Kommentar von Alexander Lapp, Adacor Hosting

Big Data, SQL und NoSQL – eine kurze Übersicht

24.05.2017 | Autor / Redakteur: Alexander Lapp / Nico Litzel |

Die Begriffe Big Data, SQL und NoSQL, relationale oder objektorientierte Datenbanken werden häufig in einen Topf geworfen und in beliebigen Querverbindungen miteinander verglichen. Tatsächlich hat das eine mit dem anderen zwangsweise nichts zu tun. Woher kommt dann diese Denkweise und wie kann man die Themen korrekt einordnen?

Der Autor: Alexander Lapp ist Geschäftsführer und Chief Customer Officer der Adacor Hosting GmbH
(Bild: Adacor Hosting)

Michael Kroker fasste es in einem seiner Wirtschaftswoche-Blogs <http://blog.wiwo.de/look-at-it/2015/02/23/big-data-datenvolumina-steigen-40-prozent-pro-jahr-und-insgesamt-um-faktor-50-bis-2020/> gut zusammen: Das Datenvolumen steigt um 40 Prozent pro Jahr – und insgesamt um den Faktor 50 bis 2020. Das heißt, in drei Jahren werden es weltweit 44 Zettabyte Datenvolumen sein, das entspricht 5.200 Gigabyte pro Erdenbürger. Verdeutlicht man sich diese Werte, so wird klar, dass zukünftig kein Unternehmen mehr um das Thema „Big Data <https://www.bigdata-insider.de/was-ist-big-data-a-562440/> “ herumkommen wird. Besonders spannend ist in diesem Zusammenhang die Frage nach den Möglichkeiten der Verarbeitung solcher Massendaten.

So sollte beispielweise der Einsatz von Datenbanken niemals von der Vorliebe der Entwicklung abhängig sein, sondern sich immer am Anwendungszweck des jeweiligen IT-Projekts – bezogen auf die Gesamtinfrastruktur – orientieren. Der Einsatz einer relationalen Datenbank kann zum Beispiel durchaus sinnvoll sein, auch wenn das Modell selbst gerade nicht en vogue ist. Im Gegensatz dazu ist es vielleicht

unvernünftig eine NoSQL <https://www.bigdata-insider.de/was-ist-nosql-a-615718/> -Datenbank für einen überschaubaren und strukturierten Datenbestand einzusetzen, nur weil das Vorgehen gerade in aller Munde ist. Welches Datenbankmodell sich im Einzelnen am besten eignet, hängt vom Einsatzzweck ab.

Übersicht der gängigen Datenbankmodelle

1. Relationale Datenbanken
2. Objektorientierte Datenbanken
3. Dokumentenbasierte Datenbanken

(Hier nicht berücksichtigt: Netzwerkdatenbankmodell, Objektrelationale Datenbanken, Hierarchische und so weiter)

1. Relationale Datenbanken

Relationale Datenbanken können über SQL abgefragt werden. Sie zählen zu den am weitesten verbreiteten Datenbankmodellen. Der Grund dafür liegt darin, dass relationale Datenbanken einfach und flexibel zu erstellen und zu steuern sind. Am bekanntesten sind die Systeme von Oracle, MySQL <https://www.bigdata-insider.de/was-ist-mysql-a-614184/> und MSSQL.

Bei dem relationalen Datenbankmodell findet die Datenspeicherung in strukturierten Tabellen statt. Diese stehen zueinander in einer Relation (Beziehung), die über einen Fremdschlüssel realisiert wird. Darüber hinaus können die Beziehungen mithilfe eines Primärschlüssels hergestellt werden. Das Ziel ist es dann, bestimmte Eigenschaften abzufragen, die den gleichen Primärschlüssel oder eine detaillierte Tabelle als Fremdschlüssel besitzen.

Jede Tabellenzeile (Tupel) steht dabei für einen Datensatz (Record). Darüber hinaus besteht jede Zeile aus einer Reihe von Attributen (Merkmalen). Diese stellen die Tabellenspalten dar. Über das Abfragen der in der Relation eingebundenen Tabellen werden die gewünschten Informationen errechnet und aufgelistet.

Die Themen „Performance“ und „die fehlende Gewährleistung der Datenintegrität“ bringen bei komplexen Relationsmodellen und vielen Daten einige Nachteile mit. Die Performance-Probleme resultieren aus der Mengenlehre. So müssen die Tabellen zueinander geführt werden, um sie auszuwerten. Das passiert mit jeder Verknüpfung in einer Menge, die dann wieder mit der neuen Verknüpfung zusammengeführt wird. In der daraus entstehenden Endmenge werden die Ergebnisse gefiltert. Dieses Vorgehen führt bei vielen Tabellen zu hoher Komplexität und großen temporären Speicheraktionen. Die Rechenvorgänge dauern deshalb entsprechend lange und verbrauchen viel Speicher. Bei der Datenintegrität hingegen gibt es potenziell falsche Ergebnisse, wenn man die Integrität nicht „programmiert“ hat. Diese Nachteile können nur teilweise über das System und die genutzte Fremdschlüsselstruktur abgefangen werden und sitzen zum Teil in der Applikation.

2. Objektorientierte Datenbanken

Objektorientierte Datenbanken basieren darauf, dass die Datenhaltung zusammen mit ihren Funktionen in einem Objekt erfolgt. Das Objekt, in dem die Daten abgelegt sind, übernimmt dann intern die Datenverwaltung. Dazu gibt es zum Datenbankmodell passende Objektsprachen. Mit deren Hilfe oder über die Objektfunktionen können die Daten ausgelesen werden.

Die Objektsprachen orientierten sich an dem Musterbild der objektorientierten Programmierung und der jeweiligen Programmiersprache wie zum Beispiel Java oder C++.

Objektorientierte Datenbanken sind in der Praxis wenig verbreitet. Dadurch gestaltet sich ihre Implementierung kompliziert, denn es fehlen einige Schnittstellen und Zwischen-Layer, um eine ähnlich flexible Entwicklungsmöglichkeiten zu erreichen wie bei relationalen Datenbanken. Weiterhin kann die Komplexität eines Objektes so stark steigen, dass die Geschwindigkeit des Objektdatenmanagementsystems (ODMS) massiv sinkt. Mit der zunehmenden Verbreitung objektorientierter Programmiersprachen dürfte das Modell in Zukunft jedoch an Bedeutung gewinnen.

3. Dokumentenorientierte Datenbanken

Bei dem dokumentenorientierten Datenbankmodell werden die Daten in Dokumenten abgelegt. Ein Dokument steht in diesem Zusammenhang für einzelne, aber in sich unterschiedlich strukturierte Einheiten. Das können Dokumente im Standarddateiformat sein oder strukturierte Dateien mit einem festlegbaren Schema an Datenfeldern. Jedes dieser Dokumente besitzt einen eindeutigen Identifikator, mit dessen Hilfe das Dokument immer wieder gefunden werden kann. Die einzelnen Dokumente stehen nicht im Zusammenhang zueinander. Die Daten in einem Dokument werden in Form von Schlüssel-/Wertpaaren gesichert. Das heißt, das Dokument besteht aus einer Reihe mit Namen versehener Schlüsselfelder, denen jeweils ein bestimmter Wert zugeordnet ist.

Bei vielen dokumentenorientierten Datenbanken erfolgt die Abfrage mit einem nicht-relationalen Ansatz (NoSQL). Das heißt, es besteht keine SQL-ähnliche Syntax um Beziehungen zwischen den Tabellen und Spalten beziehungsweise in diesem Fall Dokumenten abzufragen.

So unterscheiden sich NoSQL und SQL von den Datenbankmodellen

Aus den bisherigen Ausführungen ist zu erkennen, dass die Begriffe „NoSQL“ und „SQL“ nicht auf einer Ebene mit den Datenbankmodellen stehen. Vielmehr stellen sie die Datenbanksprachen dar, in einer Art und Weise, wie Daten innerhalb dieser Modelle gespeichert und wieder abgefragt werden. Das funktioniert bei relationalen Datenbanken, zum Beispiel mit SQL, und bei dokumentenorientierten Datenbanken mit dem Sammelbegriff NoSQL. Es gibt also diverse Unterschiede, aber es ist nirgendwo festgelegt, welche Differenzierungen tatsächlich maßgebend sind. Darüber hinaus gibt es weitere Datenbankmodelle, die außerhalb dieser drei Elemente anzusiedeln sind: zum Beispiel grafenorientierte Datenbanken.

Die vier Unterschiede zwischen NoSQL und SQL

Im Wesentlichen gibt es vier Unterschiede zwischen NoSQL und SQL als Datenbanksprache und der Art und Weise, wie die Daten innerhalb des Datenbankmodells gespeichert werden:

1. Tabellen gegenüber Sammlungen (Tables versus Collections)
2. Unterschiedlicher Normalisierungsgrad bei den Datenbankinhalten
3. Verschiedene Anforderungen an die Skalierung
4. ACID versus BASE

Bekannte Beispieltechnologien
(Bild: Adacor Hosting)

1. Tabellen gegenüber Sammlungen (Tables versus Collections)

SQL-basierte Datenbanken speichern Daten in Tabellenform. Die Tabellen werden dabei in Relation zueinander gesetzt. Das heißt, bereits ein Excel-Sheet stellt in Form und Aufbau eine Tabelle einer SQL-Datenbank dar.

Innerhalb des Datenbankmodells kann es weitere Tabellen geben, die auf die oben genannte Tabelle verweisen. Dazu wird der in der Tabelle „Adressen“ vorhandene #schluessel# verwendet und als Eintrag in weiteren Tabellen genutzt. So entsteht eine voneinander abhängige Datenstruktur, die in einer Tabellenform aufgebaut ist. Das heißt, eindeutige Schlüssel, Indexspalten zum schnelleren Auffinden von Werten in der Datenbank sowie Verknüpfungen zu anderen Datenbanktabellen (Relationen) sind obligatorisch.

Ein beliebtes Beispiel ist eine typische Adresskartei (Tabelle Adressen).
(Bild: Adacor Hosting)

NoSQL-basierte Datenbanksysteme speichern dagegen die Daten auf verschiedene Art und Weise, zum Beispiel indem die Datensätze als flexible Schlüssel-/Wertpaaren aufgebaut sind. Diese Schlüssel-/Wertpaare sind nicht statisch festgelegt und befinden sich innerhalb eines „Dokuments“ als Datensatz.

```
{
```

```
NAME: "Max Mustermann",
Strasse: "Relationsgasse 5",
PLZ: "12345",
Ort: "Datenbergen"
}
```

Ohne die eigentliche Datenbankstruktur zu ändern, könnte innerhalb dieses Datensatzes ein neues Schlüssel-/Wertpaar eingesetzt werden:

```
{
NAME: "Max Mustermann",
Strasse: "Relationsgasse 5",
PLZ: "12345",
Ort: "Datenbergen",
Telefonnummer: "5552322"
}
```

2. Unterschiedlicher Normalisierungsgrad bei den Datenbankinhalten

Wenn man die festgelegte Struktur der SQL-basierten Datenbank berücksichtigt und davon ausgeht, dass die dadurch implementierten Regeln strikt eingehalten werden (damit werden Fehler vermieden), ist ein hoher Normalisierungsgrad bei den Datenbankinhalten (Aufteilung in einzelne Werte) nötig, um dies ausreichend zu gewährleisten. Die Normalisierung <https://www.bigdata-insider.de/was-ist-normalisierung-a-840412/> macht das Datenmodell statischer und sorgt gleichzeitig dafür, dass das Aufspannen der modellierten Struktur über verschiedene Tabellen potenziell mehr Last und Logik im Datenbanksystem erzeugt. So ist die Datenablage in NoSQL-Datenbanken durch zusammenhangslose Datensammlungen wesentlich einfacher. Das Lesen und Schreiben auf ein Element passiert schneller. Dieser Vorgang wird deutlich, wenn Daten aus einem SQL-System gelesen werden sollen und parallel ein Schreibvorgang einzelne Tabellen „sperrt“, um die Transaktion mit allen ihren Abhängigkeiten abzuschließen. SQL wartet mit dem Lesen bis der Schreibvorgang abgeschlossen wird. NoSQL liest das, was in diesem Moment an Daten vorhanden ist.

3. Verschiedene Anforderungen an die Skalierung

Durch die Normalisierung und die Transaktionstreue ist die Skalierung eines SQL-basierten Systems in der Regel herausfordernder als bei NoSQL-basierten Systemen. Die Verteilung der Tabellen und Verknüpfungen auf unterschiedliche Systeme wird dadurch komplex, dass Transaktionen erst abgeschlossen werden müssen. Danach können sie auf andere Bereiche repliziert werden. Lesende Zugriffe müssen dabei nicht nur auf den Transaktionsabschluss warten, sondern auch auf die Replikation der Datensätze.

Die Skalierbarkeit von SQL-basierten Systemen erkauft man sich meist mit einer Hypothek auf die etwas länger dauernden Schreibvorgänge. NoSQL-Systeme erzeugen Kopien der Datensätze und verteilen diese über verschiedene Knoten. Sollte sich ein Datensatz ändern, werden die Sammlungen eine nach der anderen abgeändert. Gelesen werden kann immer. In diesem Fall kann es allerdings passieren, dass ein alter und noch nicht geänderter Datensatz gelesen wird, obwohl kurz vorher eine Änderung in das System eingespielt wurde.

4. ACID versus BASE

SQL-basierte Datenbanken sind in allen ihren Strukturen und Transaktionen konsistent. Dadurch, dass die lesenden Abfragen auf die schreibenden Vorgänge warten, ist davon auszugehen, dass die Daten nachvollziehbar (bei jeder Abfrage des Systems) und gleich sind. NoSQL-basierte Systeme können, sollte noch eine Synchronisation stattfinden müssen, unterschiedliche Ergebnisse liefern. Ein gutes Beispiel dafür ist ein dreifacher Reload eines Twitter-Feeds im Webbrowser, der zwei unterschiedliche Ergebnisse liefern kann, je nachdem wo die Abfrage landet.

Welche Datenbank ist nun die richtige?

Aus den bisherigen Betrachtungen lässt sich folgende wichtige Erkenntnis ableiten:

Der Einsatz von Datenbanksystemen in einem Unternehmen sollte nicht davon abhängen, ob dies „strategisch“ so vorgesehen ist, die Entwickler eine bestimmte Vorliebe haben oder das besagte Modell gerade gehypt wird.

ACID und BASE im Vergleich
(Bild: Adacor Hosting)

Vielmehr sollte die Implementierungsform immer auf den spezifischen Anwendungszweck angepasst werden. Dabei sind die Vor- und Nachteile zu beachten, die entstehen, wenn man die Acid-/Base-Situation genau beleuchtet und auf die jeweilige Datenstruktur und Anwendung projiziert. In vielen Projekten werden einige der genannten Systeme parallel zueinander betrieben. Damit wird das Ziel verfolgt, die Vorteile zu nutzen und die Nachteile auszublenden. Genau das ist die richtige Vorgehensweise. So würde ich ein System zur Nutzer-Authentifizierung und -verwaltung eher einem SQL-basierten System zuschreiben und eine Datensatzsammlung, bei der es unerheblich ist, ob sie um zehn Uhr oder zwei Sekunden später über die Applikation ausgeliefert wird, eher dem NoSQL-basierten System.

Was heißt das für Big Data?

Um diese Frage zu beantworten, lohnt es sich eine weitere Frage zu stellen: Ist es sinnvoll ein massive Anzahl einzelner Datensätze in eine SQL-Datenbank zu schreiben und jedes Mal auf den Abschluss der Transaktion zu warten? Die Antwort lautet: nein. Nun wissen Sie, warum Big Data, was eigentlich ein Begriff für massive Datenhaltung ist, häufig als Synonym für NoSQL-Datenbanken genutzt wird.

(ID:44653614)

KARRIERECHANCEN

→<https://www
jobuniverse.
q=Infrastruk

<https://www.it-jobuniverse.de//stellenangebot/it-system-expert-mwd-devops-infrastruktur-mannheim-soluvia-it-services-gmbh-885606>
Soluvia IT-Services GmbH

IT System Expert (m/w/d) DevOps Infrastruktur <https://www.it-jobuniverse.de//stellenangebot/it-system-expert-mwd-devops-infrastruktur-mannheim-soluvia-it-services-gmbh-885606>

in Mannheim | Flexible Arbeitszeit| Homeoffice

<https://www.it-jobuniverse.de//stellenangebot/informatikerin-mathematikerin-ingenieurin-o-ae-wmd-erforschung-und-entwicklung-sicherer-dateninfrastrukturen-zur-optimierung-zukuenftiger-nachhaltiger-schiffsverkehre-oldenburg-deutsches-zentrum-fuer-luft-und-raumfahrt-e-v-889238>
Deutsches Zentrum für Luft- und Raumfahrt e. V.

Informatiker/in, Mathematiker/in, Ingenieur/in o. ä. (w/m/d) Erforschung und Entwicklung sicherer Dateninfrastrukturen zur Optimierung zukünftiger, nachhaltiger Schiffsverkehre <https://www.it-jobuniverse.de//stellenangebot/informatikerin-mathematikerin-ingenieurin-o-ae-wmd-erforschung-und-entwicklung-sicherer-dateninfrastrukturen-zur-optimierung-zukuenftiger-nachhaltiger-schiffsverkehre-oldenburg-deutsches-zentrum-fuer-luft-und-raumfahrt-e-v-889238>

in Oldenburg | Weiterbildung| Flexible Arbeitszeit

<https://www.it-jobuniverse.de//stellenangebot/informatikerinnen-mathematikerinnen-physikerinnen-o-ae-wmd-erforschung-und-entwicklung-von-dateninfrastrukturen-und-optimierungsverfahren-fuer-schiffsverkehre-in-haefen-oldenburg-deutsches-zentrum-fuer-luft-und-raumfahrt-e-v-889236>
Deutsches Zentrum für Luft- und Raumfahrt e. V.

Informatiker/innen, Mathematiker/innen, Physiker/innen o. ä. (w/m/d) Erforschung und Entwicklung von Dateninfrastrukturen und Optimierungsverfahren für Schiffsverkehre in Häfen <https://www.it-jobuniverse.de//stellenangebot/informatikerinnen-mathematikerinnen-physikerinnen-o-ae-wmd-erforschung-und-entwicklung-von-dateninfrastrukturen-und-optimierungsverfahren-fuer-schiffsverkehre-in-haefen-oldenburg-deutsches-zentrum-fuer-luft-und-raumfahrt-e-v-889236>

in Oldenburg | Weiterbildung| Flexible Arbeitszeit

<https://www.it-jobuniverse.de//stellenangebot/fachinformatiker-systemintegration-it-systemadministrator-mwd-mit-dem-schwerpunkt-infrastruktur-koblenz-awk-aussenwerbung-gmbh-879504>
awk AUSSENWERBUNG GmbH

Fachinformatiker Systemintegration / IT-Systemadministrator (m/w/d) mit dem Schwerpunkt Infrastruktur <https://www.it-jobuniverse.de//stellenangebot/fachinformatiker-systemintegration-it-systemadministrator-mwd-mit-dem-schwerpunkt-infrastruktur-koblenz-awk-aussenwerbung-gmbh-879504>

in Koblenz | Weiterbildung| Betr. Altersvorsorge| Flexible Arbeitszeit| Kantine

<https://www.it-jobuniverse.de//stellenangebot/it-systemadministrator-mvz-infrastruktur-2nd-level-mwd-helios-it-service-gmbh-887307>
Helios IT Service GmbH

IT-Systemadministrator MVZ Infrastruktur 2nd Level (m/w/d) <https://www.it-jobuniverse.de//stellenangebot/it-systemadministrator-mvz-infrastruktur-2nd-level-mwd-helios-it-service-gmbh-887307>

in Berlin (+2 weitere Standorte)

<https://www.it-jobuniverse.de//stellenangebot/it-administrator-mwd-it-infrastruktur-microsoft-365-azure-wissen-bhs-corrugated-maschinen-und-anlagenbau-gmbh-878688>
BHS Corrugated Maschinen- und Anlagenbau GmbH

IT-Administrator (m/w/d) IT-Infrastruktur / Microsoft 365 / Azure <https://www.it-jobuniverse.de//stellenangebot/it-administrator-mwd-it-infrastruktur-microsoft-365-azure-wissen-bhs-corrugated-maschinen-und-anlagenbau-gmbh-878688>

in Wissen | Flexible Arbeitszeit