
STUDIENARBEIT

# Vergleich von NoSQL-Konzepten in Oracle und Microsoft SQL Server

## Datenbanksysteme

vorgelegt von

**Tobias Lindner**
www.tobiaslindner.eu

**München, den 30. Juni 2015**

Hochschule für angewandte Wissenschaften München
Fakultät für Informatik und Mathematik
Masterstudiengang Wirtschaftsinformatik

Betreuer: Prof. Dr. Rainer Schmidt

# Ehrenwörtliche Erklärung

Ich versichere hiermit, dass ich meine Studienarbeit mit dem Thema

**Vergleich von NoSQL-Konzepten in Oracle und Microsoft SQL Server**

selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

München, 30.06.2015 _____

# Kurzfassung

Die Speicherung, Verarbeitung und Analyse von Daten zählt heutzutage für viele Unternehmen zum (geschäftskritischen) Tagesgeschäft. Zudem werden immer mehr Unternehmen mit der Anforderung konfrontiert, große Mengen an nicht-relationalen, unstrukturierten und semistrukturieren Daten in Form von Dokumenten, Sonsordaten, Audiodateien, Bildern oder Videos zu Speichern und zu Verarbeiten. Die großen Datenbankhersteller, u.a. Oracle und Microsoft, haben die Notwendigkeit von NoSQL-Konzepten bereits frühzeitig erkannt und diese kontinuierlich in ihre Datenbanksysteme integriert. Die Analyse von ausgewählten Datenbanksystemen hat aufgezeigt, das die Systeme von Oracle den Systemen von Microsoft meist voraus sind - vor allem hinsichtlich der Unterstützung verschiedenster nicht-relationaler Datenmodelle sowie der SQL-Integration. Darüber hinaus sind die Ansätze der neuen Enterprise-Big-Data-Lösungen beider Firmen, die eine einheitliche und transparente Zusammenarbeit von Hadoop-, NoSQL- und SQL-Technologien ermöglichen sollen, vielversprechend. Im zweiten Teil der Studienarbeit wurde eine erste Version eines Kriterienkatalogs entwickelt, der Anwendern und Entscheidern dabei helfen soll, das für einen gegebenen Anwendungsfall passende Datenbanksystem hinsichtlich der Unterstützung von NoSQL-Konzepten auszuwählen.

# Inhaltsverzeichnis

# Abbildungsverzeichnis



# Tabellenverzeichnis

# 1 Einleitung

## 1.1 Motivation

Heutige NoSQL-Datenbanken und deren Konzepte sind das Ergebnisse der Entwicklung der Enterprise-Anwendungsarchitektur seit den 1990er Jahren [1]: Aufbauend auf der zweischichtigen Client-Server-Architektur aus den 80er Jahren (sog. two-tier-architecture) entwickelte sich durch das Aufkommen von Webanwendungen und verteilten Systemen eine drei- und mehrschichtige Architektur (three- oder multi-tier-architecture genannt). Parallel dazu sind – bedingt durch das enorme Datenwachstum sowie die Notwendigkeit, semistrukturierte und unstrukturierte Daten in Echtzeit zu verarbeiten – die Anforderungen an die Architektur und das Design von Datenbanken stark gestiegen. Von neuen Konzepten zur Datenverarbeitung und -analyse, wie beispielsweise Map/Reduce [2] oder Apache Hadoop [3], über die Einführung des neuen, weniger strikten Transaktionskonzepts BASE und die Nutzung von schemafreien, nicht-relationalen Datenmodellen bis hin zur horizontalen und verteilten Skalierung von Datenbanken – heutige Unternehmen aller Branchen sind dazu gezwungen, sich mittel- und langfristig mit diesen NoSQL-Konzepten auseinander zu setzen.

## 1.2 Problemstellung und -abgrenzung

Die Speicherung, Verarbeitung und Analyse von Daten gehört für viele Unternehmen zum (geschäftskritischen) Tagesgeschäft. Daher haben diese im Laufe der letzten Jahrzehnte viel Zeit und Geld in den Aufbau komplexer und relationaler Datenverwaltungssysteme und Anwendungen investiert. Allerdings werden heutzutage immer mehr Unternehmen mit der Anforderung konfrontiert, große Mengen an nicht-relationalen, unstrukturierten und semistrukturieren Daten in Form von Dokumenten, Sonsordaten, Audiodateien, Bildern oder Videos zu Speichern und zu Verarbeiten. Da klassische, relationale Datenbanken auch heute noch häufig als Allzweckwerkzeug ("One size fits all"; [4], [5]) angesehen werden, wird nicht selten viel Zeit investiert, um diese nicht-relationalen Daten auf das relationale Datenmodell abzubilden. In solchen Fällen sollten jedoch alternative Datenmodelle in Erwägung gezogen werden, da diese gegebenenfalls besser zu den zugrundeliegenden Daten und Anforderungen passen (Hecht, Jablonski: "Use the right tool for the job" [6]).

In diesem Zusammenhang werden daher in dieser Studienarbeit ausgewählte Datenbanksysteme aus dem Oracle und Microsoft Umfeld hinsichtlich ihrer Unterstützung bzgl. NoSQL-Konzepten näher untersucht. Konkret handelt es sich hierbei um folgende Datenbanksysteme: Oracle Database (häufig auch "Oracle RDBMS: Relational Database Management System" bezeichnet), Oracle NoSQL Database und Oracle Big Data Management System sowie Microsoft SQL Server und Microsoft Big Data Solutions. Dabei wird aufgrund der knappen zur Verfügung stehenden Zeit für die Ausarbeitung der Arbeit (2 ECTS, entspricht ca. 60 Stunden) vor allem auf die Unterstützung der einzelnen Systeme hinsichtlich der Speicherung und Verarbeitung von nicht-relationalen, unstrukturierten und semistrukturieren Daten eingegangen. Auf weitere NoSQL-Konzepte, wie beispielsweise Map/Reduce-Unterstützung, CAP und BASE oder die horizontale und verteilte Skalierung, kann nur am Rande oder überhaupt nicht eingegangen werden. Auch auf die Implementierung verschiedener Szenarien sowie der anschließenden Messung und Gegenüberstellung der Ausführungsgeschwindigkeiten muss aufgrund der begrenzten Zeit verzichtet werden.

## 1.3 Ziel der Arbeit

Ziel dieser Studienarbeit ist es, einen Überblick über die unterstützten NoSQL-Konzepte in ausgewählten Datenbanken von Oracle und Microsoft zu liefern sowie die jeweilige Umsetzung dieser Konzepte näher zu beleuchten und anschließend gegenüber zu stellen bzw. zu vergleichen. Wie bereits im vorherigen Kapitel erwähnt, wird vor allem die Unterstützung hinsichtlich der Speicherung, Verarbeitung und Analyse von nicht-relationalen, unstrukturierten und semistrukturieren Daten untersucht.

Zudem soll eine erste Version eines Kriterienkatalogs entwickelt werden, der Anwendern und Entscheidern dabei helfen soll, das für einen gegebenen Anwendungsfall passende Datenbanksystem auszuwählen. Um die spezifischen Anforderungen eines jeden einzelnen Anwendungsfalls berücksichtigen zu können, basiert die Bewertung eines Datenbanksystems auf der individuellen Gewichtung der verschiedenen Auswahlkriterien.

## 1.4 Vorgehen

Um einen ersten Überblick über die zu vergleichenden Datenbanksysteme zu erhalten, werden diese im ersten Teil der Studienarbeit zunächst im Allgemeinen untersucht und anschließend spezieller hinsichtlich der Implementierung von NoSQL-Konzepten analysiert. Da vor allem die Unterstützung bezüglich der Speicherung, Verarbeitung und Analyse von nicht-relationalen, unstrukturierten und semistrukturieren Daten innerhalb eines relationalen Datenbanksystems betrachtet werden soll, stehen die Mächtigkeit und Flexibilität des Datenmodells, die Anfragemöglichkeiten sowie die SQL-Integration im Vordergrund.

Im zweiten Teil der Arbeit, dem Vergleich der Datenbanksysteme, wird zunächst die erste Version eines Kriterienkataloges entwickelt und anschließend für den Vergleich der Systeme in Auszügen exemplarisch angewandt. Für die Erstellung des Kriterienkataloges werden zuerst die entscheidungsrelevanten Auswahlkriterien identifiziert und klassifiziert sowie eine Bewertungsskala und Gewichtungsfaktoren definiert. Anschließend erfolgt auf Basis der Analyseergebnisse aus dem ersten Teil die Bewertung der verschiedenen Datenbanksysteme. Abschließend werden die Ergebnisse zusammengefasst sowie ein kleiner Ausblick gegeben.

# 2 Analyse der Datenbanksysteme

## 2.1 Oracle

Unter dem Dach von Oracle werden eine Vielzahl verschiedener Datenbanksysteme zusammengefasst. Diese verschiedenen Systeme können allerdings nicht als einfache Alternativen, die beliebig untereinander austauschbar sind, verstanden werden, sondern vielmehr als einander ergänzende Systeme mit individuellen Stärken und Schwächen und somit spezifischen Einsatzmöglichkeiten, die zusammen ein integriertes Ökosystem bilden.

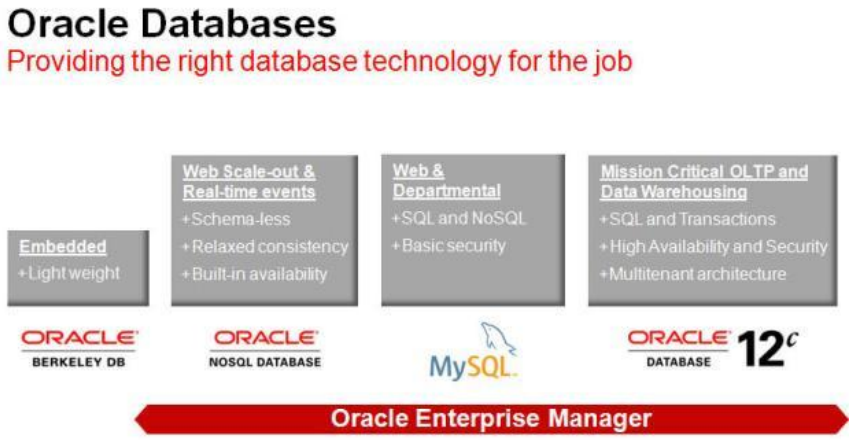


Abbildung 1: Verschiedene Datenbanksysteme von Oracle [7]

In dieser Studienarbeit werden konkret die Datenbanksysteme Oracle Database, Oracle NoSQL Database und Oracle Big Data Management System hinsichtlich der Unterstützung von NoSQL-Konzepten untersucht.

### 2.1.1 Oracle Database

Das Datenbankmanagementsystem Oracle Database (aktuell in der Version 12c) unterstützt neben der Speicherung und Verarbeitung von relationalen und objektrelationalen Daten auch intelligente Datentypen und optimierte Datenstrukturen für den Umgang mit großen Mengen an unstrukturierten und semistrukturierten Daten. So ist es u.a. möglich, JSON und XML Dokumente, Multimediadaten, Graphen, Fließtext und Standortdaten zu analysieren und zu verarbeiten [8].

***Oracle JSON Document Store:*** Der Ansatz der schemalosen Entwicklung ("data-first, schema-later-or-never", [9]) gewinnt heutzutage immer mehr an Bedeutung. Dies zeigt auch die steigende Popularität von dokumentenorientierten NoSQL-Datenbanken wie MongoDB oder CouchDB. Mit dem JSON Document Store [9] ermöglicht Oracle eine vollständig in die Oracle Database Plattform integrierte Verarbeitung, Speicherung und Analyse von dokumentenorientierten Daten im standardisierten JSON-Format (Javascript Object Notation). Die klassischen CRUD-Befehle sowie die Erstellung von Indizes ist möglich. Der Zugriff erfolgt wahlweise über SQL-Befehle oder über die eigens entwickelte API namens Simple Oracle Document Access (SODA). Fortgeschrittene Funktionalitäten wie Replikation, Kompression, Verschlüsselung und Notfallwiederherstellung werden vollständig unterstützt; Transaktionssicherheit wird mit kleinen Ausnahmen durchgehend gewährleistet.

***Oracle XML DB:*** Oracle XML DB [10] bietet eine hochperformante und vollständig in die Oracle Database Plattform integrierte Möglichkeit, XML-Daten zu verarbeiten, zu speichern und zu analysieren. Oracle XML DB unterstützt dabei die wichtigsten XML-Standards des W3C. Der Zugriff

auf die Daten erfolgt wahlweise über die standardisierte SQL/XML-Erweiterung, über XQuery und XPath oder über den Java-Standard XQJ. Neben den CRUD-Befehlen ist auch die Erstellung von strukturierten, unstrukturierten sowie Volltext-Indizes möglich. Die Speicherung erfolgt in drei verschiedenen Varianten; der neu eingeführte, native Binary XML Storage ermöglicht hierbei eine Kompression von 50 bis 75 Prozent. Replikation und Partitionierung werden unterstützt.

***Oracle Text:*** Bei Oracle Text [11] handelt es sich um eine vollständig in die Oracle Database Plattform integrierte Such-, Filter- und Analysefunktion für textuelle Daten. Neben unformatierten Text werden auch viele bekannte und weit verbreitete Dateiformate, wie beispielsweise PDF-Dateien, Microsoft Office Dokumente sowie HTML und XML unterstützt. Oracle Text ist mehrsprachenfähig und bietet eine selbständig Erkennung der Sprache sowie eine Ähnlichkeitssuche, die sprachspezifische Eigenschaften automatische berücksichtigt (z.B. bei "München", "Muenchen" und "Munchen"). Funktionen von Oracle Text können in SQL-Anfragen eingebunden werden. Neben einer Volltext-Indizierung werden noch weitere, für spezifischere Anwendungsfälle geeignete, Indizes angeboten. Oracle Text kann u.a. für das Text Mining [12] sowie für die Klassifikation, Clusterbildung und Visualisierung von textuellen Daten [13] eingesetzt werden. Fortgeschrittene Funktionalitäten wie Transaktionssicherheit, Notfallwiederherstellung, Replikation, Kompression und Verschlüsselung werden vollständig unterstützt.

***Oracle Spatial and Graph:*** Mit Spatial and Graph [14], [15] bietet Oracle zum einen eine in die Oracle Database Plattform integrierte Lösung zur Speicherung, Verarbeitung und Analyse von 3D-Daten und geografischen bzw. standortbezogenen Daten (mit Funktionen für Routing, Geokodierung, usw.) sowie zum anderen die native Unterstützung von NDM-Graphen (Network Data Model) und W3C-standardisierten RDF-Semantic-Graphen (Resource Description Framework). Die Integration in SQL-Anweisungen ist ebenso möglich wie die Nutzung der standardisierten, graphenbasierten Abfragesprache SPARQL oder einer Java API.

***Oracle Multimedia:*** Oracle Multimedia [8] ermöglicht die direkte Speicherung und Verarbeitung von Audio-, Video- und Bild- sowie anderen Multimedia-Daten innerhalb der Oracle Database Plattform. Der Zugriff erfolgt über SQL-Befehle oder eine Java-API. Zur Indexierung werden u.a. die Metadaten extrahiert. Zudem wird der DICOM-Standard (Digital Imaging and Communication in Medicine) zur Verwaltung von medizinischen Inhalten unterstützt.

***Oracle In-Memory Column Store:*** Bei Oracle In-Memory Column Store [16] handelt es sich um eine in die Oracle Database Plattform integrierte Speicherkonzept-Erweiterung, die eine spaltenorientierte Speicherung im Hauptspeicher ermöglicht und so die Koexistenz der zeilen- und spaltenbasierten Speicherkonzepte innerhalb einer Datenbank zulässt. Eine Anpassung des zugrundeliegenden Datenmodells ist hierbei nicht erforderlich. Die Daten werden zunächst immer zeilenbasiert abgelegt, bei Bedarf werden Tabellen, einzelne Spalten oder Indizes zusätzlich spaltenorientiert im Hauptspeicher gehalten werden. Eine reine spaltenorientierte Speicherung ist also nicht möglich. Der Datenzugriff mittels SQL erfolgt identisch zur zeilenorientierten Speicherung.

### 2.1.2 Oracle NoSQL Database

***Architektur:*** Bei der Oracle NoSQL Database [17], [18] handelt es sich um eine verteilte und hochverfügbare Schlüssel-Wert-Datenbank, die auf die bewährte Speichertechnologie der Oracle Berkeley DB Java Edition aufbaut. Die Skalierung arbeitet nach dem 1-Master/N-Replica-Prinzip. Inserts und Updates werden zunächst auf dem Master-Knoten und anschließend auf einer beliebigen, vorher festgelegten Anzahl an Replica-Knoten durchgeführt. Je nach Konfiguration erfolgt

dies synchron oder asynchron. Lesende Zugriffe erfolgen je nach Konfiguration nur auf dem Master-Knoten oder auch auf den Replica-Knoten. Im Falle eines Masterausfalls wird gemäß dem PAXOS-Protokoll automatisch ein neuer Master ermittelt. Vergleiche der Oracle NoSQL Database mit anderen NoSQL-Datenbanksystemen können unter [19], [20], [21], [22], [23], [24] und [25] nachgelesen werden.

***Konsistenzmodell:*** Gemäß dem CAP-Theorem ist die Oracle NoSQL Database entweder hinsichtlich C/P (Schreibender Zugriff: Synchrone Replikation/Skalierung; Lesender Zugriff: Nur über Master-Knoten) oder A/P (Schreibender Zugriff: Asynchrone Replikation/Skalierung; Lesender Zugriff: Es kann sowohl vom Master-Knoten als auch von den Replica-Knoten gelesen werden) konfigurierbar. Demnach wird das klassische ACID-Konsistenzmodell nicht unterstützt. Laut [26] unterstützt Oracle NoSQL Database im Gegensatz zu vielen anderen NoSQL-Systemen auch kein Eventual Consistency und somit auch nicht vollständig das BASE-Konsistenzmodell.

***Datenmodell:*** Wie bereits eingangs beschrieben, werden die Daten in einen Schlüssel-Wert-Speicher abgelegt. Der Schlüssel eines Schlüssel-Wert-Paares besteht jedoch aus zwei Teilen, einem "Major-Key" und einem "Minor-Key". Hierbei wird garantiert, dass Schlüssel-Wert-Paare mit dem gleichen Major-Key physikalisch auf dem gleichen Knoten abgelegt werden. Der Wert eines Schlüssel-Wert-Paares kann in drei Varianten abgelegt werden: im JSON-Format, binär oder in Tabellenform.

***Datenzugriff, API und Integration:*** Oracle NoSQL Database stellt eine einfache Java- und C-API für das Abrufen, Hinzufügen, Aktualisieren und Löschen von Schlüssel-Wert-Paaren bereit. Um die Filterung nach einem Wert zu beschleunigen, können sogenannte Secondary Indizes über beliebige Felder innerhalb des Werts eines Schlüssel-Wert-Paares erstellt werden. Eine native Unterstützung von Anweisungen in SQL sowie Joins werden nicht angeboten. Jedoch ist es in der Enterprise Edition einer Oracle Database möglich, durch die Integration von externen Tabellen Daten mit SQL-Befehlen aus einer Oracle NoSQL Database auszulesen. Darüber hinaus existiert eine Apache Hadoop Integration. Mit dieser ist es möglich, von Map/Reduce-Jobs in Apache Hadoop lesend auf Daten in einer Oracle NoSQL Database zuzugreifen.

### 2.1.3 Oracle Big Data Management System

Im April 2015 hat Oracle die Veröffentlichung einer neuen Enterprise-Big-Data-Lösung namens Oracle Big Data Management System [27] angekündigt. Diese Lösung ist "[...] Teil der Vision von Oracle, wonach Hadoop-, NoSQL- und SQL-Technologien zusammenarbeiten und sicher in jedem Modell eingesetzt werden können [...]" [28]. Das Oracle Big Data Management System setzt sich hierbei hauptsächlich aus den drei nachfolgenden Bestandteilen zusammen:

***Oracle Database:*** Ein Data Warehouse auf Basis der Oracle Database und der Oracle Exadata Database Machine, zur primären Speicherung, Verarbeitung und Analyse der zentralen Unternehmensdaten.

***Oracle Big Data Appliance:*** Ein zentraler Datenspeicher zur Speicherung, Verarbeitung und Analyse von strukturieren, semistrukturierten und unstrukturierten Daten. Die Basis bildet die Oracle NoSQL Database sowie die Open-Source Hadoop-Distribution von Cloudera.

***Oracle Big Data SQL:*** Die Verbindung zwischen Oracle Database, Oracle NoSQL Database und Cloudera Hadoop. Um auf die eben genannten Datenbank zugreifen zu können, müssen bisher

individuelle und nicht kompatible Technologien, wie SQL (bei Oracle Database), Java-/C-API (bei Oracle NoSQL Database) oder Map/Reduce (bei Cloudera Hadoop), verwendet werden.


Abbildung 2: Verschiedene Datenbanken und deren Möglichekiten zum Datenzugriff [29]

Mit Oracle Big Data SQL setzt Oracle an dieser Stelle auf einen "Unified Query"-Ansatz (auch "Query Franchising"-Ansatz genannt) und ermöglicht dadurch einen, in standardisierten SQL-Anweisungen geschriebenen, einheitlichen und transparenten Zugriff auf Daten in Oracle Database, Oracle NoSQL Database und Cloudera Hadoop. Die erweiterten Sicherheitsrichtlinien der Oracle Database werden dabei übergreifend auf Daten in NoSQL und Hadoop angewandt. Eine detailliertere Beschreibung der Funktionsweise würde jedoch den Rahmen dieser Studienarbeit bei weitem sprengen.


Abbildung 3: Der "Unified Query"-Ansatz von Oracle Big Data SQL [29]

## 2.2 Microsoft

### 2.2.1 Microsoft SQL Server

***Native JSON-Unterstützung:*** Mit der Implementierung einer nativen JSON-Unterstützung in den zukünftigen SQL Server 2016 [31], [32] reagiert das Entwicklerteam von Microsoft auf die starke Nachfragen aus der Community: Der Wunsch nach einer native JSON-Unterstützung besteht bereits seit Sommer 2011 und ist das am meisten nachgefragte Feature der Community [33]. Die native JSON-Unterstützung wird sich voraussichtlich auf zwei Funktionalitäten beschränken: Einen Parser, um eine JSON-kodierte Zeichenkette und die darin enthaltenen Daten in relationale Tabellen einzulesen sowie die Möglichkeit, das Ergebnis einer SQL-Abfrage als JSON-kodierte Zeichenkette zurückzuliefern. Die Implementierung eines eigenen JSON Datentyps, wie beispielsweise bei XML, ist nicht geplant. Aus Gründen der Rückwärtskompatibilität legt die Datenbank die JSON-Daten als NVARCHAR ab.

***XML Datentyp:*** Für die Speicherung, Verarbeitung und Analyse von XML-Dokumenten bietet Microsoft SQL Server einen eigenen Datentyp [30], [34] an, welcher in relationalen Tabellen als Datentyp einer Spalte verwendet werden kann. Für die Speicherung stehen zwei Speicheroptionen (systemeigene XML-Speicherung und XML-Sicht-Speicherung) zur Verfügung, die die Genauigkeit der Dokumente in unterschiedlichem Ausmaß beibehalten und mit unterschiedlichen Möglichkeiten für Abfragen, Änderungen und Indizierungen sowie Leistungsmerkmalen verbunden sind. Innerhalb von SQL-Anweisungen können mit Hilfe von XQuery und XPath Daten aus XML abgefragt und bearbeitet werden. Laut einem Vergleich [35] der dokumentenorienterten NoSQL-Datenbank MongoDB mit dem XML Datentyp des SQL Servers, schneidet der XML-Datentyp gegenüber der NoSQL-Datenbank mit teilweise gravierenden Einbußen bei der Geschwindigkeit ab.

***Sparse Columns und Column Sets:*** Bei Sparse Columns [36] handelt es sich um reguläre Spalten einer Tabelle, die jedoch auf eine sehr effiziente Verwaltung und Speicherung von so genannten NULL-Werten optimiert sind. Sparse Columns eigenen sich vor allem bei Spalten, die hauptsächlich NULL-Werte beinhalten. Dies ist zum Beispiel bei der Speicherung von Titeln (Dr., Prof., usw.) eines Kunden oder bei optionalen Angaben (zweite Adresszeile oder ähnliches) häufig der Fall. Innerhalb einer SQL-Anweisung werden die klassischen CRUD-Befehle unterstützt. Die Erstellung von gefilterten Indizes ist ebenfalls möglich. Die maximale Anzahl von Sparse Columns innerhalb einer Tabelle ist nicht auf die üblichen 1024 Spalten pro Tabelle sondern auf 100.000 begrenzt [37].

Mit Hilfe eines Column Sets [38] ist es möglich, alle Sparse Columns einer Tabelle zusammen zu fassen und als strukturierte XML-Repräsentation zurück zu geben. Dies erleichtert zum einen den Zugriff auf die einzelnen Sparse Columns und erhöht zum anderen die Geschwindigkeit. Dabei wird ein Column Set nicht physikalisch gespeichert sondern zur Laufzeit errechnet. Column Sets können in SQL-Anweisungen verwendet werden und erlauben alle CRUD-Befehle.

***FileStream und FileTable:*** Bei FileStream [30], [39] handelt es sich um einen Datentyp, der es ermöglicht, unstrukturierte Daten in einer SQL Server Datenbank abzulegen. Die Daten werden dabei intern als BLOBs (Binary Large Objects) gespeichert und im Dateisystem abgelegt. Die klassischen SQL-CRUD-Befehle sind ebenso mögliche wie eine Volltextsuche. Obwohl die Daten außerhalb der Datenbank in Dateisystem gespeichert werden, wird Transaktionssicherheit, die Anwendung des interne Sicherheitskonzept des SQL Servers sowie Backup und Notfallwiederherstellung vollständig gewährleistet.

Bei FileTable [30], [39] handelt es sich um eine spezielle Tabelle mit fester Struktur, die Daten des FileStreams sowie zugehörige Dateiattribute (z.B. Name und Pfad der Datei im Dateisystem oder Erstell- und Änderungsdatum, usw.) enthält. Darüber hinaus wird innerhalb der Tabelle eine Verzeichnisstruktur abgebildete; dies bedeutet, jede Zeile entspricht einem Verzeichnis oder einer Datei im Dateisystem. Für externe Anwendungen sowie SQL-Befehle innerhalb der Datenbank funktioniert FileTable ähnlich wie der Zugriff auf einen Dateiserver, bietet darüber hinaus jedoch volle Transaktionssicherheit, die Anwendung des interne Sicherheitskonzept des SQL Servers sowie Backup und Notfallwiederherstellung.

***ColumnStore Index:*** Beim ColumnStore Index [39], [40] handelt es sich um eine Erweiterung des Speicherkonzept, die eine spaltenorientierte Speicherung ermöglicht und so die Koexistenz der zeilen- und spaltenbasierten Speicherkonzepte innerhalb einer Datenbank zulässt. Das zugrundeliegende Datenmodell muss hierbei nicht angepasst werden, da die Daten sowohl zeilen- als auch spaltenbasierte abgelegt werden. Eine reine spaltenorientierte Speicherung ist nicht möglich.

### 2.2.2 Microsoft Big Data Solutions

Microsoft bietet mit den Big Data Solutions zwei Enterprise-Big-Data-Lösungen für die Verarbeitung und Analyse von strukturierten und unstrukturierten Daten an [40]:

***HDInsight:*** Bei HDInsight handelt es sich um eine auf Apache Hadoop basierten Cloud-Plattform zur Analyse von Daten aller Art. Die große Stärke von HDInsight ist der sehr schnelle Aufbau bzw. die sehr schnelle Bereitstellung einer Infrastruktur für die Analyse und Verarbeitung von Massendaten. Dies eignet sich u.a. für die Erstellung eines Proof-of-Concept oder für die Ermittlung von Anforderungen für eine später zu erstellende produktive Umgebung.

***PolyBase:*** Bei PolyBase [40], [41] handelt es sich um eine Schnittstelle, die es ermöglicht, mit SQL-Befehlen gleichzeitig auf Daten zuzugreifen die im Microsoft SQL Server Parallel Data Warehouse sowie in Apache Hadoop gespeichert sind. Auch Joins zwischen relationalen Daten im SQL Server und nicht-relationalen Daten in Hadoop sowie das verschieben von Daten zwischen den beiden Systemen ist möglich.

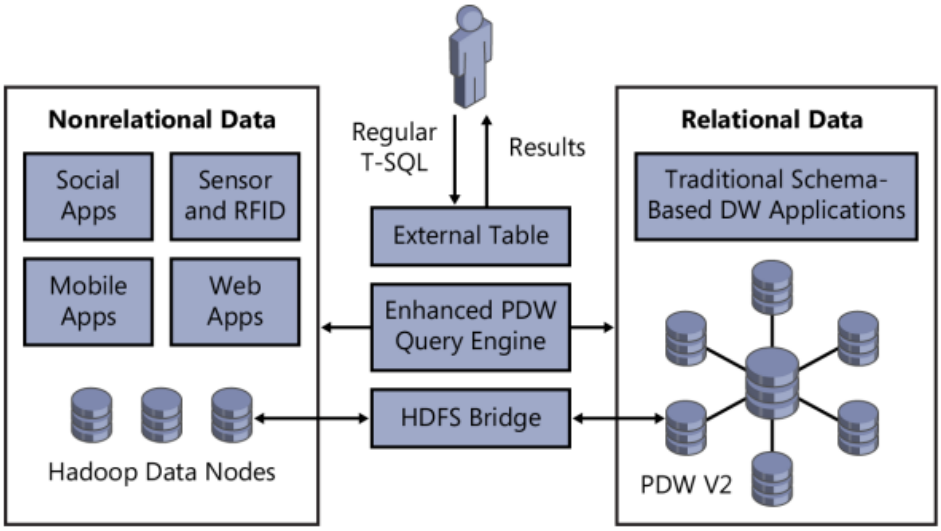


Abbildung 4: PolyBase als Schnittstelle zwischen Hadoop und SQL Server

PolyBase ist sehr flexibel und daher weder abhängig von einem bestimmten Betriebssystems noch von einer Hadoop-Distribution eines bestimmten Herstellers. Darüber hinaus werden alle Arten von HDFS-Dateiformaten unterstützt.

# 3 Vergleich der Datenbanksysteme

## 3.1 Erstellung des Kriterienkataloges

Ziel dieses Kapitels und Teilziel dieser Studienarbeit ist es, eine erste Version eines Kriterienkataloges zu erstellen, der Datenbankanwender und Entscheider dabei unterstützt, das für einen gegebenen Anwendungsfall passende Datenbanksystem auszuwählen. Die Grundlage hierfür bilden die gewonnenen Erkenntnisse aus der Analyse der unterstützten NoSQL-Konzepte der einzelnen Datenbanksysteme.

Da es sich bei den meisten entscheidungsrelevanten Kriterien nicht um quantitative sondern um qualitative Systemeigenschaften handelt, wurde für die Erstellung des Kriterienkataloges der Einsatz von Nutzwertanalysen [42] als Bewertungsverfahren gewählt. Die Anzahl der notwendigen Prozessschritte beim Aufstellen von Nutzwertanalysen kann im allgemeinen auf die nachfolgenden vier Schritte reduziert werden.

### 3.1.1 Identifikation und Klassifikation von Auswahlkriterien

Der erste Schritt umfasst die Identifikation sowie Klassifikation der für die Entscheidungsfindung relevanten Auswahlkriterien. Um auch bei einer hohen Anzahl von Kriterien eine leichte und sinnvolle Anwendung der Nutzwertanalyse zu gewährleisten, können gleichartige Kriterien zu Gruppen zusammengefasst sowie anschließend ggf. erneut unterteilt werden. Dadurch entsteht eine mehrere Ebenen umfassende Baumstruktur.

Wie bereits in der Einführung erwähnt, soll in dieser Studienarbeit vor allem auf die Unterstützung hinsichtlich der Speicherung, Verarbeitung und Analyse von nicht-relationalen Daten eingegangen werden. Daher bildet die oberste Ebene des Kriterienkataloges die Struktur bzw. Orientierung der zu verarbeitenden Daten:

- ***Dokumenten-Orientiert:*** Daten dieser Kategorie zeichnen sich durch ihre schemafreie Organisation aus und weisen u.U. datensatzübergreifend keine einheitliche Struktur auf. Hierzu zählen nicht nur die klassischen Formate wie JSON-Objekte und XML-Dokumente sondern jede Art von schemafreien Dateien, beispielsweise Office-Dokumente (Word, Excel, usw.) oder HTML und YAML. Aufgrund der hohen Verbreitung bei JSON und XML werden diese in einer separaten Kategorie betrachtet.
- ***Graphen-Orientiert:*** Hierzu zählen Daten zur Darstellung von Beziehungen und anderen stark vernetzten Informationen und bestehen meist aus Knoten und Kanten sowie ggf. Eigenschaften. Darüber hinaus sind dies Art von Daten oftmals generisch.
- ***Spalten-Orientiert:*** Hierunter fallen Daten, die sich nicht in einem Datensatz mit einer vorher festgelegten festen Anzahl an Spalten darstellen lässt sondern aus sehr vielen dynamischen Spalten bestehen kann.
- ***Schlüssel-Wert-Paar-Orientiert:*** Daten dieser Kategorie sind strukturell in einer sehr einfachen Form gehalten und bestehen lediglich aus Paaren von Schlüsseln und zugehörigen Werten.
- ***XML-Orientiert:*** XML-Dokumente fallen genau genommen in die Dokumenten-orientierte Kategorie, werden jedoch aufgrund der hohen Verbreitung und der nativen Implementierung in vielen datenverarbeitenden Systemen als gesondertes Auswahlkriterium aufgelistet.

- ***JSON-Orientiert:*** JSON-Objekte fallen genau genommen in die Dokumenten-orientierte Kategorie, werden jedoch aufgrund der hohen Verbreitung und der nativen Implementierung in vielen datenverarbeitenden Systemen als gesondertes Auswahlkriterium aufgelistet.
- ***Binär/Multimedia-Orientiert:*** Hierzu zählen Daten im Binärformat sowie Multimediadaten, wie beispielsweise Audio, Video und Bilder.
- ***Geodaten-Orientiert:*** Hierbei handelt es sich um geografische und standortbezogene Daten.

Jedes Kriterium der obersten Ebene des Kriterienkataloges enthält die nachfolgenden Unterkriterien:

- ***Mächtigkeit des Datenmodells:*** Dieses Kriterium dient zur Bewertung der Mächtigkeit, Ausdrucksstärke und Flexibilität des Datenmodells. Hier fließen auch die bereitgestellten Modellierungsmöglichkeiten mit ein.
- ***Mächtigkeit der Anfrageschnittstelle:*** Beschreibt den Umfang der bereitgestellten Anfragefunktionalitäten. Neben der Art und Anzahl der zur Verfügung stehenden Operationen werden an dieser Stelle auch die Bereitstellung von Verbundoperationen untersucht. Eine detailliertere Gliederung in weitere Unterkategorien, wie beispielsweise einzelne Operationen, sind hier möglich, würden den Umfang dieser Studienarbeit jedoch bei weitem sprengen.
- ***Integration in SQL:*** Da die Umsetzung von NoSQL-Konzepten in bestehenden relationalen Datenbanksystemen untersucht werden soll, ist es notwendig, die Art und den Umfang einer SQL-Integration zu bewerten. Auch hier ist die Untergliederung in weitere Kategorien möglich, würde jedoch den Umfang der Studienarbeit sprengen.
- ***Transaktionsunterstützung:*** Mit diesem Kriterium soll die Unterstützung von Transaktionen analysiert und bewertet werden.
- ***Konsistenzmodell:*** Viele NoSQL-Datenbanksysteme verzichten aus diversen Gründen auf die vollständige Unterstützung der ACID-Eigenschaften. Dieses Kriterium dient zur Bewertung des unterstützten Konsistenzmodells. Um die Bewertung zu erleichtern, kann beispielsweise nur zwischen den beiden Konsistenzmodellen ACID und BASE unterschieden werden. Alternativ können die Konsistenzmodelle auch auf die einzelnen Elemente herunter gebrochen werden oder eine Bewertung anhand des CAP-Theorems vorgenommen werden.
- ***Leistungsfähigkeit:*** Da viele Datenbanksysteme auf unterschiedliche Operationen hin optimiert werden, wird mit diesem Kriterium die Leistungsfähigkeit der Lese- und Schreiboperationen untersucht und bewertet.

Darüber hinaus konnten noch weitere Auswahlkriterien identifiziert werden. Diese sind zum Teil nicht direkt dem NoSQL-Konzepten zuordenbar und zudem meist datenmodellunabhängigen bzw. übergreifend anwendbar.

- ***Skalierbarkeit:*** Dient zur Bewertung der Mächtigkeit der Skalierbarkeit und des Skalierbarkeitsverhaltens.
- ***Entwicklungsreife:*** Da Datenbanksysteme meist in geschäftsrelevanten Bereichen zum Einsatz kommen, spielt eine hohe Entwicklungsreife und somit eine geringe Fehleranfälligkeit eine nicht zu unterschätzende Rolle.
- ***Werkzeugunterstützung:*** Umfang und Qualität der zur Verfügung gestellten Werkzeuge weichen zum Teil stark voneinander ab und sollte daher berücksichtigt werden. Dies ist allerdings u.U. quantitativ schwer messbar.

- ***Fachwissen:*** Das zur Verfügung stehende Fachwissen, beispielsweise in Form von qualitativ hochwertigen und umfangreichen Dokumentationen oder Fachliteratur, kann Bestandteil der Entscheidungsgrundlage sein. Dies ist allerdings u.U. quantitativ schwer messbar.
- ***Standardisierung:*** Um den Austausch oder die Erweiterung von Systemen zu vereinfachen, kann die Unterstützung von Standards als Auswahlkriterium eine Rolle spielen.

### 3.1.2 Bewertungsskala

Der zweite Schritt der Nutzwertanalyse umfasst die einheitliche Bewertung der einzelnen Alternativen hinsichtlich der Auswahlkriterien, um diese anschließend vergleichen zu können. Hierfür wird im Allgemeinen eine Bewertungsskala von null bis zehn Punkten verwendet, wobei zehn Punkte die beste und null Punkte die schlechteste Bewertung darstellt. Da bei der Bewertung der qualitativen Eigenschaften oftmals ein großer Ermessensspielraum zum Tragen kommt, muss darauf geachtet werden, dass die Beurteilung der einzelnen Alternativen nicht verzerrt wird.

| Qualitative Bewertung | Quantitative Bewertung |
|---|---|
| Sehr Gut | 10 |
| | 9 |
| Gut | 8 |
| | 7 |
| Befriedigend | 6 |
| | 5 |
| | 4 |
| Ausreichend | 3 |
| | 2 |
| Mangelhaft | 1 |
| | 0 |

Tabelle 1: Bewertungsskala

### 3.1.3 Gewichtungsfaktoren

Um bei der Entscheidungsfindung unterschiedliche Prioritäten bezüglich der Auswahlkriterien zu berücksichtigen, können die einzelnen Kriterien mit einem Gewichtungsfaktor versehen werden. Dieser wird anschließend mit dem Punktewert der quantitativen Bewertung multipliziert und ist für jedes Kriterium, jede Gruppe und jeder Ebene einzeln anzuwenden. Die Gewichtung ist subjektiv und hängt dabei von den individuellen Präferenzen des Entscheidungsträgers ab.

Die Gewichtungsfaktoren können prozentual (in diesem Fall muss die Summe aller Einzelfaktoren 100 Prozent ergeben) oder nach einer freien Skalierung, wie beispielsweise in nachfolgender Tabelle abgebildet, dargestellt werden.

| Qualitative Bewertung | Quantitative Bewertung |
|---|---|
| Sehr Gut / Sehr Wichtig | 9 |
| Gut / Eher Wichtig | 8 |
| | 7 |
| | 6 |
| Mittel / Weder-Noch | 5 |
| | 4 |
| Schlecht / Eher Unwichtig | 3 |
| | 2 |
| | 1 |
| Sehr Schlecht / Unwichtig | 0 |

Tabelle 2: Gewichtungsfaktor

### 3.1.4 Nutzwertberechnung

Nachdem die Auswahlkriterien identifiziert und klassifiziert sowie die Eigenschaften bewertet und die Kriterien gewichtet wurden kann im vierten und letzten Schritt der Nutzwert der einzelnen Alternativen errechnet werden. Der Nutzwert einer Alternative ergibt sich hierbei aus der Verrechnung aller Teilnutzwerte, die zuvor für jedes Kriterium, jede Gruppe und jede Ebene ermittelt wurden. Die Alternative mit dem höchsten Nutzwert ist für den jeweils gegebenen Anwendungsfall die beste Wahl.

### 3.1.5 Kriterienkatalog

Nachfolgend ist der Aufbau des entwickelten Kriterienkataloges vollständig abgebildet. Die Bewertungsspalten der einzelnen Alternativen sind aufgrund der geringen zur Verfügung stehenden Zeit einer Studienarbeit noch leer. Im nachfolgenden Abschnitt werden jedoch einige exemplarisch gewählte Kriterien anhand der gewonnen Erkenntnisse aus Kapitel 2 bewertet.

| Kriterium | Gewichtung | Oracle | | | Microsoft | |
|---|---|---|---|---|---|---|
| | | Database | NoSQL Database | Big Data Mgmt Sys | SQL Server | Big Data Solutions |
| **Dokumenten-Orientiert** | **7,7%** | | | | | |
| Datenmodell | 16,6% | | | | | |
| Anfrageschnittstelle | 16,6% | | | | | |
| SQL-Integration | 16,6% | | | | | |
| Transaktionsunterstützung | 16,6% | | | | | |
| Konsistenzmodell | 16,6% | | | | | |
| ACID<br>BASE | 0%<br>100% | | | | | |
| Leistungsfähigkeit | 16,6% | | | | | |
| Lesend<br>Schreibend | 50%<br>50% | | | | | |
| **Graphen-Orientiert** | **7,7%** | | | | | |
| Datenmodell | 16,6% | | | | | |
| Anfrageschnittstelle | 16,6% | | | | | |
| SQL-Integration | 16,6% | | | | | |
| Transaktionsunterstützung | 16,6% | | | | | |
| Konsistenzmodell | 16,6% | | | | | |
| ACID<br>BASE | 0%<br>100% | | | | | |
| Leistungsfähigkeit | 16,6% | | | | | |
| Lesend<br>Schreibend | 50%<br>50% | | | | | |
| **Spalten-Orientiert** | **7,7%** | | | | | |
| Datenmodell | 16,6% | | | | | |
| Anfrageschnittstelle | 16,6% | | | | | |
| SQL-Integration | 16,6% | | | | | |
| Transaktionsunterstützung | 16,6% | | | | | |
| Konsistenzmodell | 16,6% | | | | | |
| ACID<br>BASE | 0%<br>100% | | | | | |
| Leistungsfähigkeit | 16,6% | | | | | |
| Lesend<br>Schreibend | 50%<br>50% | | | | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| **XML-Orientiert** | **7,7%** | | | | | |
| Datenmodell | 16,6% | | | | | |
| Anfrageschnittstelle | 16,6% | | | | | |
| SQL-Integration | 16,6% | | | | | |
| Transaktionsunterstützung | 16,6% | | | | | |
| Konsistenzmodell | 16,6% | | | | | |
| ACID<br>BASE | 0%<br>100% | | | | | |
| Leistungsfähigkeit | 16,6% | | | | | |
| Lesend<br>Schreibend | 50%<br>50% | | | | | |
| **JSON-Orientiert** | **7,7%** | | | | | |
| Datenmodell | 16,6% | | | | | |
| Anfrageschnittstelle | 16,6% | | | | | |
| SQL-Integration | 16,6% | | | | | |
| Transaktionsunterstützung | 16,6% | | | | | |
| Konsistenzmodell | 16,6% | | | | | |
| ACID<br>BASE | 0%<br>100% | | | | | |
| Leistungsfähigkeit | 16,6% | | | | | |
| Lesend<br>Schreibend | 50%<br>50% | | | | | |
| **Schl.-W.-Paar-Orientiert** | **7,7%** | | | | | |
| Datenmodell | 16,6% | | | | | |
| Anfrageschnittstelle | 16,6% | | | | | |
| SQL-Integration | 16,6% | | | | | |
| Transaktionsunterstützung | 16,6% | | | | | |
| Konsistenzmodell | 16,6% | | | | | |
| ACID<br>BASE | 0%<br>100% | | | | | |
| Leistungsfähigkeit | 16,6% | | | | | |
| Lesend<br>Schreibend | 50%<br>50% | | | | | |
| **Binär/Multime.-Orientiert** | **7,7%** | | | | | |
| Datenmodell | 16,6% | | | | | |
| Anfrageschnittstelle | 16,6% | | | | | |
| SQL-Integration | 16,6% | | | | | |
| Transaktionsunterstützung | 16,6% | | | | | |
| Konsistenzmodell | 16,6% | | | | | |
| ACID<br>BASE | 0%<br>100% | | | | | |
| Leistungsfähigkeit | 16,6% | | | | | |
| Lesend<br>Schreibend | 50%<br>50% | | | | | |
| **Skalierbarkeit** | **7,7%** | | | | | |
| **Entwicklungsreife** | **7,7%** | | | | | |
| **Werkzeugunterstützung** | **7,7%** | | | | | |
| **Fachwissen** | **7,7%** | | | | | |
| **Standardisierung** | **7,7%** | | | | | |

Tabelle 3: Erste Version des Kriterienkataloges

## 3.2 Anwendung des Kriterienkataloges

Nachdem die einzelnen Alternativen hinsichtlich ihrer Unterstützung von NoSQL-Konzepten - vor allem im Bezug auf das Verarbeiten und Speichern von unstrukturierten und semistrukturierten Daten - analysiert und anschließend, darauf aufbauend, die erste Version eines Kriterienkataloges entwickelt wurde, kann dieser nun bei der Entscheidungsfindung Hilfestellung leisten. Aufgrund der geringen zur Verfügung stehenden Zeit kann in dieser Studienarbeit nur ein Teil der umfangreichen Kriterien bewertet werden und somit nur ein Teil des Kriterienkataloges Anwendung finden. Die Gewichtungen wurden nachfolgend exemplarisch auf alle Kriterien gleich verteilt.

| Kriterium | Gewichtung | Oracle | | | Microsoft | |
|---|---|---|---|---|---|---|
| | | Database | NoSQL Database | Big Data Mgmt Sys | SQL Server | Big Data Solutions |
| **Dokumenten-Orientiert** | **16,6%** | | | | | |
| Datenmodell | 33,3% | 9 | 8 | 8 | 7 | 8 |
| Anfrageschnittstelle | 33,3% | 9 | 7 | 8 | 5 | 8 |
| SQL-Integration | 33,3% | 9 | 5 | 8 | 5 | 8 |
| **Graphen-Orientiert** | **16,6%** | | | | | |
| Datenmodell | 33,3% | 7 | 1 | 2 | 1 | 2 |
| Anfrageschnittstelle | 33,3% | 8 | 1 | 2 | 1 | 2 |
| SQL-Integration | 33,3% | 7 | 1 | 2 | 1 | 2 |
| **Spalten-Orientiert** | **16,6%** | | | | | |
| Datenmodell | 33,3% | 0 | 9 | 10 | 5 | 10 |
| Anfrageschnittstelle | 33,3% | 0 | 9 | 9 | 4 | 9 |
| SQL-Integration | 33,3% | 0 | 5 | 9 | 8 | 9 |
| **Schl.-W.-Paar-Orientiert** | **16,6%** | | | | | |
| Datenmodell | 33,3% | 5 | 10 | 10 | 5 | 10 |
| Anfrageschnittstelle | 33,3% | 7 | 9 | 9 | 7 | 9 |
| SQL-Integration | 33,3% | 10 | 5 | 9 | 10 | 9 |
| **XML-Orientiert** | **16,6%** | | | | | |
| Datenmodell | 33,3% | 10 | 8 | 9 | 9 | 9 |
| Anfrageschnittstelle | 33,3% | 9 | 7 | 9 | 8 | 9 |
| SQL-Integration | 33,3% | 8 | 5 | 9 | 8 | 8 |
| **JSON-Orientiert** | **16,6%** | | | | | |
| Datenmodell | 33,3% | 10 | 10 | 9 | 7 | 9 |
| Anfrageschnittstelle | 33,3% | 9 | 9 | 9 | 5 | 5 |
| SQL-Integration | 33,3% | 9 | 5 | 9 | 3 | 3 |

Tabelle 4: Anwendung des Kriterienkataloges

Anwender können jederzeit eine individuelle Gewichtung gemäß den spezifischen Anforderungen des zu prüfenden Anwendungsfalls vornehmen. Die Summe der Einzelgewichtungen muss dabei allerdings immer 100% ergeben. Ist die Anforderung beispielsweise die Verarbeitung von XML- und JSON-Daten, müssen die 100% auf die Kriterien XML und JSON verteilt werden. Die übrigen Kriterien werden mit 0% bewertet und somit nicht berücksichtigt. Die Unterkategorien können ebenfalls individuell gewichtet werden.

# 4 Zusammenfassung und Ausblick

Die Analyse der Datenbanksysteme hat aufgezeigt, das sowohl Oracle als auch Microsoft die Notwendigkeit von NoSQL-Konzepten erkannt haben und stetig neue Konzepte in Ihre Datenbanksysteme integriert haben und integrieren. Vor allem hinsichtlich der Unterstützung verschiedenster nicht-relationaler Datenmodelle sowie der SQL-Integration sind die Systeme von Oracle den Systemen von Microsoft jedoch meist voraus.

Durch die native Integration von XML und JSON sowie die Erweiterungen Oracle Text und Oracle Spatial and Graph wird die ursprünglich rein relationale Datenbank Oracle Database um viele nicht-relationale Datenmodelle erweitert. Lediglich für die spaltenorientierte Speicherung muss auf die neu entwickelte Datenbank Oracle NoSQL Database ausgewichen werden. Diese lässt sich jedoch durch den Einsatz von externe Tabellen problemlos in das bestehende Oracle-Ökosystem integrieren.

Die relationale Datenbank SQL Server von Microsoft wurde in den letzten Versionen um viele Funktionen zur Speicherung und Verarbeitung von nicht-relationalen Daten erweitert, hinkt jedoch in den meisten Bereichen der Oracle Database hinterher. Lediglich mit der Erweiterung Sparse Columns und Column Sets bietet die relationale Datenbank von Microsoft ein an die spaltenorientierte Speicherung angelehntes Konzept, welches allerdings nicht so mächtig ist wie die neue Oracle NoSQL Database oder NoSQL-Datenbanksysteme wie HBase oder Cassandra.

Die neuen Enterprise-Big-Data-Lösungen Oracle Big Data Management System und Microsoft Big Data Solutions, die eine einheitliche und transparente Zusammenarbeit von Hadoop-, NoSQL- und SQL-Technologien ermöglichen sollen, konnten - aufgrund des Umfangs sowie der bisher geringen Erfahrungswerte - nur am Rande untersucht werden. Die dahinterliegenden Visionen, Konzepte und Ansätze sind jedoch vielversprechend.

Eine mögliche Folgearbeit könnte die einzelnen Datenbanksysteme noch tiefergehender untersuchen und detaillierter vergleichen sowie verschiedene Szenarien implementieren und anschließend Vergleichsmessungen der Ausführungsgeschwindigkeiten durchführen. Dadurch könnte relevante und vergleichbare Erkenntnisse über den Administrations- und Implementierungsaufwand sowie über die Leistungsfähigkeit der einzelnen Datenbanksysteme gewonnen werden.

Die erstellte erste Version eines Kriterienkatalog auf Basis einer Nutzwertanalyse bietet eine transparente und nachvollziehbare Bewertungsgrundlage zum Vergleich der gegebenen Datenbanksysteme hinsichtlich der Unterstützung von NoSQL-Konzepten. Dabei können qualitative und quantitative Auswahlkriterien berücksichtigt werden. Nicht vergessen werden darf jedoch, dass sowohl die Gewichtung der Auswahlkriterien als auch die Bewertung der Alternativen und somit die errechneten Nutzwerte von vielen subjektiven Parametern abhängen. Deshalb sollte der Kriterienkatalog nur als Hilfestellung und nicht als alleinige Entscheidungsgrundlage interpretiert werden.

# 5 Literaturverzeichnis

[1] Oracle: *Oracle NoSQL Database*; An Oracle White Paper; September 2011

[2] Dean, Jeffrey; Ghemawat, Sanjay: *MapReduce: Simplified Data Processing on Large Clusters*; Google Labs; in: OSDI'04: Sixth Symposium on Operating System Design and Implementation; San Francisco; CA; December 2004

[3] Wartala, Ramon: *Hadoop: Zuverlässige, verteilte und skalierbare Big-Data-Anwendungen*; Open Source Press; München; 1. Auflage; 2012

[4] Stonebraker, M.; Cetintemel, U.: *One Size Fits All: An Idea Whose Time has Come and Gone*; in: Proceedings of the 21st International Conference on Data Engineering (ICDE); 2005; S. 2 - 11

[5] Stonebraker, M.; et al: *The End of an Architectural Era (It's Time for a Complete Rewrite)*; Proceedings of VLDB '07; 2007; Wien; S. 1150 - 1160

[6] Hecht, R.; Jablonski, S.: *NoSQL Evaluation: A Use Case Oriented Survey*; in Proceedings of the 2011 International Conference on Cloud and Service Computing (CSC '11); 2011; S. 336 - 341

[7] Do You Know NoSQL?
http://www.forbes.com/sites/oracle/2013/08/22/do-you-know-nosql/
Veröffentlicht: 22. August 2013; Abgerufen: 9. Juni 2015

[8] Oracle: *Unstructured Data Management with Oracle Database 12c*; An Oracle White Paper; September 2014

[9] Oracle: *Schemaless Application Development with Oracle Database 12c*; An Oracle White Paper; April 2015

[10] Oracle: *Oracle XML DB in Oracle Database 12c*; An Oracle White Paper; June 2013

[11] Oracle: *Oracle Text*; An Oracle Technical White Paper; June 2013

[12] Oracle: *Text Mining with Oracle Text*; An Oracle White Paper; April 2005

[13] Oracle: *Classification, Clustering and Information Visualization with Oracle Text*; An Oracle White Paper; February 2003

[14] Oracle: *Oracle Spatial and Graph*; An Oracle Data Sheet; 2013

[15] Oracle: *Oracle Spatial and Graph: Advanced Data Management*; An Oracle White Paper; September 2014

[16] Oracle: *Oracle Database In-Memory*; An Oracle White Paper; October 2014

[17] Oracle: *Oracle NoSQL Database, 12CR1 Version 3.0, Community Edition*; An Oracle Data Sheet; 2013

Literaturverzeichnis

[18] Oracle: *Oracle NoSQL Database, 12CR1 Version 3.0, Enterprise Edition*; An Oracle Data Sheet; 2013

[19] Oracle: Oracle NoSQL Database Compared to Cassandra
http://www.oracle.com/technetwork/database/database-technologies/nosqldb/documentation/nosql-vs-cassandra-1961717.pdf
Veröffentlicht: nicht bekannt; Abgerufen: 11. Juni 2015

[20] Oracle: Oracle NoSQL Database Compared to Couchbase
http://www.oracle.com/technetwork/database/database-technologies/nosqldb/documentation/nosql-vs-couchbase-1961718.pdf
Veröffentlicht: nicht bekannt; Abgerufen: 11. Juni 2015

[21] Oracle: Oracle NoSQL Database Compared to CouchDB
http://www.oracle.com/technetwork/database/database-technologies/nosqldb/documentation/nosql-vs-couchdb-1961720.pdf
Veröffentlicht: nicht bekannt; Abgerufen: 11. Juni 2015

[22] Oracle: Oracle NoSQL Database Compared to DynamoDB
http://www.oracle.com/technetwork/database/database-technologies/nosqldb/documentation/nosql-vs-dynamodb-1961721.pdf
Veröffentlicht: nicht bekannt; Abgerufen: 11. Juni 2015

[23] Oracle: Oracle NoSQL Database Compared to HBase
http://www.oracle.com/technetwork/products/nosqldb/documentation/nosql-vs-hbase-1961722.pdf
Veröffentlicht: nicht bekannt; Abgerufen: 11. Juni 2015

[24] Oracle: Oracle NoSQL Database Compared to MongoDB
http://www.oracle.com/technetwork/database/database-technologies/nosqldb/documentation/nosql-vs-mongodb-1961723.pdf
Veröffentlicht: nicht bekannt; Abgerufen: 11. Juni 2015

[25] Oracle: Oracle NoSQL Database Compared to Riak
http://www.oracle.com/technetwork/database/database-technologies/nosqldb/documentation/nosql-vs-riak-1961724.pdf
Veröffentlicht: nicht bekannt; Abgerufen: 11. Juni 2015

[26] Overview of the Oracle NoSQL Database
http://dbmsmusings.blogspot.de/2011/10/overview-of-oracle-nosql-database.html
Veröffentlicht: 4. Oktober 2011; Abgerufen: 11. Juni 2015

[27] Oracle: *Oracle Big Data Management System - A Statement of Direction for Big Data and Data Warehousing Platforms*; An Oracle Statement of Direction; April 2015

[28] Oracle stellt neue und erweiterte Produkte für Big-Data-Analysen vor
https://www.oracle.com/de/corporate/pressrelease/enterprise-big-data-20150219.html
Veröffentlicht: 19. Februar 2015; Abgerufen: 12. Juni 2015

[29] Oracle: *Unified Query for Big Data Management Systems - Integrating Big Data Systems with Enterprise Data Warehouses*; An Oracle White Paper; January 2015

Literaturverzeichnis

[30] Konopasek, Klemens: *SQL Server 2014: Der schnelle Einstieg*; Hanser; September 2014

[31] Microsoft: SQL Server 2016 Datasheet
http://download.microsoft.com/download/F/D/3/FD33C34D-3B65-4DA9-8A9F-0B456656DE3B/SQL_Server_2016_datasheet.pdf
Veröffentlicht: nicht bekannt; Abgerufen: 16. Juni 2015

[32] Official Microsoft SQL Server Blog: SQL Server 2016 first public preview now available!
http://blogs.technet.com/b/dataplatforminsider/archive/2015/05/27/sql-server-2016-first-public-preview-now-available.aspx
Veröffentlicht: 27. Mai 2015; Abgerufen: 16. Juni 2015

[33] Microsoft Connect: Add native support for JSON to SQL Server, a la XML (as in, FOR JSON or FROM OPENJSON)
https://connect.microsoft.com/SQLServer/Feedback/Details/673824
Veröffentlicht: 3. Juni 2011; Abgerufen: 16. Juni 2015

[34] XML-Datentyp und XML-Spalten in Microsoft SQL Server
https://msdn.microsoft.com/de-de/library/hh403385.aspx
Veröffentlicht: nicht bekannt; Abgerufen: 16. Juni 2015

[35] MongoDB vs. SQL Server's XML Data Type
http://lifeinvistaprint.com/techblog/mongodb-vs-sql-servers-xml-data-type/
Veröffentlicht: 25. März 2015; Abgerufen: 16. Juni 2015

[36] Use Sparse Columns
https://msdn.microsoft.com/en-us/library/cc280604.aspx
Veröffentlicht: nicht bekannt; Abgerufen: 18. Juni 2015

[37] SQL SERVER – 2008 – Introduction to SPARSE Columns
http://blog.sqlauthority.com/2008/07/10/sql-server-2008-introduction-to-sparse-columns/
Veröffentlicht: 10. Juli 2008; Abgerufen: 18. Juni 2015

[38] Use Column Sets
https://msdn.microsoft.com/en-us/library/cc280521.aspx
Veröffentlicht: nicht bekannt; Abgerufen: 18. Juni 2015

[39] Misner, Stacia; Mistry Ross: *Introducing Microsoft SQL Server 2012*; Microsoft Press; 2012

[40] Misner, Stacia; Mistry Ross: *Introducing Microsoft SQL Server 2014 Technical Overview*; Microsoft Press; 2014

[41] David J. DeWitt, Alan Halverson, Rimma Nehme, Srinath Shankar, Josep Aguilar-Saborit, Artin Avanes, Miro Flasza and Jim Gramling: *Split Query Processing in Polybase*; Microsoft Corporation

[42] Hoffmeister, W.: *Investitionsrechnung und Nutzwertanalyse*, 2. Auflage; Berlin: Berliner Wissenschafts-Verlag; 2008