Neues aus der nicht-, semi- und relationalen Welt

Thomas Klughardt
Dell Software
Köln

Schlüsselworte

Big Data, NoSQL, Unstrukturierte Daten, Skalierbarkeit, NewSQL, Konsistenz, ACID, Fehlertoleranz

Einleitung
Seit ein paar Jahren beherrscht das Thema Big Data die IT Fachpresse. Gemeint ist damit die Möglichkeit, Daten zu speichern und auszuwerten, die bisher Aufgrund der schieren Menge, die anfällt, aufgrund der Geschwindigkeit, mit der diese Daten anfallen oder einfach aufgrund von fehlenden Strukturen und Abhängigkeiten bisher nicht verarbeitet werden konnten, oder bei denen es sich nicht gelohnt hat. Das können sowohl die anwendergenerierten Daten sein, also zum Beispiel Daten von sozialen Netzwerken wie Twitter oder Facebook, als auch maschinengenerierte Daten von Sensoren, Steuerungseinheiten und anderen maschinellen Datenquellen. Für das Geschäft sind diese Daten nützlich, sie auswerten zu können bringt dem Unternehmen einen Wert, für die IT ist das Verarbeiten dieser Daten eine Herausforderung. Es mussten also neue Systeme her. Und in diese Bresche sind zuerst ein paar neue Anbieter gesprungen, aber auch die Platzhirsche erobern den neuen Markt. Hier geht es darum, einen Überblick über die neuen Systeme zu geben, aber auch darum, warum Big Data eben nicht immer NoSQL bedeutet und warum die klassischen relationalen Datenbanken durchaus auch noch ihre Existenzberechtigung haben.

Unstrukturierte Daten – was bedeutet das?
Ein Begriff, den man immer wieder im Zusammenhang mit Big Data hört, ist der Begriff der unstrukturierten Daten. Oft wird dann vermutet, dass es hier darum geht, dass ein Stream an komplett unbekannten Binärdaten in das System geschossen wird, der keinen Anfang und kein Ende hat. Und möglicherweise gibt es solche Anwendungen tatsächlich, irgendwo müssen Kryptographen ihre Daten schließlich auch verarbeiten, aber das ist sicher nicht der Normalfall. Mit unstrukturierten Daten arbeiten heißt, dass wir an sich schon wissen, mit was für Daten wir zu tun haben, sie aber keine normalisierte Schemastruktur haben. Das können Texte oder Zahlen sein, Audio- und Videostreams oder Bilder, aber es sind eben schon Daten, die wir verstehen und Auswerten können.
Im Gegensatz zu Daten, die in einer relationalen Datenbank gespeichert werden, muss man die Struktur erst beim Verarbeiten kennen, weshalb diese Systeme auch „Schema on Read“ Systeme nennt. Eine relationale Datenbank nimmt Daten nur entgegen, wenn sie schon in der richtigen Schemastruktur kommen, deshalb heißen sie auch „Schema on Write“ Systeme.

Grenzen relationaler Datenbanken
Natürlich könnte man auch in einer relationalen Datenbank unstrukturierte Daten speichern, zum Beispiel in BLOB Feldern. Dabei wäre die ganze Intelligenz, die so eine relationale Datenbank mitbringt, aber einfach nutzloser Overhead, der keinen Vorteil bringen würde.
Die Alternative wäre, die Daten schon beim Schreiben in eine Schemastruktur zu pressen, aber dabei werden sie verändert und unter Umständen gehen Informationen verloren, die man gerne auswerten würde. Man sagt auch, es geht dann Schärfe verloren. Insofern sind in diesem Fall Datenspeicher besser, die eben genau darauf optimiert sind, unstrukturierte Daten zu speichern und auszuwerten.

Es gibt aber auch noch einen weiteren Grund, warum klassische relationale Datenbanken nicht unbedingt geeignet sind, große Datenmengen zu verarbeiten: Das CAP Theorem.

Im Wesentlichen besagt das CAP Theorem, dass es keine Möglichkeit gibt, die Daten zur gleichen Zeit Konsistent, jederzeit (schnell) Verfügbar sein und gut skalieren können. Will man eine bestimmte Eigenschaft erreichen, muss man Einschränkungen bei den anderen Eigenschaften hinnehmen.

1. *Konsistenz*
   Wenn die Daten immer konsistent sein müssen, dann muss die Konsistenz sichergestellt sein, bevor die Daten wieder verfügbar werden. Das heißt, dass es eine zentrale Instanz der Daten, den Master, geben muss, bei dem die Konsistenz sichergestellt wird.

2. *Verfügbarkeit*
   Wenn die Daten jederzeit verfügbar sein müssen, dann kann man das Zurückgeben der Daten nicht verzögern. Das bedeutet dann auch, dass bei redundanter Datenhaltung alle Datenbestände jederzeit abgefragt werden können und das Ergebnis sofort geliefert werden muss.

3. *Skalierbarkeit*
   Wenn der Datenbestand gut skalierbar sein soll, muss das System unabhängig von Datenmenge und Last gleiche oder ähnliche Antwortzeiten liefern. Das funktioniert nur dann, wenn die Daten redundant vorgehalten werden und die Einzelbestände unabhängig voneinander abgefragt werden können.

Wenn man sich die einzelnen Anforderungen für die Eigenschaften ansieht, wird schnell klar, dass sie sich teilweise widersprechen und es deshalb kein System geben kann, das in der Lage ist, alle Anforderungen zu erfüllen. Das ist die Aussage des CAP Theorems, das bisher nicht widerlegt wurde.

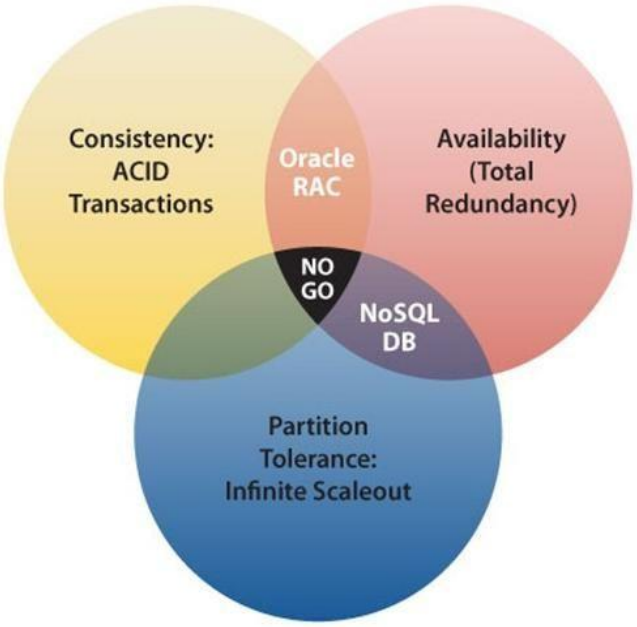

Abb. 1: Das CAP Theorem als Venn Diagramm (Quelle: DBPedias.com)

Das CAP Theorem sagt also: Eine relationale Datenbank, die ACID konform arbeitet kann also, egal wie optimiert sie läuft und wie gut sie entwickelt ist, nicht unbegrenzt skalieren.

NoSQL Systeme

„Klassische relationale Datenbanken sind tot!“ – wenn man den Marketingaussagen einiger Hersteller von NoSQL Systemen Glauben schenkt, könnte man das tatsächlich denken. Da werden Benchmarks gezeigt, wie auf Hadoop Clustern Peta- und Exabyteweise Daten verarbeitet werden, wie Abfragen auf riesigen Datenmengen in Document Stores in Bruchteilen von Sekunden fertig sind und sie alle schlagen die relationalen Datenbanksysteme. Klingt gut, allerdings werden da Äpfel mit Birnen verglichen. NoSQL Systeme eignen sich nicht als Ersatz für eine relationale Datenbank, wenn ständige Konsistenz gefordert ist.
Die meisten NoSQL Systeme nutzen statt ACID das „BASE“-Prinzip (Basically Available, Soft state, Eventual consistency). Es geht ihnen gar nicht darum, die Daten immer konsistent zu halten und normalisierte Schemastrukturen zu nutzen, sondern die Daten werden irgendwann immer mal wieder über einen Job glatt gezogen sind also erst irgendwann konsistent und man weiß nie wann genau. NoSQL bedeutet übrigens nicht, dass diese Systeme keine SQL Abfragen verarbeiten können, für viele gibt es inzwischen Bibliotheken, die es möglich machen über Statements auf Daten zuzugreifen und sie zu verändern.

Zunächst muss man auch mal zwischen den verschiedenen Systemen unterscheiden, die es so gibt. Dazu gibt es auf http://nosql-databases.org/ eine sehr schöne Übersicht von Professor Edlich, wo einfach mal die verschiedenen NoSQL Datenbanken kategorisiert und kurz beschrieben sind. Je nachdem, welchen Einsatzzweck man hat, können unterschiedliche Systeme sinnvoll sein. Ein Hadoop Cluster ist zum Beispiel dazu da, im Batchbetrieb große Datenmengen zu verarbeiten und daraus ein Ergebnis zu kondensieren. Hadoop ist aber erst mal nur ein Objektspeicher mit einer MapReduce Implementierung, was ein ursprünglich von Google entwickeltes Verfahren ist, Berechnungen auf großen Datenmengen zu parallelisieren.
Die aggregatorientierten Datenbanken wie zum Beispiel Wide Column Stores oder Document Stores sind gut darin, Objekte über einen Schlüssel aufzufinden, was im Prinzip einer Hashtable entspricht. Das ist für Webanwendungen geeignet, die bei vielen Anfragen schnell Daten liefern können, dafür sind hier keine Joins zwischen Dokumenten möglich. Wenn man sich eine Auftragsverwaltung denkt, kann man hier extrem schnell einen Auftrag finden und zurückliefern, wir können aber nicht wie in einer relationalen Datenbank herausfinden, welcher Umsatz mit welchen Produkten gemacht wird. Es gibt aber eine Vielzahl von NoSQL Systemen, die jeweils für verschiedene Anwendungsszenarien geeignet sind. Um das zu vertiefen sind auch die Seite von Martin Fowler (http://martinfowler.com/) und das Buch „NoSQL Distilled“ sehr empfehlenswert.
Um diese Information nicht zu unterschlagen: es gibt auch von Oracle einen Key-Value-Store, die Oracle NoSQL Database. Der hat allerdings mit dem mächtigen Oracle Database Server, also der relationalen Oracle Datenbank, die wir alle kennen, nichts zu tun.

NewSQL Datenbanken

Seit einiger Zeit gibt es auch die sogenannten NewSQL Datenbanken, die versuchen, ACID konforme Transaktionen bei guter Skalierbarkeit zu erreichen. Die bekanntesten Vertreter hierbei sind Google’s Spanner Datenbank und NuoDB. Sie sollen den Anwendungsentwicklern ermöglichen, wie bisher ganz normal SQL Statements zu schreiben und so Anwendungen leichter portierbar machen, während aber im Hintergrund eine auf vielen Rechenknoten verteilte Shared Nothing Architektur läuft. Auch diese Systeme sind derzeit eher für ganz bestimmte Anwendungen geeignet, bei denen zum Beispiel nur selten aus verschiedenen Sessions auf die gleichen Datenbereiche zugegriffen wird, bei denen wenn überhaupt sehr kurze Sperren gehalten werden und nur kleine Datenbereiche angefasst werden. Bei Jobs, die über große Datenbereiche gehen und zum Beispiel Full Table Scans machen, vielleicht auch zu Wait Events führen oder bei denen komplexe Abfragen stattfinden, sind sie nicht gut geeignet. Auch hier schlägt dann wieder das CAP Theorem zu. Es lohnt sich aber trotzdem, auch diese Systeme im Auge zu behalten, es kommt eben auf die Anwendung an.

Die Platzhirsche machen mit: In Memory Column Stores
Im Zusammenhang mit Big Data hört man auch oft von zwei Produkten, die im Markt große Wellen schlagen, auch wenn noch nicht viele Leute Erfahrungen damit gemacht haben: SAP HANA und inzwischen der Oracle In Memory Option. Beide sind Column Stores, das heißt anders als bei einer zeilenbasierenden Speicherung wie in einer normalen Tabelle, werden die Daten hier Spaltenweise abgespeichert. Das hat große Vorteile, wenn man bestimmte Spalten mehrerer Zeilen abfragt, auch weil für numerische Felder nach ein paar Zeilen immer die Aggregatwerte, also Summe, Minimum, Maximum und Durchschnitt abgespeichert werden, was für Abfragen auf aggregierte Werte die Verarbeitung erheblich beschleunigt. Es ist auch wesentlich einfacher, die Daten zu komprimieren und weiter an IO Zugriffen zu sparen, weil sich in der Praxis Werte in Spalten oft wiederholen.
Nicht nur Oracle und SAP sind hier unterwegs, auch IBMs DB2, der Microsoft SQL Server und einige andere Datenbanken bieten In-Memory-Columnstores an. Die sind vor allem für OLAP Anwendungen also zum Beispiel für Data Warehouses interessant. Für OLTP Anwendungen sind Zeilenbasierte Speicher oft besser, weil zum Beispiel Insert Operationen ihren Datensatz einfach hinter die letzte Zeile schreiben können, während ein Column Store erst mal in jeder Spalte die richtige Stelle finden muss.
Während SAP HANA für die SAP ERP Anwendungen optimiert ist und auch in der Lage ist, einen Teil der Anwendungsfunktionalität in die Datenbank auszulagern, sind die anderen Datenbanken für alle Einsatzszenarien verwendbar und es wird sich zeigen, wer sich hier welchen Marktanteil erkämpfen kann.

Fazit
Es gibt gerade einige Bewegung auf dem Markt und im Moment ist noch nicht abzusehen, wer letztendlich das Rennen machen wird. Derzeit sind es alleine bei den NoSQL Datenbanken über 150 verschiedene Produkte in verschiedenen Kategorien, die sich teilweise sehr stark voneinander unterscheiden. Dazu kommen die neuen Systeme der großen Hersteller, die auch im Big Data Bereich mitspielen möchten.
Big Data ist kein Synonym für NoSQL, es gibt viele weitere Systeme und auch die relationalen Datenbanken haben meist noch Luft nach oben und sind lange nicht ausgereizt. Insgesamt ist der Big Data Begriff auch noch nicht wirklich scharf definiert, ähnlich wie bei dem Begriff Cloud vor einigen Jahren, wird es aber konkreter und es darf schon lange nicht mehr jeder Hersteller von sich behaupten, eine Big Data Lösung im Portfolio zu haben.
Für die IT Abteilungen ist es deshalb wichtig, sich frühzeitig mit dem Thema auseinanderzusetzen. Die Anfragen nach Big Data Lösungen, die es möglich machen zusätzliche Datenquellen zu nutzen, werden kommen und die Geschäftsbereiche haben auch allen Grund dazu, schließlich sind diese Daten wertvoll. Wenn die Anforderung kommt, ist es aber wichtig, sich von der IT Seite schon damit auseinandergesetzt zu haben, vielleicht auch alleine schon deshalb um den Big Data Consultants die dann gerne dazu kommen, ein wenig den Wind aus den Segeln zu nehmen.

Kontaktadresse:
Thomas Klughardt
Dell Software
Im Mediapark 4e
D-50670 Köln

Telefon: +49 (0) 221-5777 4114
Fax: +49 (0) 221-5777 4114
E-Mail thomas.klughardt@software.dell.com
Internet: software.dell.com