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

Relationale Datenbanken und die Abfragesprache SQL bilden seit Jahrzehnten den de facto Standard für das Speichern, Verwalten und Abfragen 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] Um mit den wandelnden Anforderungen mitzuhalten durchlief die über 50 Jahre alte Technologie mehrere Evolutionszyklen. Trotz Optimierungsversuchen hinsichtlich Skalierbarkeit, Performance und Flexibilität stoßen relationale Datenbanken in diesen Bereichen jedoch zunehmend an ihre Grenzen. In diesem Kontext wurden alternative Datenbanktechnologien wie NoSQL- und NewSQL-Systeme entwickelt, die speziell für verteilte, flexible und hochgradig skalierbare Datenverarbeitung ausgelegt sind.[2]

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.[3] Seine Forschungsarbeit basiert auf der Mengenlehre, einem Teilgebiet der Mathematik.[4] 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 mit den

[1] Red Gate Software Ltd., 2025.
[2] **<empty citation>**.
[3] Codd, 1970.
[4] Langer und Mukherjee, 2023, S. 66.

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.[5]

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.[6]

[5] Meier, 2018, S. 15-24.
[6] Meier, 2018, S. 5.

2 Grenzen relationaler Datenbanken

Moderne Big-Data-Anwendungen erfordern skalierbare und flexible Datenspeicher, um mit der wachsenden Menge dynamischer Daten Schritt zu halten. Trotz kontinuierlicher Optimierung stoßen relationale Datenbanken dabei zunehmend an ihre Grenzen. Dieses Kapitel erläutert die Hintergründe dieser Limitierungen und unterstreicht dabei die Notwendigkeit und Einsatzbereiche alternativer Datenbanktechnologien.

2.1 Fehlende Flexibilität & Impedance Mismatch

Bei der Erstellung von Tabellen in einer relationalen Datenbank müssen Spalten und Datentypen in einem Schema definiert werden. Zur Vermeidung von Redundanzen und zur Sicherstellung der Datenintegrität werden komplexe Datenmodelle zunächst normalisiert. Dabei wird das Modell auf mehrere Tabellen verteilt und mit Primär- und Fremdschlüsseln verknüpft, wodurch auch komplexe Objektstrukturen auf einfache Tabellen abbildbar sind. Die strikte Schematisierung stellt sicher, dass nur Datensätze akzeptiert werden, die der definierten Struktur und Vorgaben des Schemas entsprechen. Weicht die Länge oder der Datentyp eines Wertes ab, wird der Datensatz nicht akzeptiert. Die resultierende Datenkonsistenz ist eine der zentralen ACID-Eigenschaften relationaler Datenbanken und maßgeblicht verantwortlich für die Verlässlichkiet und weite Verbreitung dieser Systeme.[7]

Die Anforderungen moderner Software-Anwendungen haben sich im Laufe der Zeit stark gewandelt. Nutzer cloudbasierter Software-as-a-Service-Lösungen fordern kontinuierliche Wartung und Erweiterung ihrer Software. Die heute verbreitete agile Softwareentwicklung setzt hierfür auf einen iterativ-dynamischen Entwicklungsprozess, der häufig Änderungen oder Erweiterungen des Datenmodells erfordert. Zwar unterstützen moderne relationale Datenbanksysteme die Modifikation bestehender Tabellen, doch diese erfordert besondere Vorsicht und bietet nicht der gewünschten Flexibilität und Stabilität.[8] Darüber hinaus sind Big-Data-Anwendungen zunehmend mit den Herausforderung der effizienten Verwaltung riesiger Mengen unstrukturierter oder semi-strukturierter Daten konfrontiert. Große

[7] Phiri und Kunda, 2017, S. 3.
[8] Harrison, 2015, S. 197.

Anbieter haben die strikte Schematisierung des relationalen Modells als einschränkenden Faktor erkannt und suchen nach Alternativen, welche die starren Strukturen aufbrechen vereinfachen.

Ein weiteres Problem in diesem Kontext, das Entwickler schon seit Jahrzehnten beschäftigt, ist der Impedance Mismatch. Der Begriff beschreibt den grundlegenden Unterschied zwischen dem objektorientierten Datenmodell und dem relationalen Modell. Während Anwendungen häufig mit komplexen Objektstrukturen arbeiten, die Vererbung und Typisierung unterstützen, basieren RDBMS auf einfachen Tabellenstrukturen. Dieser Modellkontrast führt zu einem hohen Übersetzungsaufwand zwischen Anwendung und Datenbank.[9] Die Persistierung komplexer Objekte in relationaler Form lässt sich mit einem Parkhaus vergleichen, in dem Autos in ihre Einzelteile zerlegt gelagert werden. Jeder Lese- oder Schreibzugriff erfordert das Zusammensetzen beziehungsweise Zerlegen der Objekte. Reine Objektdatenbanken, die dieses Problem lösen sollten, konnten sich nicht durchsetzen, sodass moderne RDBMS heute objektorientierte Konzepte unterstützen. Auch der Einsatz sogenannter objektrelationaler Abbildungen (ORM) kann helfen, die Auswirkungen des Impedance Mismatch zu minimieren, indem er den Übersetzungsschritt automatisieren. Das grundlegende Problem jedoch bleibt: Objektstrukturen sowie semi-strukturierter Daten sind eine grundlegende Schwäche relationaler Datenbanken.

2.2 Eingeschränkte Skalierbarkeit

Das rasante Datenwachstum der letzten zwei Dekaden bedingte die Notwendigkeit kontinuierlich expandierender digitaler Infrastruktur. 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. Die genannten Limitierungen konnten durch Parallelisierung gelöst werden und ermöglichen theoretisch unbegrenz-

[9] Neward, 2006.

te 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. Mit den wachsenden Anforderungen wurden jedoch Ansätze wie entwickelt, um auch die horizontale Skalierbarkeit zu ermöglichen: Der sogenannte Memcache verlagerte die typischerweise häufiger auftretenden Lesezugriffe auf mehrere Replikationsserver. Zusätzlich wurden beim Sharding gesamte Datenbanken partitioniert und auf mehrere Knoten verteilt.[10]. Das CAP-Theorem besagt, kein verteiltes System könne gleichzeitig **C**onsistency (Konsistenz), **A**vailability (Verfügbarkeit) sowie **P**artition Tolerance (Ausfalltoleranz) erfüllen. Die Ausfalltoleranz ist bei in verteilten Systemen stets besonders wichtig, da sonst der Ausfall eines Knoten das gesamte System beeinträchtigt. Demnach muss in diesem Fall zwischen Verfügbarkeit und Konsistenz gewählt werden - Weil die Verfügbarkeit meist Priotität hat, kann eine grundlegende ACID-Eigenschaft bei der Verteilung relationaler Datenbanken nicht mehr garantiert werden.

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.[11]

Die ACID-Eigenschaften werden bei großen Datenmengen zu einem zunehmend limitierenden Hindernis, welches.

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.[12].

[10] Dertinger, 2025, S. 16.
[11] Schreiner et al., 2019, S. 1.
[12] Harrison, 2015, S. 41-43.

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.[13]
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.

[13] 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.[14]

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...“*[15].

[14] Fowler, 2012.
[15] 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.

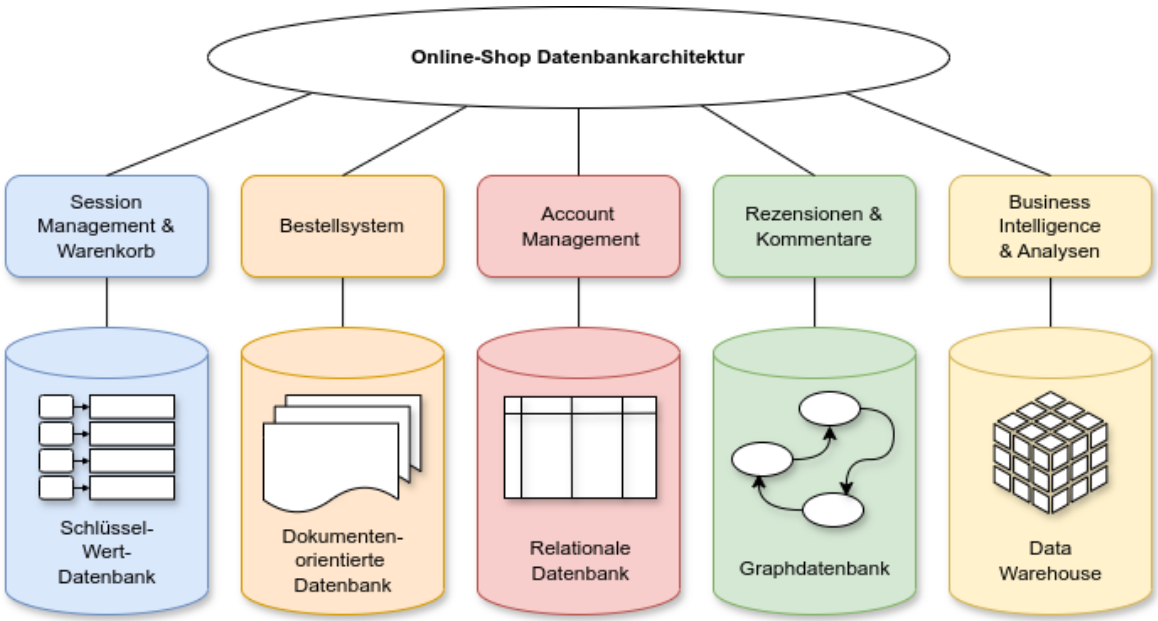

Abbildung 1: Datenbankarchitektur eines Online-Shops, angelehnt an D'Onofrio und Meier, 2021.

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.

Polyglot Persitstance

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

Dertinger, M. (2025). *NoSQL-Datenbanken Und Nicht-relationale Datenbanken* (Vorlesung). DHBW Karlsruhe.

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

Fowler, M. (2012, 19. Januar). *Aggregate Oriented Database*. martinfowler.com. Verfügbar 6. März 2025 unter https://martinfowler.com/bliki/AggregateOrientedDatabase.html

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

Langer, A., & Mukherjee, A. (2023). *Developing a Path to Data Dominance: Strategies for Digital Data-Centric Enterprises*. Springer International Publishing. https://doi.org/10.1007/978-3-031-26401-6

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

Phiri, M. H., & Kunda, D. D. (2017). A Comparative Study of NoSQL and Relational Database. *1*(1).

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