu ermöglichen. Als Vertreter dieser Datenbankkategorie ist vor allem Neo4j zu nennen. Mit dieser Datenbank beschäftigt sich auch das Buch Neo4j 2.0 – Eine Graphdatenbank für alle (Michael Hunger).

Kriterien für einen Vergleich

Wie der gesamte NoSQL-Markt wird der shortcut eine Vielzahl von Technologien und Ansätzen für die Lösung ganz unterschiedlicher Probleme umfassen. Um dennoch eine Vergleichbarkeit herzustellen, gibt es für jede Datenbank eine tabellenartige Übersicht. Im Folgenden werden die wesentlichen Begriffe aus der Tabelle erläutert – sie sind an der *kursiven* Schrift erkennbar. Das erste Beispiel für diese Tabelle finden Sie in Michael Hungers Buch zu Neo4j.

Als Erstes ist natürlich das *Datenmodell* relevant. So kann die Datenbank einer Kategorie wie Key-Value dokumentenorientiert, Graphen oder Wide Column zugeordnet werden. Ein weiterer Punkt sind die *Suchmöglichkeiten*. Viele