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 zunehmend an ihre Grenzen. In diesem Kontext wurden alternative Datenbanktechnologien 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 relationenorientierten Operatoren Selektion und Projektion. Die Datenabfrage mit SQL

[1] Red Gate Software Ltd., 2025.
[2] Harrison, 2015, S. 13-17.
[3] Codd, 1970.
[4] Langer und Mukherjee, 2023, S. 66.

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 gegenwärtigen Trend stetig wachsender Datenmengen. Eine einheitliche Definition existiert in der Literatur nicht. Stattdessen werden häufig lediglich charakteristischen Merkmale anhand der drei V's erläutert: Eine enorme Datenmenge (*Volume*) aus unterschiedlichen Quellen und Formaten (*Variety*) muss in hoher Geschwindigkeit (*Velocity*) verarbeitet und gespeichert werden.[6] Typische Anwendungsfälle sind große E-Commerce-Plattformen, soziale Netzwerke und digitale Streaming-Dienste. Neuere Definitionen von Big Data beziehen zudem den Unternehmenswert (Value) mit ein, der den wirtschaftlichen Nutzen und die Relevanz solcher Technologien unterstreicht. Die Extraktion dieses Mehrwerts aus der Datenflut ist eine zentrale Herausforderung und wird treffend unter dem Motto „Extracting Value from Chaos" zusammengefasst.[7] Bereits im Jahr 2011 definierten Manyika et al. in einem Bericht des McKinsey Global Institute Big Data als Datenmengen, die die Verarbeitungsgrenzen herkömmlicher Datenbanksysteme überschreiten. Sie verzichteten dabei bewusst auf die Definition einer quantitative Grenze – etwa in Terabyte oder Petabyte – in der Annahme, dass sich diese Schwelle mit dem technologischen Fortschritt stetig verschiebt.[8] Die größten Technologieunternehmen wie Google und Facebook, aber auch der Versandgigant Amazon wurden durch wachsende Datenmengen schnell mit den Grenzen und Limitierungen traditioneller Datenbanken konfrontiert. Um Big Data effektiv und gewinnbringend einzusetzen arbeiteten sie an innovativen Lösungen und Ansätzen, was die Datenspeicherung nachhaltig veränderte und eine neue Generation von Datenbanken einleitete.

[5] Meier, 2018, S. 15-24.
[6] Meier, 2018, S. 5.
[7] Chen et al., 2014, S. 173 ff.
[8] Manyika et al., 2011.

2 Grenzen relationaler Datenbanken

Moderne Big-Data-Anwendungen erfordern skalierbare und flexible Datenspeicher, um mit der wachsenden Menge dynamischer Daten Schritt zu halten, relationale Datenbanken stoßen hier zunehmend an ihre Grenzen.[9] Dieses Kapitel erläutert die Hintergründe dieser Limitierungen und unterstreicht die Notwendigkeit 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.[10]

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.[11] Darüber hinaus sind Big-Data-Anwendungen zunehmend mit den Herausforderung der effizienten Verwaltung riesiger Mengen unstrukturierter oder semi-strukturierter Daten konfrontiert. Große

[9] Schreiner et al., 2019, S. 1.
[10] Phiri und Kunda, 2017, S. 3.
[11] 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.[12]

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.[13] 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) hilft die Auswirkungen des Impedance Mismatch zu minimieren, indem er den Übersetzungsschritt automatisieren. Das grundlegende Problem jedoch bleibt: Komplexere Objektstrukturen sowie semistrukturierter Daten sind eine grundlegende Schwäche relationaler Datenbanken.[14]

2.2 Eingeschränkte Skalierbarkeit

Das rasante Datenwachstum der letzten zwei Dekaden bedingte die Notwendigkeit kontinuierlich expandierender digitaler Infrastruktur zur Verwaltung und Analyse dieser Datenmengen.[15]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. Zur Lösung des Problems wurde vermehrt auf

[12] Chen et al., 2014, S. 175.
[13] Neward, 2006.
[14] Harrison, 2015, S. 13.
[15] Petroc, 2024.

die horizontale Skalierung gesetzt, wobei physikalisch getrennte Rechenknoten zusammenarbeiten. 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, z.B. auf Grund von Wartungsarbeiten nicht erreichbar, wird die verlorene Rechenleistung von anderen Knoten ausgeglichen, ohne merkliche Auswirkungen auf die Anwendung und deren Nutzer.[16]

Relationale Datenbanken sind traditionell für die vertikale Skalierung und den Betrieb auf einem einzelnen Server ausgelegt.[17] 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.[18] 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 priorisiert wird, kann eine grundlegende ACID-Eigenschaft bei der Verteilung RDBMS nicht mehr garantiert werden.[19]

Die effektive horizontale Skalierung relationaler Datenbanken ist also aufgrund der strengen ACID-Eigenschaften und Einschränkungen des CAP-Theorems nur eingeschränkt bzw. mit erheblichem Kostenaufwand möglich.[20] Besonders bei großen Datenmengen werden die ACID-Eigenschaften zunehmend zu einem limitierenden Faktor. In der Praxis zeigte sich, dass viele Skalierungsansätze nicht nur hohe Kosten und zusätzliche Komplexität verursachen, sondern auch die für Anwendungen wie soziale Netzwerke oder Online-Shops erforderliche Skalierbarkeit nicht gewährleisten können.[21]

[16] Uyanga et al., 2021, S. 120.
[17] Sahatqija et al., 2018.
[18] Dertinger, 2025, S. 16.
[19] Meier, 2018, S. 33 ff.
[20] Schreiner et al., 2019, S. 1.
[21] 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, die sich von dem traditionellen relationalen Datenbankmodell abgrenzen. In der Literatur wird dieser typischerweise als „not only SQL“ interpretiert, entgegen der intuitiven Übersetzung „kein SQL“. Der Unterschied zu relationalen Datenbanken liegt in der Art der Datensepicherung: Anstelle eines fest definierten Schemas bieten NoSQL-Datenbanken eine flexible Struktur. Grundsätzlich wird zwischen vier Typen von NoSQL-Datenmodellen unterschieden.[22]

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.[23]
2. **Spaltenorientierte Datenbanken** speichern Daten in Spaltenfamilien statt, wie bei RDBMS üblich, in Zeilen. Zusammengehörige Daten werden aggregiert gespeichert und ermöglichen hohe Skalierbarkeit durch Verteiliung der unterschiedlichen Spaltenfamilien auf verschiedene Knoten.[24]
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.[25]

[22] Schreiner et al., 2019, S. 1 f.
[23] Wang und Yang, 2017, S. 2.
[24] **<empty citation>**.
[25] **<empty citation>**.

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.[26]

3.2 Fallbeispiel: Polyglot Persistence

Die Wahl eines geeigneten Datenbanksystems war in der Vergangenheit unkompliziert, da sie sich ausschließlich auf die Entscheidung für eines der verfügbaren relationalen Datanbanksystems beschränkte. Mit dem Aufkommen von NoSQL-Datenbanken wurde diese Entscheidung jedoch deutlich komplexer. Neben der grundlegenden Wahl zwischen relationalen und nicht-relationalen Datenbanken muss nun auch innerhalb der NoSQL-Landschaft zwischen verschiedenen Datenbanktypen abgewogen werden. Zudem spielt die Erfahrung der Entwickelnden eine größere Rolle, da NoSQL-Datenbanken meist keine universelle Abfragesprache wie SQL unterstützen. Während relationale Datenbanken für die Mehrheit der Anwendungen mit strukturierten Daten weiterhin die bevorzugte Wahl darstellen, gibt es Szenarien, in denen NoSQL-Lösungen besser geeignet sind.[27] Der Einsatz verschiedener Datenbanktypen für verschiedene Anwendungsfälle wird als **Polyglot Persistence** bezeichnet. Das von *Fowler* popularisierte Architekturmuster steht kontrastierend zum *„one-size-fits all“*-Prinzip, das eine monolithische und anwendungsübergreifende Datenbankarchitektur anstrebt. Stattdessen verfolgt Polyglot Persistence das Ziel, für jede Anwendung die am besten geeigneten Datenbanksysteme zu wählen. Vorteile sind die gesteigerte Produktivität und die Reduktion des Impedance Mismatch. Beispielsweise eignen sich Dokumentendatenbanken besonders für die Persistierung komplexerer und semi-strukturierter Objekte, während für strukturierte Daten weiterhin die Vorteile

[26] Fowler, 2012.
[27] Harrison, 2015, S. 194.

relationaler Datenbanken genutzt werden können.[28] Abbildung 1 skizziert die Datenbankarchitektur einer E-Commerce Plattform basierend auf der Polyglot-Persistence.

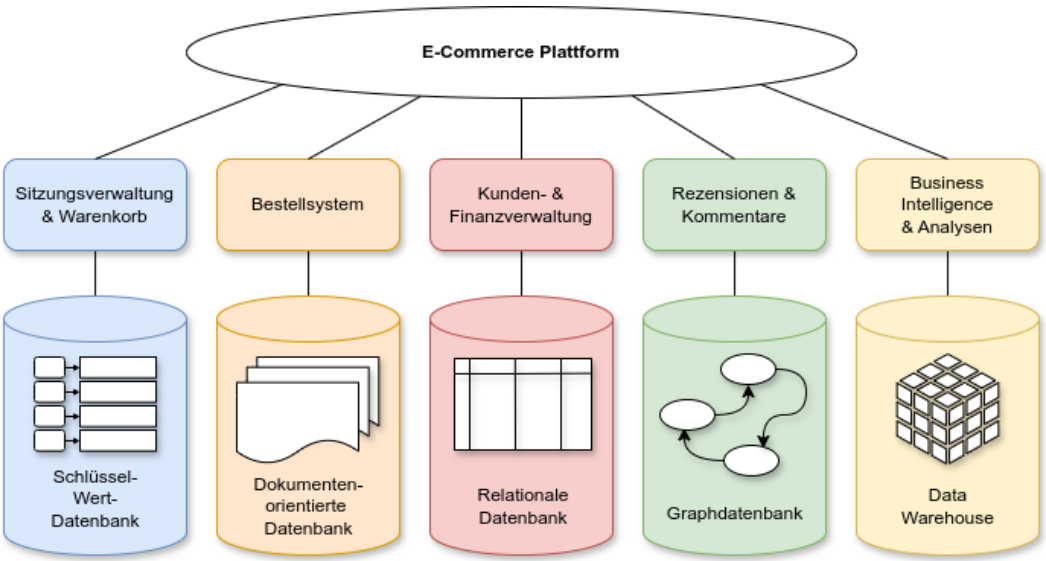

Abbildung 1: Beispielhafte Datenbankarchitektur einer modernen und skalierbaren E-Commerce Plattform. In Anlenung an D'Onofrio und Meier, 2021.

Erkennbar ist, das für die unterschiedlichen Funktionen verschiedene Datenbankmodelle gewählt wurden. Für die Verwaltung der Nutzersitzungen und Warenkörbe könnte eine einface Schlüssel-Wert-Datenbank sinnvoll sein, da sie hohe Verfügbarkeit und Ausfalltoleranz gewährleistet. Das Bestellsystem erfordert ebenfalls hohe Verfügbarkeit sowie Flexibilität und Skalierbarkeit, weshalb die Verwendung eines Dokumentenspeichers in Betracht zu ziehen ist. Die Verwaltung von Kundendaten und Transaktionen in der Finanzverwaltung stellt hohe Anforderungen an Konsistenz, weshalb relationale Datenbanken mit ihren ACID-Eigenschaften besonders geeignet erscheinen. Bei Produktrezensionen, Kommentaren und Interaktionen handelt es sich um vernetzte Daten, für die sich eine Graphdatenbank anbietet, da sie gezielte Auswertungen von Kundenbeziehungen ermöglicht. Für Business-Intelligence-Aufgaben wie Aggregationen, Analysen und die Bereitstellung aktueller Kennzahlen empfiehlt sich der Betrieb eines Data Warehouses.[29] [30]

[28] Gessert und Ritter, 2015, S. 1 f.
[29] D'Onofrio und Meier, 2021, S. 8.
[30] Meier, 2018, S. 47.

3.3 NewSQL & Die Datenbanken der Zukunft

Während Polyglot Persistence eine Möglichkeit bietet, die Vorteile verschiedener Datenbanktypen zu kombinieren, bringt dieser Ansatz auch Herausforderungen mit sich. Dazu zählen die Wahl der optimalen Datenbank für einen Anwendungsfall sowie Konsistenzprobleme durch unterschiedliche Datenmodelle und Abfragesprachen.[31] Die ideale Lösung wäre eine Brücke zwischen RDBMS und NoSQL-Systemen, die die jeweiligen Stärken beider Technologien vereint. Ein solcher Ansatz existiert bereits unter dem Begriff NewSQL und beschreibt relationale Datenbanken, die Skalierbarkeit und Flexibilität moderner NoSQL-Technologien integrieren. Obwohl sich reine NewSQL-Datenbanken noch nicht durchsetzen konnten, argumentieren Stonebraker und Pavlo, dass sie die Konvergenz zwischen relationalen und nicht-relationalen Systemen vorangetrieben haben. Insbesondere zeigte NewSQL, dass sich verteilte NoSQL-Architekturen mit ACID-Eigenschaften kombinieren lassen.[32]

Anstelle eines Kompromisses zwischen inkompatiblen Technologien skizzierte Harrison (2015) die Datenbank der Zukunft als ein einheitliches System, das sich flexibel an verschiedene Anforderungen anpassen kann. Ein solches Modell müsste sowohl die Normalform relationaler Systeme als auch die Speicherung komplexer Objektstrukturen in Dokumenten ermöglichen.[33] Heute sind wir dieser Vision näher denn je: NoSQL-Datenbanken setzen zunehmend auf ACID-Transaktionen und bieten SQL-kompatible Schnittstellen. Gleichzeitig adaptieren RDBMS Konzepte des nicht-relationalen Modells und unterstützen die Speicherung von Schlüssel-Wert-Paaren und Dokumentstrukturen.[34] Während Multi-Modell-Datenbanken die Grenzen zwischen RDBMS, NoSQL und NewSQL zunehmend auflösen, bleibt das erweiterte relationale Modell in Verbindung mit modernem SQL als zentrale Konstante bestehen – gemäß dem Prinzip:

„What Goes Around Comes Around... And Around...“ [35]

[31] Gessert und Ritter, 2015, S. 3 f.
[32] Stonebraker und Pavlo, 2024, S. 29 ff.
[33] Harrison, 2015, S. 195–202, S. 214.
[34] Stonebraker und Pavlo, 2024, S. 22 ff.
[35] Stonebraker und Pavlo, 2024.

4 Zusammenfassung und Ausblick

Diese Arbeit hat die Grenzen relationaler Datenbanken im Kontext wachsender Datenmengen in modernen Big-Data-Anwendungen untersucht. Während relationale Datenbanken lange Zeit das Standardwerkzeug für die Datenspeicherung waren, führten Skalierbarkeits- und Flexibilitätsprobleme zur Entwicklung alternativer Systeme. NoSQL-Datenbanken wurden eingeführt, um diese Herausforderungen zu bewältigen, brachten jedoch neue Probleme, wie fehlende Standardisierung und Konsistenz, mit sich. Als ein hybrider Ansatz entstand NewSQL, welcher die Vorteile relationaler Datenbanken mit der Skalierbarkeit von NoSQL kombiniert. Heute zeigt sich eine zunehmende Konvergenz zwischen RDBMS, NoSQL und NewSQL. Die zunehmende Adaption von ACID-Transaktionen in NoSQL-Systemen sowie die Erweiterung relationaler Datenbanken um Schema-Flexibilität verdeutlichen, dass sich die Grenzen zwischen den Technologien immer weiter auflösen. Dieser Wandel zeigt, dass keine einzelne Lösung für alle Anwendungsfälle optimal ist – vielmehr ist die Wahl des passenden Datenbankmodells stark von den spezifischen Anforderungen einer Anwendung abhängig.

Die Zukunft der Datenbanken wird von hybriden Architekturen geprägt, die verschiedene Modelle innerhalb eines Systems vereinen. Insbesondere die rasante Entwicklung von Big Data und künstlicher Intelligenz könnte die Datenbanktechnologien in den nächsten Jahren weiter vorantreiben. Langfristig könnten universelle Datenbanken entstehen, die sich flexibel an unterschiedliche Anwendungsfälle anpassen und sowohl relationale als auch nicht-relationale Strukturen effizient verwalten. Die bereits existierenden Multi-Modell-Datenbanken bilden hierfür ein solides Fundament, welches es zu perfektionieren gilt.

Quellenverzeichnis

Chen, M., Mao, S., & Liu, Y. (2014). Big Data: A Survey. *Mobile Networks and Applications*, *19*(2), 171–209. https://doi.org/10.1007/s11036-013-0489-0

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 Advanced Database Technology). Duale Hochschule 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). *Aggregate Oriented Database*. martinfowler.com. Verfügbar 6. März 2025 unter https://martinfowler.com/bliki/AggregateOrientedDatabase.html

Gessert, F., & Ritter, N. (2015). Polyglot Persistence. *Datenbank-Spektrum*, *15*(3), 229–233. https://doi.org/10.1007/s13222-015-0194-1

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

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (2011). *Big Data: The next Frontier for Innovation, Competition, and Productivity*. McKinsey Global Institute. Verfügbar 8. März 2025 unter https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/big-data-the-next-frontier-for-innovation

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). *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

Petroc, T. (2024). *Amount of Data Created, Consumed, and Stored 2010-2023, with Forecasts to 2028* (Statista). Verfügbar 8. März 2025 unter https://www.statista.com/statistics/871513/worldwide-data-created/

Phiri, H., & Kunda, D. (2017). Comparative Study of NoSQL and Relational Database. *Zambia ICT Journal*, *1*, 1–4. https://doi.org/10.33260/zictjournal.v1i1.8

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

Sahatqija, K., Ajdari, J., Zenuni, X., Raufi, B., & Ismaili, F. (2018, 1. Mai). *Comparison between Relational and NOSQL Databases*. https://doi.org/10.23919/MIPRO.2018.8400041

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

Uyanga, S., Munkhtsetseg, N., Batbayar, S., & Bat-Ulzii, S. (2021). A Comparative Study of NoSQL and Relational Database. In J.-S. Pan, J. Li, K. H. Ryu, Z. Meng & A. Klasnja-Milicevic (Hrsg.), *Advances in Intelligent Information Hiding and Multimedia Signal Processing* (S. 116–122, Bd. 212). Springer Singapore. https://doi.org/10.1007/978-981-33-6757-9_16

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