Extended abstract

NoSQL

Oberseminar: Datenbanken – Aktuelle Trends

Kurt Junghanns 12IN-M

Die Präsentation bietet einen kurzen Einblick in NoSQL.

Zu Beginn wird einleitend NoSQL definiert und die wichtigsten Vertreter gezeigt.
Darauf folgt ein geschichtlicher Abriss der nennenswerten Systeme.
Im dritten Gliederungspunkt wird der Hintergrund zu NoSQL beschrieben und die Anwendungsnotwendigkeit genannt.
Es folgen theoretische Grundlagen und die Kategorisierung der Systeme.
Im folgenden wird von Datenbanken und Systemen gesprochen. Systeme ist dabei der Überbegriff für Datenbanken, DBMS und Frameworks.

NoSQL ist momentan regelmäßig in der Literatur anzutreffen und tritt oft als „Buzzword“ in Erscheinung.
NoSQL steht für not only SQL oder no SQL.
Der Allgemeinheit ist der Begriff NoSQL bereits in der Literatur und in den Medien begegnet, jedoch ist dessen Definition und die dahinterliegenden Ideen weitestgehend unbekannt.
In vielen Fällen gab es Berührungen mit bekannten Systemen, wie BerkleyDB, Redis, Neo4J, MongoDB, Terrastore, CouchDB, Hypertable, Hadoop, Hive, HBase, OrientDB oder Amazons Dynamo.
In der Präsentation wird die Definition aus dem Buch „NoSQL“ von Stefan Edlich und andere aus der zweiten Auflage der Seite 2 herangezogen:

> *Datenbanksysteme, welche folgende Punkte berücksichtigen:*
> *1. zugrundeliegende Datenmodell ist nicht relational*
> *2. System ist verteilt und skaliert horizontal*
> *3. System ist OpenSource*
> *4. System ist schemafrei oder hat nur schwache Schemarestriktionen*
> *5. Aufgrund der verteilten Architektur unterstützt das System eine einfache Datenreplikation*
> *6. System bietet eine einfache API*
> *7. Konsistenzmodell: Eventually consistent und BASE - nicht ACID*

Diese Definition entspricht den Auffassungen des Autors. Dabei ist zu beachten, dass nicht alle Systeme alle Punkte berücksichtigen.
Bereits jetzt wird ersichtlich, dass man sich als Programmierer und Nutzer von altbekanntem verabschieden muss. Dazu zählt Schema, Relationen von Daten und ACID konformer Konsistenz.

Im Laufe der vergangen Jahre wurde der Begriff NoSQL oder auch NoRel immer populärer, wobei Eric Evans auf einer Konferenz 2009 den Begriff hervorhob und NoSQL seit dem zu den Modewörtern zählt.
Als das erste NoSQL System kann IMS von IBM aus dem Jahre 1966 angesehen werden. Es

arbeitete hierarchisch und wurde für das Apollo Programm verwendet. 1979 veröffentlichte Ken Thompsen DBM, ein Datenbanksystem welches die Ablage beliebiger Daten unter Hashwerten ermöglichte.
In den 80ern entstanden weitere Vorfahren heutiger DBS wie beispielsweise Lotus Notes, BerkleyDB, NDBM und GT.M. 1989 nutzte Carlo Strozzi NoSQL als Überbegriff seiner freien leichtgewichtigen Datenbank, was erste Auseinandersetzung in der Literatur über NoSQL nach sich zog.
Im 2. Jahrtausend entstanden Neo4J, Memcached, Infogrid, CouchDB, Amazon Dynamo, MongoDB, Cassandra, Project Voldemort, Terrastore, Redis, HBase, Hypertable und VertexDB. Auch veröffentlichte Google ein Paper namens BigTable, was in Form von HBase, Cassandra und Amazon SimpleDB umgesetzt wurde.
Die Einordnung als NoSQL System kann meist nicht objektiv getroffen werden, da wie bereits genannt, nicht alle NoSQL Systeme alle Punkte der Definition berücksichtigen. Werden Punkte wie die verteilte Arbeitsweise oder hohe Parallelisierbarkeit von Systemen berücksichtigt, fällt in der Literatur zuweilen der Begriff NewSQL.

Besonders durch aktuelle Entwicklung und Web 2.0 entstanden neue Anforderungen an Datenbankmanagementsysteme.
Dazu zählt vorrangig der Bedarf an horizontaler Skalierung, verteiltes arbeiten, geringer Kosten- und Zeitaufwand sowie die Verarbeitung sehr großer Datenmengen. Weiterhin sollen die Systeme parallel arbeiten, hochverfügbar und schemafrei sein.
Dazu trugen auch Big Data, Data Mining und social media bei, bei welchen die Verarbeitung heterogener Daten eine große Rolle spielt.
Bekannte Nutzer von NoSQL Systemen sind Instagramm, Twitter, Tumblr, Amazon, Ebay, Google, Flickr, Cern (für Teilchenbeschleuniger), Wikipedia und Facebook.

Nachfolgend wird auf die wichtigsten theoretischen Grundlagen eingegangen, welche aber ebenfalls nicht bei allen NoSQL Systemen Anwendung finden.

2004 beschrieben Jeffrey Dean und Sanjay Ghemawat in ihrem Paper "*Simplified Data Processing on Large Clusters*" in ihren Rollen als Google Mitarbeiter MapReduce. Dabei handelt es sich um ein Framework, was mit Hilfe von map- und reduce-Funktionen Informationen aus großen verteilten Datenbeständen sammelt, aggregiert und zusammenfasst. Die Funktionen sind dabei wie folgt zu definieren: map erzeugt aus Schlüssel-Werte Paare neue und reduce fasst Paare anhand gleicher Schlüssel zusammen.
Das Framework übernimmt automatisch die Verteilung und Parallelisierung der Funktionen und Daten auf mehrere Nodes/Knoten. Weiterhin die Durchführung und Überwachung von Lese- und Schreiboperationen, Fehlerbehandlung und allgemeine Kommunikation. Zusätzlich ist dieses Framework für die Durchführung auf Standardhardware ausgelegt.

Herr Brewer definierte das CAP Theorem consistency, availability and partition tolerance. Dabei steht consistency für das Erreichen eines konsistenten Zustandes von Transaktionen im gesamten System. Availability steht für gegebene Verfügbarkeit und akzeptable Reaktionszeit mit 100% Datenverfügbarkeit. Partition tolerance dagegen meint, dass der Ausfall von Kommunikationsverbindungen (oder Veränderungen der Nodes) nicht zum Ausfall des Systems führt und dieses eingeschränkt noch nutzbar ist.
Von diesen drei Eigenschaften können DBS/DBMS höchstens zwei besitzen. ACID konforme relationale Datenbanken erfüllen consistency und availability. NoSQL Systeme erfüllen zumeist die Eigenschaften availability und partition tolerance.

Auf Grund der neuen Anforderungen fand nicht nur eine Abkehr des relationalen Charakters der Daten statt, sondern auch von der ACID konformen Konsistenz. Der Konsistenzbegriff ist in diesem Rahmen ein anderer, in welchem die Konsistenz fließend ist, d.h. von Zeit zu Zeit für einen Teil der Datensätze gegeben ist. Dieses Konzept heißt BASE und steht für basically available, soft state, eventual consistent. Dabei handelt es sich nicht nur um den lockeren Konsistenzbegriff, sondern um eine Menge von weiteren Konzepten und Ansätzen ohne Sperren und mit Synchronisation.
Dazu zählen die nächsten drei Konzepte.

Der aus BASE stammende Konsistenzbegriff eventual consistency beschreibt, dass die Konsistenz der verteilten Daten nicht gegeben ist, sondern mit Synchronisation über Nodes eines Netzes hergestellt werden muss. Dazu existiert ein Zeitfenster, ein inconsistence-window, in welchem die Konsistenz bestimmter Daten hergestellt wird. Innerhalb dieses Zeitfensters existieren verschiedene Daten und Datenversionen.

1997 definierte David Karger consistent hashing. Dabei handelt es sich um ein Hashverfahren mit dem Ziel bei horizontaler Skalierung eine Gleichverteilung der Daten anhand deren Hashwerte zu erreichen. Speziell werden Daten und Server einem Hashwert zugeordnet und anschließend die Daten anhand ähnlicher Hashwerte bezüglich der Hashwerte der Server diesen zugeteilt. Dabei wird oftmals die Leistungsfähigkeit der Server für den Hashwert berücksichtigt und es werden Kopien der Daten der Vorgänger mitgespeichert.
Neben der Gleichverteilung der Daten ist dadurch auch ein geringer Aufwand für das entfernen oder hinzufügen von Servern/Nodes möglich und der Zugriff auf die Daten ist ohne Umwege über deren Hashwert möglich.

Das dritte Konzept, welches zu BASE gezählt wird, ist MVCC multiversion concurrency control. Auch bei BASE existieren Transaktionen. Jedoch arbeiten diese ohne Sperren, ergo handelt es sich um einen optimistischen Ansatz. Daten besitzen hierbei neben ihren eigentlichen Informationen eine ID, einen Zeitstempel und einen Verweis auf ihren eigenen Vorgänger. Auf Grund des Konsistenzmodelles stellt die Benutzung von Daten mit neuen Nachfolgern durch Transaktionen keinen Abbruchgrund dieser dar. Einzig Schreiboperationen innerhalb einer Transaktion auf veränderte Daten führt zur Wiederholung dieser Transaktion. Die Überprüfung auf eventuell neue Datenversionen findet bei schreibenden Aktionen am Ende der Transaktion statt.
Diese Konzepte werden von BASE konformen Systemen angewandt und ermöglichen verteiltes arbeiten und Ausfallsicherheit.

Eine weitere Grundlage ist REST oder RESTful. Representational state transfer findet in HTTP Anwendung. Dabei wird davon ausgegangen, dass jede Adresse zu genau einer Ressource führt. Es existieren folgende Befehle: HEAD, GET, PUT, POST und DELETE.
Außer POST liefern alle Befehle immer dieselben Informationen(HEAD) oder Ressourcen(GET), haben immer dieselben Nebeneffekte (PUT und DELETE). Werden die 4 Befehle je mehrmals hintereinander ausgeführt, hat dies den gleichen Effekt wie eine einzige Ausführung. POST dagegen liefert nicht immer dieselbe Ressource, da Nebenwirkungen durch POST diese verändern können.
Das Ziel dahinter ist, Zustandsinformationen nicht zwischen den Anfragen zu speichern.
REST findet sich in den verschiedenen Abfragesprachen der Systeme wieder, wobei einige über HTTP ansprechbar sind.

Der letzte Hauptpunkt ist die Kategorisierung der Systeme. Nicht jedes System kann genau einer Kategorie zugeordnet werden, da mehrere Eigenschaften der Kategorien erfüllt sein können.

Die erste Kategorie ist Key/Value. Hierbei werden die Daten als Schlüssel-Werte Paare gespeichert. Der Schlüssel ist eine Zeichenkette, welche meist einen Hashwert darstellt. Für den Wert gibt es keine Vorgabe des Syntax oder der Struktur. Der Zugriff erfolgt über den Schlüssel. In diesen Systemen gibt es keine einheitliche Abfragesprache.
Vorteile sind hier die einfache Handhabung, der performante Zugriff über den Hash, die Abwesenheit der Notwendigkeit von Indexen, die immanente Skalierung, welche aus der Unabhängigkeit der Daten rührt, und die Möglichkeit beliebige Daten abzulegen.
Bekannte Vertreter sind Redis, Chrordless, Riak, MEMBASE, Voldemort und Amazon Dynamo.

Dokumentorientierte Datenbanken speichern zu Hashwerten strukturierte und versionierte Dokumente ab. Diese haben zumeist das Format JSON, XML oder YAML. In diesen Systemen sind meist keine Joins möglich.
Die Schemaverantwortung wird an die Anwendung abgegeben.
Es sollen schemafreie Daten abgelegt, der Datenbestand in der Handhabung skaliert und Zugriff auf und in die Dokumente ermöglicht werden.
Wichtige Vertreter sind MongoDB, CouchDB und Terrastore.

In spaltenorientierten Datenbanken werden beliebige Attribute je in einer eigenen Zeile hintereinander abgelegt.
Dies hat den Vorteil, dass die Anzahl der Attribute beliebig sein kann und Spalteneinfügungen kostengünstig sind. Die Literatur weist darauf hin, dass hier garbage collection effektiv arbeitet und diese Systeme zur Datenanalyse, Datenkompression und Caching ausgelegt sind und sich für OLAP und data warehousing eignen.
Als Nachteil ist der hohe Aufwand beim Lesen und Schreiben von zusammengehörigen Spaltendaten zu nennen.
Zu spaltenorientierten Datenbanken zählt Sybase IQ, FluidDB, C-Store und MonetDB.
Dieses Konzept wurde durch das Paper von Google namens BigTable erweitert und wurde vielfältig adaptiert, Umsetzungen sind HBase, Cassandra und Amazon SimpleDB.

> „*A Bigtable is a sparse, distributed, persistent multi-dimensional sorted map. The map is indexed by a row key, column key, and a timestamp; each value in the map is an uninterpreted array of bytes.*“ (Bigtable: A Distributed Storage System for Structured Data, Google, Inc.)

In BigTable sind die mehrdimensionalen Tabellen von folgendem Format:
n*[Domain / Keyspace] x [item / Column Family] x [Key x] n*[key+Value]
Auf der oberen Ebene sind die Daten gruppiert und auf den untersten konkrete Schlüssel-Werte Paare.
Dieser Ansatz ist sehr gut skalierbar und für sehr große Datenmengen ausgelegt.

Die letzte Kategorie ist die der graphenorientierten Datenbanken. Die Daten werden in Knoten und Kanten abgebildet, wobei Knoten strukturierte Objekte und Kanten typisierte Beziehung sind.
Diese Datenbanken können auch ACID konform sein.
Das Schema der Daten ist optional und es existiert keine einheitliche Abfragesprache sowie keine Joins.
Geeignet ist dies zur Darstellung von semantischen Beziehungen zwischen Objekten, wie sie bei social media, semantic web, Bioinformatik und Internetrouting vorkommen.
Vertreter sind Neo4J und FlockDB.

Die Landschaft der NoSQL Systeme ist mannigfaltig, sodass hier nicht alle Eigenschaften beschrieben werden können. Auch ist abzuwägen, welches System für welchen Anwendungsfall geeignet ist.
Weiterhin fällt auf, dass die wichtigsten Vertreter der NoSQL Systeme von großen Unternehmen unterstützt oder sogar entworfen und implementiert wurden.

Quellen

- NoSQL, 2. Auflage, Hanser Verlag, Stefan Edlich, Achim Friedland, Jens Hampe, Benjamin Brauer, Markus Brückner
- Entwurf und Realisierung einer verteilten NoSQL-Anwendung, Alexander Ponomarenko, Bachelorarbeit, 2011, Hochschule für Angewandte Wissenschaften Hamburg
- NoSQL-Datenbanken, Philipp Heinze, 2010, Friedrich-Schiller Universität Jena
- NoSQL Databases, Christof Strauch, Hochschule der Medien Stuttgart
- Bigtable: A Distributed Storage System for Structured Data, Google, Inc., 2006
- MapReduce: Simplified Data Processing on Large Clusters, Google, Inc., 2004
- http://www.ivanomalavolta.com/diving-into-nosql/, Diving into NoSQL, Ivano Malavolta, Abruf: 26.05.2013
- http://de.slideshare.net/cloudera/hw09-hadoop-development-at-facebook-hive-and-hdfs, Hadoop and Hive Development at Facebook, Dhruba Borthakur Zheng Shao, Abruf: 02.06.2013
- http://blog.knuthaugen.no/2010/03/a-brief-history-of-nosql.html, A brief history of NoSQL, Knut Haugen, Abruf: 23.5.2013
- http://techcrunch.com/2012/08/22/how-big-is-facebooks-data-2-5-billion-pieces-of-content-and-500-terabytes-ingested-every-day/, How Big Is Facebook’s Data? 2.5 Billion Pieces Of Content And 500+ Terabytes Ingested Every Day, Josh Constine, Abruf 7.6.2013