DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

Überwindung der Grenzen relationaler Datenbanken in Big-Data-Umgebungen

Assignment

im Modul

Advanced Database Technology

Im Rahmen der Prüfung zum Bachelor of Science (B. Sc.)

Verfasser:	Manuel Rettig
Kurs:	WWI23B3
Dozentin:	Maria Dertinger
Abgabedatum:	18. März 2025

Selbstständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit mit dem Thema:

Überwindung der Grenzen relationaler Datenbanken in Big-Data-Umgebungen

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Karlsruhe, _____

Manuel Rettig

Inhaltsverzeichnis

1 Einleitung

Die über 50 Jahre alte Technologie durchlief im Laufe der Jahrzehnte mehrere Evolutionszyklen, um mit den Wandelnden Anforderungen mitzuhalten. Zusammen mit mit der Structured Query Language (kurz: **SQL**) bilden relationale Datenbanken bis heute den de facto Standard für das Speichern und Verwalten digitaler Daten und Informationen aller Art und Güte. In der montalich aktualisierten Rangliste der Popularität verschiedener DBMS der Webseite *db-engines.com* sind im März 2025 sieben der 10 meistgenutzten Datenbanken von primär relationaler Natur.[1]

Trotz kontinuierlicher Weiterentwicklungen und Optimierungen hinsichtlich Skalierbarkeit, Performance und Flexibilität stoßen relationale Datenbanken bei bestimmten modernen Anwendungsfällen an ihre Grenzen. In diesem Kontext haben sich alternative Datenbanktechnologien wie NoSQL- und NewSQL-Systeme entwickelt, die speziell für verteilte, flexible oder hochgradig skalierbare Datenverarbeitung konzipiert wurden.

1.1 Relationale Datenbanken und SQL

Das relationale Modell in Verbindung mit der Structured Query Language (SQL) bildet die Grundlage relationaler Datenbankmanagementsysteme (RDBMS) und wurde im Jahr 1970 erstmals durch den Mathematiker Edgar F. Codd konzeptioniert und vorgeschlagen.[2] Seine Forschungsarbeit basiert auf der Mengenlehre, einem Teilgebiet der Mathematik. Eine relationale Datenbank ist demnach eine Menge von Tabellen, den sogenannten Relationen. Das Relationenschema definiert die Anzahl und Art der möglichen Spalten bzw. Attribute einer Tabelle und wird bei Erzeugung der Tabelle festgelegt. Eine Zeile der Tabelle, bildet einen Datensatz. Diese werden zeilenweise in Form von Tupeln in die Tabellen eingefügt und müssen dabei den Regeln des Relationenschemas folgen. Ein weiterer Teil Codd's Forschung war die Entwicklung der relationalen Algebra, deren Operationen die Grundlage der universellen Abfragesprache SQL bilden. Diese kombiniert mengenorientierte Operatoren wie die Vereinigungsmenge oder das Kartesische-Produkt

[1] Red Gate Software Ltd., 2025.
[2] Codd, 1970.

mit den relationenorientierten Operatoren Selektion und Projektion. Die Datenabfrage mit SQL folgt dem SELECT-FROM-WHERE Ausdruck, mit welchem Daten von einer oder mehreren Tabellen unter Selektions- und Filterbedingungen abgefragt und tabellarisch angezeigt werden. Auf Basis dieser Forschungsarbeit entstanden in den folgenden Dekaden zahllose Datenbanksysteme wie beispielsweise IBM's Db2, die Oracle Database und der Microsoft SQL Server.[3]

1.2 Big Data - Große Daten, große Probleme?

Der Begriff Big Data (dt. Große Daten) beschreibt den allgegenwärtigen Trend wachsender Datenmengen. Hierbei liegt der Fokus nicht lediglich auf der Wachsenden Größe einzelner Datensätze wie die Übersetzung vermuten ließe, sondern vielmehr der horizontalen Skalierung und somit einer überwältigenden Anzahl von Datensätzen welche in Echtzeit verarbeitet und gespeichert werden müssen. Passende Beispiele hierfür sind beispielsweise der Datenfluss großer Online-Shops, Soziale-Netzwerke und digitale Streaming-Platformen.[4]

[3] Meier, 2018, S. 15-24.
[4] Meier, 2018, S. 5.

2 Grenzen relationaler Datenbanken

2.1 Eingeschränkte Skalierbarkeit

Das rasante Datenwachstum der letzten zwei Dekaden bedingte die Notwendigkeit kontinuierlich expandierender digitaler Infrastruktur, wie etwa Server und Datenbanken. Die parallel zunehmende Rechenleistung und Speicherkapazitäten einzelner Computerchips und Festplatten konnten diesem Trend nicht kompensieren, sodass die vertikale Skalierung durch Erhöhung der Rechenleistung einzelner Computer an ihre technischen und wirtschaftlichen Grenzen stieß. Um das Problem zu lösen, wurde vermehrt auf die horizontale Skalierung gesetzt, bei der mehrere physikalisch getrennte Rechenknoten zusammenarbeiten. Diese Netzwerke bilden die Grundlage heutiger Rechenzentren und Supercomputer. Die genannten Limitierungen konnten durch Parallelisierung gelöst werden und ermöglichen theoretisch unbegrenzte Kapazität. Darüber hinaus führt die Verteilung zu einer höheren Verfügbarkeit und Ausfallsicherheit. Ist ein Knoten, beispielsweise auf Grund von Wartungsarbeiten nicht erreichbar, wird die verlorene Rechenleistung von anderen Knoten ausgeglichen, ohne merkliche Auswirkungen auf die Anwendung und deren Nutzer. Relationale Datenbanken sind traditionell für die vertikale Skalierung ausgelegt, und speziell optimiert für den Betrieb auf einem einzelnen Server. Die effektive und effiziente (optimale) horizontale Skalierung relationaler Datenbanken ist aufgrund der strengen ACID-Eigenschaften und der Einschränkung des CAP-Theorems nicht, oder nur mit erheblichem Kostenaufwand, möglich.[5] Ein Ansatz war der sogenannte Memcache, wobei die häufigeren Lesezugriffe auf mehrere Replikationsserver verteilt wurden. Beim Sharding werden große Tabellen über mehrere Datenbankserver partitioniert. Es stellte sich jedoch heraus, dass diese komplizierten Ansätze nicht nur viele Nachteile und hohe Kosten verursachen, sondern die notwendige Skalierbarkeit, wie etwa von Sozialen Netzwerken oder Online-Shops benötigt, nicht erreichbar ist.[6]

[5] Schreiner et al., 2019, S. 1.
[6] Harrison, 2015, S. 41-43.

2.2 Fehlende Flexibilität & Impedance Mismatch

Für die Erstellung der Tabellen einer relationalen Datenbank müssen die Spalten und deren Datentypen streng definiert werden. Jede Spalte benötigt dabei einen eindeutig identifizierbaren Primärschlüssel, um die Zeilen voneinander unterscheiden zu können. Komplexe Datenmodelle werden zunächst normalisiert, um Redundanzen zu eliminieren, dabei wird das Datenmodell auf mehrere Tabellen verteilt, und miteinander via Primär- und Fremdschlüsseln verknüpft. Die vollständige Modellierung des Datenmodells kann unterstützten, um die Abhängigkeiten und Zusammenhänge zu visualisieren. Beim Einfügen eines Datensatzes, müssen die Länge genau mit der Anzahl der Spalten und den festgelegen Datentypen übereinstimmen. Ist dies nicht der Fall, lehnt das Datenbankmanagmentsystem die Daten ab, und gibt eine entsprechende Fehlermeldung zurück.

Dieser Ansatz ermöglicht neben weiteren Vorteilen eine hohe Datenkonsistenz, jedoch auf Kosten der Schema-Flexibilität. Der Ansatz der agilen Softwareentwicklung fordert einen schmalen iterativ-dynamischen Entwicklungsprozess, sodass häufige Änderungen oder Erweiterungen des Datenmodell notwendig werden. Moderne RDBMS unterstützten zwar die Modifikation des Schemas bestehender Tabellen, diese erfordern jedoch besondere Vorsicht und erfüllen nicht die geforderte einfache Flexibilität und verlässliche Stabilität, dynamischer Datenmodelle.[7]

[8]

[7] Harrison, 2015, S. 197.
[8] Neward, 2006.

3 Lösungsansätze in Big-Data Umgebungen

Im Big-Data Umfeld wurden die Grenzen relationaler Datenbanken früh erkannt und an entsprechenden Lösungsansätzen gearbeitet. Als größtes Hindernis wurde Distanz zum relationalen Datenmodell und deren strikten Einschränkungen geschaffen.

3.1 NoSQL - flexibel und skalierbar

Der Begriff NoSQL beschreibt eine Familie von Datenbanken welche sich von dem traditionellen relationalen Datenbankmodell distanzieren. In der Literatur wird dieser typischerweise interpretiert mit „not only SQL“, entgegen der intuitive Übersetzung „kein SQL“. Der Unterschied zu relationalen Datenbanken liegt in der Art und Weise wie Daten gespeichert werden. Statt der Bindung an ein fest definiertes Datenschema, sind NoSQL-Datenbanken flexibel. Es wird im Allgemeinen zwischen vier Datenmodellen unterschieden:

1. **Schlüssel-Wert-Datenbanken** speichern Daten als Paarungen aus einem eindeutigen Schlüssel und einem zugehörigen Wert. Der Schlüssel dient zur Identifizierung des Werts, ähnlich wie ein Primärschlüssel in RDBMS.[9]
2. **Spaltenorientierte Datenbanken** speichern Daten in Spaltenfamilien statt, wie bei RDBMS üblich, in Zeilen. Zusammengehörige Daten werden somit aggregiert gespeichert und ermöglichen hohe Skalierbarkeit durch Verteiliung der unterschiedlichen Spaltenfamilien auf verschiedene Knoten.
3. **Dokumentorientierte Datenbanken** verwalten Daten in flexiblen, hierachisch strukturierten Dokumenten, meist im JSON-Format. Jedes Dokument kann eine unterschiedliche Struktur aufweisen, wodurch dynamische und variable Datenmodelle untersützt werden.
4. **Graphdatenbanken** speichern Daten als Knoten und deren Beziehungen als Kanten. Vernetzte Datenstrukturen, bei denen die Beziehungen zischen Datenelementen im Vordergrund stehen, können in Graphdatenbanken effizient abgebildet werden.

[9] Wang und Yang, 2017, S. 2.

Die ersten drei Datenbanktypen unterscheiden sich in ihrer Art, stark von den Graphdatenbanken. Fowler schlägt daher die Teilung der NoSQL-Familie in zwei übergreifende Kategorien vor: Die Aggregat-orientierten Datenbanken, speichern zusammengehörige gemeinsam ab. Hier ist der Unterschied zu relationalen Datenbanken erkennbar, wo zusamenhängende Daten meist durch die Normalisierung auseinandergerissen werden. Graphdatenbanken hingegen, sind spezialisiert auf die noch stärkere Verteilung von Daten welche nicht im direkten Zusammenhang zueinander stehen, sondern lediglich in Bestimmten Situationen oder Anwendung in Verbindung oder Beziehung zueinander stehen.

3.2 Zurück zu SQL - NewSQL

Das zweite Kapitel dieser Arbeit behandelte bereits ausführlich die Limitierungen und Grenzen relationaler Datenbanken, insbesondere in den domänenspezifischen Anforderungen von BigData. Nichts, desto trotz scheinen relationale Datenbanken immernoch der Standard zu sein, trotz aller Nachteile. Der Grund hierfür liegt aller Vermutung nach in der Abfragesprache SQL. Die universelle Datenbankübergreifende Sprache, basierend auf dem relationalen Modell, scheint der Vorteil zu sein, der sämtliche Nachteile in den Schatten stellt. Die chaotische, Abfragesituation bei NoSQL bei welcher jede Datenbankimplementierung eigenen Syntax und Grammatik verwendet wird hierbei nicht hilfreich gewesen sein.

In den letzten Jahren wurden die Froderungen nach einer erneuten Vereinheitlichung der Abfragesprachen lauter. Neue Datenbanken sollten nun, sämtliche Vorteile der NoSQL Ära mit mit den Vorteilen der SQL Äre verbinden. Dieser Ansatz wird heute als NewSQL bezeichnet, und beschreibt den Trend zurück zu SQL oder SQL ähnlicher Abfragesprachen. Die Autoren *Stonebreaker und Pavlo* beschreiben diese Situation treffend in ihrem Artikel aus mit dem Titel *„What Goes Around Comes Around... And Around...“*[10].

[10] Stonebraker und Pavlo, 2024.

3.3 Fallstudie: Datenbankarchitektur eines Online-Shops

In der Vergangenheit war die Wahl der Datenbank unkompliziert - Es musste lediglich die Entscheidung getroffen werden auf welches RDBMS gesetzt werden soll. Doch mit dem Aufkommen der NoSQL Datenbanken wurde diese Entscheidung deutlich schwieriger. Nicht musste von nun an zwischen zwei Datenbankmodellen entschieden werden, sondern auch innerhalb von NoSQL gibt es verschiedene Datenbanktypen optimiert für unterschiedliche Anwendungsseznarien. Auch die Erfahrung der Entwickler spielt nun eine Rolle, da man sich nicht mehr auf die universellen SQL-Kenntnisse verlassen kann. Doch die resultierenden Vorteile kompensierten den erhöhten Aufwand. Während für die allermeisten Anwendungen mit strukturierten Daten, relationale Datenbanken die beste Wahl darstellen, gibt es Anwendungsseznarien wo eine andere Wahl sinnvoller ist. Abbildung 1 zeigt eine beispielhafte Datenbankarchitektur eines großen Online-Shops wie z.B. Amazon.de.

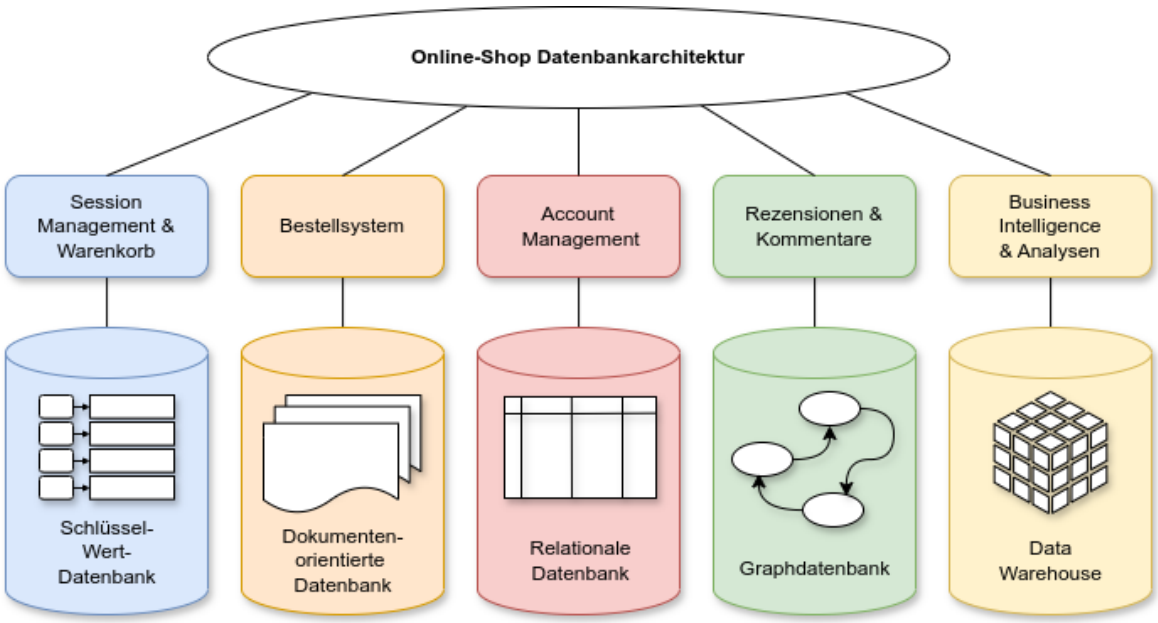

Abbildung 1: Datenbankarchitektur eines Online-Shops, angelehnt an D'Onofrio und Meier, 2021.

Erkennbar ist, das für alle Funktionen verschiedene Datenbankmodelle gewählt wurden. Für die Verwaltung der Nutzersessions und Warenkörbe könnte eine simple Schlüssel-Wert-Datenbank sinnvoll sein. Für das Bestellsystem, ist die Flexibilität und Skalierbar-

keit von NoSQL ebenfalls wichtig, weshalb ein Dokumentspeicher in Betracht zu ziehen ist. Für die Verwaltung der Benutzeraccounts sind die ACID eigenschaften relationaler Datenbanken besonders wichtig. Bei den Produktrezensionen und Kommentaren liegen stark vernetzte Daten vor, weshalb eine Graphdatenbank sinnvoll erscheint. Auch für die Business Intelligence möchte man das passende System verwenden, hier kommt Beispielweise ein Data Warehouse basiernd auf einer Spaltenfamiliendatenbank in Frage.

4 Zusammenfassung

4.1 Fazit

4.2 Ausblick

Quellenverzeichnis

Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. *Commun. ACM*, *13*(6), 377–387. https://doi.org/10.1145/362384.362685

D'Onofrio, S., & Meier, A. (Hrsg.). (2021). *Big Data Analytics: Grundlagen, Fallbeispiele und Nutzungspotenziale*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-32236-6

Harrison, G. (2015). *Next Generation Databases*. Apress. https://doi.org/10.1007/978-1-4842-1329-2

Meier, A. (2018). *Werkzeuge der digitalen Wirtschaft: Big Data, NoSQL & Co.* Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-20337-5

Neward, T. (2006, 26. Juni). *The Vietnam of Computer Science*. Neward & Associates LLC. Verfügbar 2. März 2025 unter http://blogs.newardassociates.com/blog/2006/the-vietnam-of-computer-science.html

Red Gate Software Ltd. (2025, 2. Mai). *DB-Engines Ranking*. DB-Engines. Verfügbar 2. März 2025 unter https://db-engines.com/de/ranking

Schreiner, G. A., Duarte, D., & Mello, R. d. S. (2019). When Relational-Based Applications Go to NoSQL Databases: A Survey. *Information*, *10*(7), 241. https://doi.org/10.3390/info10070241

Stonebraker, M., & Pavlo, A. (2024). What Goes Around Comes Around... And Around... *SIGMOD Rec.*, *53*(2), 21–37. https://doi.org/10.1145/3685980.3685984

Wang, R., & Yang, Z. (2017). SQL vs NoSQL: A Performance Comparison. https://www.cs.rochester.edu/courses/261/fall2017/termpaper/submissions/06/Paper.pdf