Key-Value-Datenbanken

Key-Value-Datenbanken (Manchmal auch **Key-Value-Stores** oder **Schlüssel-Wert-Datenbanken**) weisen das wohl einfachste NoSQL-Datenmodell auf. Sie sind bereits seit den 70ern im Einsatz (z.B. Ken Thompsons „DBM“ für UNIX von 1979) und gehören zu den ältesten nicht-relationalen Datenbanken.

Datenmodell

Wie der Name bereits vermuten lässt, verwalten sie ihre Datensätze mit Key-Value-Paaren, wobei zu jedem Schlüssel ein Wert existiert. Anhand der Schlüssel kann die Datenbank schnell durchsucht und auf die Werte zugegriffen werden. Sie können eine willkürliche oder eine festgelegte Zeichenkette sein und in Namensräumen oder Datenbanken aufgeteilt werden (Edlich et al. 2010: S. 7). oder Bei einem Wert kann es sich dabei direkt um Daten in beliebig strukturierter Form handeln (z.B. einzelner Buchstabe, ganzer Text, Bild, Video, etc.) oder aber um eine Referenz, die wiederum auf Daten verweist. Die Daten sind für gewöhnlich Aggregate von zusammengehörenden Informationen, z.B. eine Bestellhistorie eines Kunden inklusive seiner Kundendaten. Die Implementierung eines Key-Value-Stores kann sowohl In-Memory, als auch On-Disk erfolgen (Gull 2012: S. 19f). Eine Variante des Key-Value-Datenmodells ist das dokumentenorientierte Modell.

Vorteile

Im Gegensatz zu ihren frühen Vertretern wurden moderne Key-Value-Systeme mit einem Fokus auf Skalierbarkeit entwickelt. Dabei ist von Vorteil, dass die Daten als Aggregate gehalten werden, denn so können sie als (atomare) Einheit auf einem Cluster-Knoten gespeichert und/oder geladen werden, ohne verstreut zu werden. Ferner erlaubt das einfache Modell über die Schlüssel sehr schnelle Datenzugriffe, wobei sich ein zusätzlicher Performancevorteil ergibt, wenn die Daten im Hauptspeicher gehalten werden. Und schließlich bietet das (für die Datenbank opak gehaltene) aggregatorientierte Datenmodell, von etwaigen Größenbeschränkungen abgesehen, größtmögliche Freiheit, was die Struktur der Daten angeht (Sadalage/Fowler 2012: S. 20f).

Nachteile

Das schlichte Modell, das den Key-Value-Stores bei der Datenhaltung zum Vorteil gereicht, birgt zugleich einige Nachteile bei der Datenverarbeitung. So kann auf die Value-Daten nur über deren Schlüssel zugegriffen werden, was ihre Abfragemöglichkeiten doch sehr einschränkt. Da die Wert-Aggregate für die Datenbank undurchsichtig gehalten sind (eben nur eine beliebige Bit-Reihenfolge), ist eine Suche oder Manipulation innerhalb dieser Wert-Objekte nicht möglich und um sie zu ändern, müssen sie überschrieben werden. Auch das Herstellen und Erkennen von Beziehungen der Aggregate untereinander ist so nicht möglich. Effektiv bietet eine API im einfachsten Falle mit Operationen wie `put(key, value)`, `get(key)`, `remove(key)`, einen doch recht eingeschränkten Funktionsumfang (Vgl. Voldemort o. J.). Insgesamt ist es in seiner Grundform ein sehr simples Datenmodell, das sich wohl nicht für Anwendungen mit komplexeren Abfragen eignet.

Verwendung

Key-Value-Stores können dort eingesetzt werden, wo der Datenzugriff über Schlüssel stattfinden soll und schnelle Reaktionszeiten bei großen Datenmengen gefragt ist und Beziehungen keine große Rolle spielen. So kommen sie u.a. in Web-Anwendungen für Warenkörbe bei Online-Shops oder für die Speicherung von Session-Daten zum Einsatz.

Werden sie nicht als eigenständige Datenbank verwendet, so kommen sie oft auch als Bestandteil anderer Applikationen daher. So können sie (speicherresident) z.B. als Zwischenspeicher für eine Datenbank oder einen Webserver dienen oder als eingebettete Datenbank in UNIX-Systemen (wie „DBM“).

Bekannte Key-Value-Datenbanken sind u.a.: Redis, Riak, Memcached, Cassandra, Amazon DynamoDB.