Graphdatenbanken

Graphdatenbanken (auch **graphenorientierte Datenbanken**) speichern Datensätze in Form von Graphen ab und eignen sich entsprechend gut für das Arbeiten mit derartigen Strukturen, etwa dem Finden und Analysieren von Beziehungen.

Datenmodell

Das Datenmodell der Graphdatenbanken basiert im Grunde auf der Graphentheorie Leonard Eulers. Sie werden in Form von Graphen mit **Knoten** für die Entitäten und gerichteten **Kanten** für die Beziehungen zwischen ihnen modelliert. Beide können jeweils mit Eigenschaften (Properties) versehen werden. Man nennt es auch „**Property-Graph-Datenmodell**“. Alle Knoten und Kanten bilden zusammen die Graphdatenbank.

Jedes Knoten-Element bekommt eine eigene, einzigartige Bezeichnung und enthält zudem, im Gegensatz zu relationalen Systemen, gleich den Satz von ein- und ausgehenden Verbindungen und Beziehungen zu den benachbarten Elementen, sowie die Eigenschaften in Form eines Key-Value-Paares. Auch die Kanten haben eine eindeutige Bezeichnung, Eigenschaften und außerdem Informationen zu ihrem Start- und Endknoten. (Rouse 2014)

Ein Schema muss im Vorhinein nicht definiert werden, Knoten, Eigenschaften, Verbindungen und neue Beziehungsarten können jederzeit dynamisch ergänzt oder entfernt werden. Jedoch sollte beachtet werden, dass Beziehungen immer einen Start- und Endknoten benötigen. Das Löschen eines Knotens kann nur erfolgen, wenn keine Verbindungen zu ihm mehr bestehen. Anders als bei RDBMS ist es möglich, den Datenbestand, basierend auf den Beziehungsgeflechten, auf unterschiedlichste Weise zu interpretieren und zu nutzen, ohne die eingepflegten Daten anpassen zu müssen.

Graphdatenbanken unterscheiden sich von den anderen, aggregatorientierten NoSQL-Modellen. Während die Motivation hinter den anderen Konzepten die war, Systeme zu schaffen, die sich leicht auf große Cluster skalieren lassen und dafür mit relativ großen Datensätzen arbeiten, die nur schwach miteinander verbunden sind, ist es bei Graphdatenbanken genau andersherum. Sie verfolgen ein anderes Ziel, das aus dem Unvermögen relationaler Datenbanken (aber auch anderer NoSQL-Vertreter) beim Umgang mit stark verknüpften Daten hervorging. Sie laufen meist auf Ein-Server-Architekturen (da sich Graphen nicht so einfach partitionieren lassen) und halten eher kleine Datensätze, die aber auf komplexe Weise miteinander verbunden sein können. (Sadalage/Fowler 2012: S. 26ff) Zudem können sie volle ACID-konforme Transaktionen unterstützen. Beispielsweise erlaubt die Datenbank Neo4J Änderungen an einem Graphen nur, wenn dies in einer Transaktion geschieht (Neo4J 2015).

Vorteile

Wenn es um komplexe Beziehungen innerhalb großer Datenmengen geht, sind Graphdatenbanken klar im Vorteil gegenüber den RDBMS, aber auch den anderen NoSQL-Datenbanken.

RDBMS sind dafür eher ungeeignet, da es für das Darstellen und Finden von Verbindungen unter Umständen viele zeitintensive JOINs benötigt, die immer kostspieliger werden, je mehr Datensätze

hinzukommen und je tiefer die Verbindungen zwischen ihnen werden sollen. Graphdatenbanken bieten sich für dieses Problem als Lösung an. Sie wurden speziell dafür entwickelt und optimiert und kommen mit entsprechenden Algorithmen daher.

Da Beziehungen schon bei ihrer Erstellung als Elemente mit angelegt werden, müssen diese bei Abfragen nicht erst aufwendig zur Laufzeit berechnet werden, sondern können direkt verwendet werden. Das bedeutet zwar höhere „Kosten“ beim Anlegen, ermöglicht dafür jedoch ein konstant schnelles Traversieren der Verbindungen/Kanten (und nicht etwa über Keys) bei Abfragen unabhängig von der Gesamtdatenmenge, da nur die für die Abfrage relevanten Verbindungen berücksichtigt werden. Man spricht bei der Navigation durch die Beziehungen auch von **indexfreier Adjazenz**, da sie nicht über globale Indizes erfolgt, sondern eben über das Traversieren der Kanten (Armbruster 2014).

Ein weiterer Vorteil zeigt sich in der Konzeptionsphase bei der Modellierung. Hier hört man gelegentlich das Motto *„if you can whiteboard it, you can graph it“*. Das meint, dass sich Graphen, wie sie leicht verständlich auf Papier oder Whiteboards aufgemalt werden, auch genauso als Datenbank umsetzen lassen können.

Zudem bieten Graphdatenbanken, im Gegensatz zu den anderen NoSQL-Datenmodellen, auch ACID-Konsistenzeigenschaften.

Nachteile

Im Gegenteil zu den übrigen NoSQL-Datenbanken, die ihre Daten hauptsächlich nach ihrem Primärschlüssel in Aggregaten verteilen, müssen Graphdatenbanken ihr Beziehungsgeflecht irgendwie aufbrechen, um das System auf einer verteilten Architektur skalieren zu können und müssen dafür auch entsprechende Operationen bereitstellen. Sharding ist also kein so einfacher Prozess wie bei den anderen Systemen und führt eher zu Performanceverlust, als –gewinn, da diese Systeme eher für Ein-Server-Architekturen entworfen wurden, auf denen das Traversieren schneller geschieht, als auf verteilten Systemen. (Sadalage/Fowler 2012: S. 119) Reicht die Kapazität des Servers nicht aus und soll die Graphdatenbank verteilt werden, muss der Graph in Teilgraphen partitioniert werden, wobei sich das Finden einer sinnvollen Stelle dafür als schwierig erweisen kann und eingehend untersucht werden sollte.

Verwendung

Ihre Stärken liegen in der Findung und Darstellung von Beziehungsgeflechten. Sie eignen sich daher besonders für solche Szenarien, in denen vernetzte Beziehungen in den Daten vorliegen, etwa bei Navigationssystemen, in sozialen Netzwerken („wer kennt wen über wen?“), auch bei Onlineshops für Kaufempfehlungen („Kunden, die dieses Produkt kauften, kauften auch…“) oder für das Bereitstellen personalisierte Werbeinhalte (eventuell auch unter Einbeziehung von Geo-Daten auf mobilen Endgeräten), aber auch im Bereich der Betrugsaufdeckung, wenn versteckte Beziehungen gefunden werden müssen.

Vertreter dieser Kategorie sind: Neo4j, InfiniteGraph, FlockDB.