LANDESBERUFSSCHULE 4 SALZBURG

NoSQL – Big Data

Datenbanken

LBS 4

.

Inhalt

Lernziele und Kompetenzcheck

Ich kann/kenne

- Unterschied zwischen relationalen- und nicht relationalen Datenbanken
- den Aufbau von NoSQL-Systemen
- Anwendungszweck von Big Data und NoSQL-Datenbanken

Einleitung

Der enorme Zuwachs an unstrukturierten Informationen kann mit herkömmlichen Techniken (SQL-Datenbanken) nicht mehr in der geforderten Geschwindigkeit verwaltet und ausgewertet werden. Die Entwicklung von NoSQL begann schon zurzeit als relationale Datenbanken erfolgreich waren. Die ersten waren einfache Key/Value Systeme. NoSQL beschreibt Techniken mit nicht relationalen Ansätzen. Hier werden Daten nicht mehr nur in Tabellen, sondern in dynamischen Strukturen wie Graphen, Dokumenten usw. gespeichert. Mit der Verbreitung von NoSQL kam der Begriff Big Data dazu. Big Data beschreibt einen Weg wie aus unstrukturierten Informationen (gespeichert in NoSQL) Wissen generiert wird und für Entscheidungen verwendet werden kann. Nachstehend werden die wichtigsten Begriffe und Techniken, die im Zusammenhang mit Big Data und NoSQL stehen, erläutert. NoSQL erlaubt einfaches Skalieren von Systemen sowie eine hohe Verfügbarkeit der Daten. Klassische Systeme können riesige Datenmengen nicht effizient verwalten.

CAP-Theorem

Dieser Ansatz klassifiziert verteilte Datenbanksysteme. Zwei von drei der nachstehenden Eigenschaften sind für diese Systeme mindestens erreichbar. Es muss immer einen Kompromiss zwischen den drei Eigenschaften geben.

Konsistenz

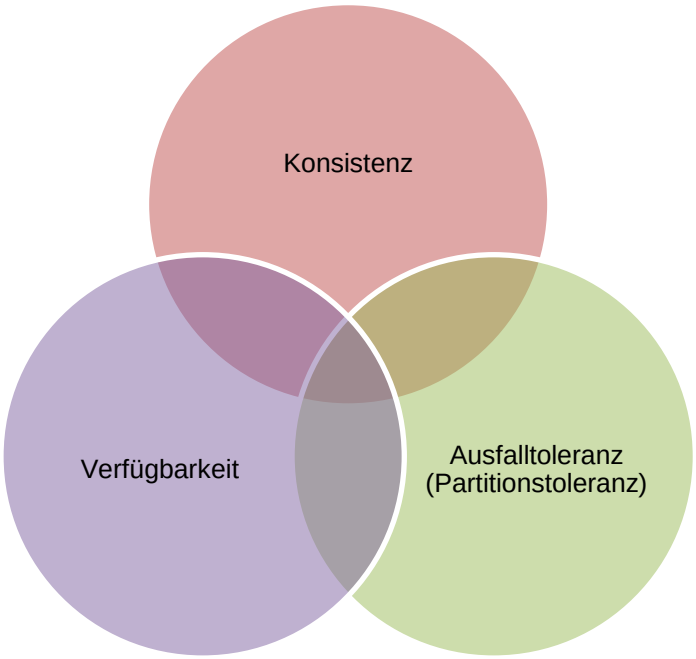
Nach Abschluss einer Transaktion ist der Datenbestand im verteilten System konsistent. Allen Clients steht bei gleichem Zugriff dieselbe Version einer Datei zur Verfügung.

Verfügbarkeit

Die Reaktionszeiten müssen für das verwendete System akzeptabel sein. Jede Anfrage muss ein aktuelles Ergebnis liefern.

Ausfalltoleranz (Partitionstoleranz)

Ein Ausfall eines Knotens in einem System, darf nicht zum Ausfall des Gesamtsystems führen.



Annahme des Theorems

Es gibt ein System, dass die Eigenschaften Konsistenz, Verfügbarkeit und Ausfalltoleranz zur Verfügung stellt.

Beispiel:

An einem Knoten wird gelesen an einem anderen geschrieben. Ein Client kann eine Datei lesen hat aber zu diesem Zeitpunkt nicht die aktuelle Version. Erst nach der Replikation kann die neue Datei am DB-Knoten gelesen werden. In diesem Fall ist das System nicht im konsistenten Zustand, weil es verschiedene Versionen gibt.
Schlussfolgerung: hier wird die Konsistenz geopfert, um die Verfügbarkeit zu gewährleisten.

Big Data

Das Schlagwort steht nicht nur für eine Technologie, sondern auch für eine Sammlung an Werkzeugen, die enorme Mengen an unstrukturierte Information verwalten können. Diese Tools können aus umfassenden Datenbeständen Wissen für Entscheidungen generieren. Vor allem die Echtzeitanalyse von Datenströmen ist für Vorhersagen eine wichtige Grundlage.

Ziel von Big Data ist es, aus diesen Daten wertvolle Erkenntnisse zu gewinnen, die Unternehmen und Organisationen in verschiedenen Bereichen nutzen können, um

- Effizienz zu verbessern
- Kosten zu senken
- neue Produkte und Dienstleistungen zu entwickeln
- Kunden besser zu verstehen
- Geschäftsprozesse zu optimieren
- wettbewerbsfähig zu bleiben

Die Definition von Big Data wird oft mit den vier V’s beschrieben:

volume:

die Anzahl der Daten und Informationen

variety:

beschreibt die Vielfalt der Datentypen die strukturiert oder unstrukturiert sind

velocity:

beschreibt die Ausführungsgeschwindigkeit, in der ein Datenstrom analysiert werden kann

veracity:

die Richtigkeit und die Korrektheit der Daten

Datenquellen

Die Quellen von Big Data sind vielfältig und nehmen stetig zu:

Verkehrsleitsysteme: Sensordaten liefern Informationen über Verkehrsströme, um Staus zu vermeiden und die Infrastruktur zu optimieren.

Smartphones: Sensoren in Smartphones erfassen Daten über Bewegung, Umgebung und Nutzungsverhalten.

Bezahlvorgänge: Transaktionsdaten aus dem Online- und Offline-Handel ermöglichen Analysen des Kaufverhaltens.

Social Media: Posts, Likes und Kommentare in sozialen Netzwerken spiegeln Meinungen, Interessen und Trends wider.

Benutzerprofile: Online-Profile liefern Informationen über Nutzerpräferenzen und -verhalten.

Maschinendaten: Sensoren in Produktionsanlagen, Kraftwerken und anderen Maschinen generieren wertvolle Daten für die Optimierung von Prozessen.

Web-Aktivitäten: Klicks, Seitenaufrufe und andere Nutzungsdaten auf Websites geben Aufschluss über das Nutzerverhalten im Internet.

Nachstehend ein Beispiel für die Vielfalt an möglichen Daten(typen) von Big Data.

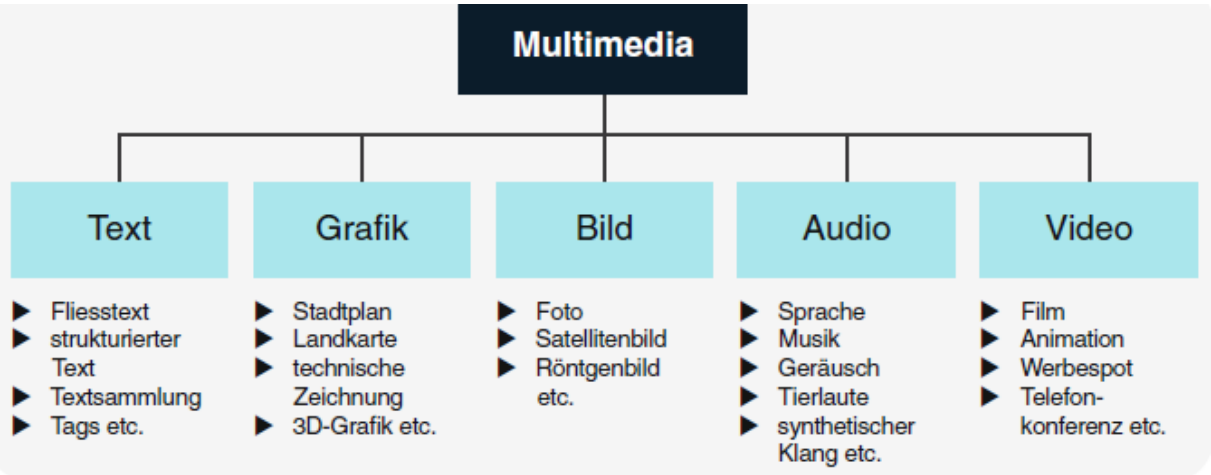


Abbildung 1: Big Data

Die Herausforderungen sind von Big Data sind:

- die Speicherung und Verarbeitung der riesigen Datenmengen
- die Gewährleistung der Datensicherheit und des Datenschutzes
- die Gewinnung relevanter Erkenntnisse aus den Daten
- der ethische Umgang mit Big Data

Datawarehouse

Ein Data Warehouse (DWH) ist weit mehr als nur eine gewöhnliche Datenbank. Es handelt sich um ein strategisches Werkzeug, das Unternehmen dabei unterstützt, ihre Daten gewinnbringend zu nutzen und datenbasierte Entscheidungen zu treffen.

Die Hauptaufgaben sind:

Sammlung und Integration von Daten aus vielfältigen Quellen: Das DWH vereint Daten aus verschiedenen Systemen wie CRM, ERP, Transaktionsdatenbanken und externen Quellen, um ein umfassendes Bild des Unternehmens zu erhalten.

Speicherung strukturierter Daten: Im Gegensatz zu operativen Systemen, die oft unstrukturierte Daten enthalten, speichert ein Data Warehouse Daten in einem standardisierten, strukturierten Format, das für Analysen optimiert ist.

Optimierte Abfrageleistung: Dank ausgefeilter Indexierungs- und Optimierungstechniken ermöglicht ein Data Warehouse schnelle und effiziente Abfragen, auch bei großen Datenmengen.

Transformation und Bereinigung von Daten: Bevor Daten für Analysen verwendet werden können, müssen sie oft transformiert und bereinigt werden. Das Data Warehouse bereinigt Inkonsistenzen, ergänzt fehlende Werte, standardisiert Datentypen und aggregiert Daten, um eine zuverlässige Grundlage für Analysen zu schaffen.

Vorteile eines Data Warehouses:

Unterstützung datenbasierter Entscheidungen: Durch die Bereitstellung einer zentralen, konsistenten Datenquelle ermöglicht ein Data Warehouse detaillierte Analysen und fundierte Entscheidungen.

Effizienzsteigerung: Data Warehouses ermöglichen es Unternehmen, ihre Daten effizienter zu nutzen, indem sie redundante Datenhaltung reduzieren und die Zugriffszeiten auf relevante Informationen verkürzen.

Gewinnung neuer Erkenntnisse: Durch die Zusammenführung von Daten aus verschiedenen Quellen können neue Muster und Zusammenhänge erkannt werden, die in einzelnen Systemen verborgen bleiben.

Förderung von Innovationen: Data Warehouses liefern die Grundlage für datengetriebene Innovationen und die Entwicklung neuer Produkte und Dienstleistungen.

SQL- und NoSQL-Technologien

SQL-Grundlagen

Das Realtionenmodell wurde 1970 von F.C. Codd entwickelt. Als Abfrage und Manipulationssprache wird Structured Query Language (SQL) verwendet. Ein relationales Datenbanksystem besteht zumindest aus einer Datenbank sowie einem Managementsystem wie in nachstehender Abbildung zu sehen ist.

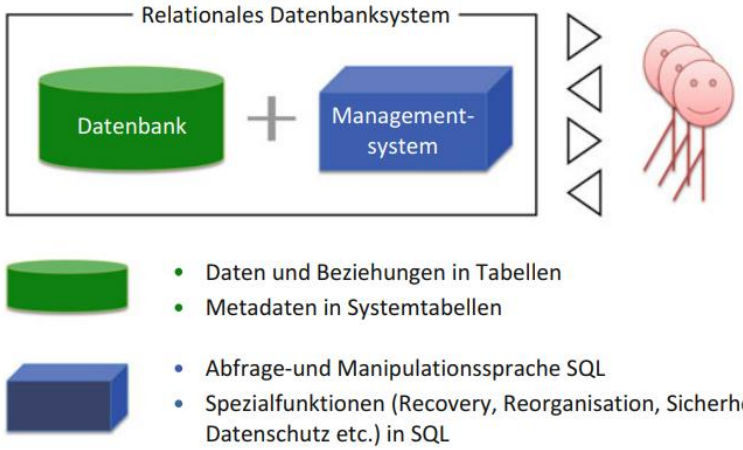


Abbildung 2: Aufbau relationales System

Die Datenbank kann die Relationen (Zeilen und Spalten) sowie die Beziehungen zwischen den Tabellen abspeichern. Die Daten werden ebenfalls in den Tabellen organisiert. Zusätzlich gibt es Systemtabellen, die Beschreibungsinformationen sowie Rechteinformationen von Benutzern verwalten.

Die wichtigsten Aspekte von relationalen-Systemen sind:

Modell

Das Datenmodell basiert auf Tabellen, die in Beziehungen zueinander stehen.

Die Normalformen (NF1, NF2, NF3, NF4, NF5) definieren den Aufbau und die Qualität des Modells, um Datenanomalien zu vermeiden.

Architektur

Die Daten und Verwaltungsprogramme sind unabhängig voneinander.

Die meisten Systeme folgen einer Struktur mit physischer Ebene (Speicherung), logischer Ebene (Datenmodell) und Anwendungsebene (Benutzeroberfläche). Das System gewährleistet die Konsistenz der Daten, auch bei parallelem Zugriff durch mehrere Benutzer. Tabellen werden mit Identifikationsschlüsseln und Regeln zur Gewährleistung der Integrität definiert.

Schema

Tabellen werden mit Identifikationsschlüsseln (Primärschlüsseln) und Spalten definiert, die Datentypen wie Ganzzahlen, Zeichenketten, Datumsangaben usw. enthalten. Beziehungen zwischen Tabellen werden durch Fremdschlüssel definiert. Regeln wie referenzielle Integrität und Entitätsintegrität stellen die Datenkonsistenz sicher.

Sprache

SQL ist die Standardsprache zur Interaktion mit relationalen Datenbanken. Diese ist in drei Gruppen eingeteilt. Bei der Datendefinition werden Datenbanken, Tabellen, Spalten und Integritätsregeln erstellt. Dazu wird die DDL verwendet.

Für die Datenauswahl und zum Ändern von Daten wird die DML verwendet. Befehle sind z.B.: SELECT-, INSERT-, UPDATE- und DELETE-Anweisungen.

Die DCL wird für Benutzerverwaltung und Rechteverwaltung verwendet. Befehle sind z.B.: GRANT, REVOKE, CREATE USER, usw.

Mehrbenutzerbetrieb

Es können mehrere Benutzer gleichzeitig dieselbe Datenbank abfragen. Mechanismen wie Sperren und Transaktionen sorgen für Datenkonsistenz und Integrität. Systeme können skaliert werden, um die Anforderungen einer wachsenden Anzahl von Benutzern und Daten zu erfüllen

Konsistenz

Die Daten werden fehlerfrei und korrekt gespeichert. Integritätsregeln und Mechanismen zur Datenkonsistenz gewährleisten die Genauigkeit und Zuverlässigkeit der Daten.

Anwendung von relationalen Datenbanken

Relationale Datenbanken sind ideal für strukturierte Daten, die in Tabellen organisiert sind. RDB sind die häufig genutzte Wahl für KMU, da sie eine robuste, skalierbare und einfach zu verwaltende Lösung bieten. Weiters bei Webanwendungen zur Speicherung von Benutzerdaten, Produktdaten und anderen strukturierten Daten.

Angesichts der rasanten Zunahme unstrukturierter Daten ist eine Erweiterung der relationalen Datenbanktechnologie erforderlich, insbesondere im Bereich von Web- und Big-Data-Anwendungen, durch die Integration von NoSQL-Systemen.

NoSQL-Grundlagen

NoSQL (Not only SQL) verwendet keine starren Modelle zur Datenhaltung. Der Begriff wird sehr oft für nicht relationale Ansätze verwendet und beschreibt zwei Bedingungen:

- die Speicherung der Daten erfolgt nicht (immer) in Tabellen
- die Datenbanksprache ist (meistens) nicht SQL

Die Entwicklung von NoSQL-Technologien wurde vor allem für technische- und wissenschaftliche Anwendungen durchgeführt. So war es schwierig Teile von technischen Objekten (Maschinenbau) in einzelnen Spalten und Zeilen zu verwalten, da diese sehr oft verändert und zur Laufzeit manipuliert wurden. Mit der Zunahme von webbasierten Anwendungen stieg die Zahl der Datenformate und auch die Menge an Informationen, die gespeichert werden mussten. NoSQL-Systeme sind normalerweise verteilte Architekturen mit verschiedenen Speicherstrukturen. Nachstehende Abbildung zeigt ein mögliches System.

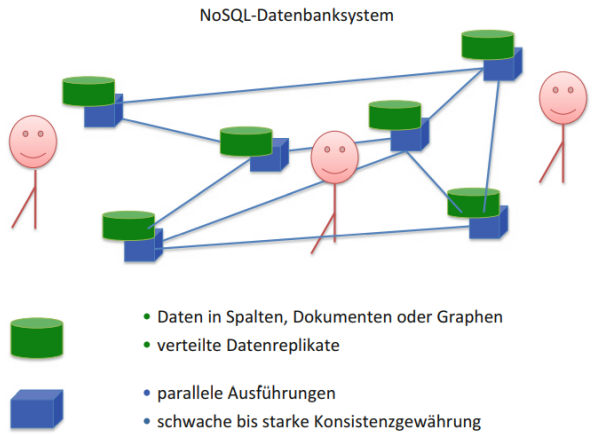


Abbildung 3: NoSQL-Technologie

Diese Systeme unterstützen spezielle Replizierungsverfahren sowie parallele Auswertungsverfahren.

NoSQL-Systeme haben genauso wie relationale Ansätze einen bestimmten Aufbau.

Modell

NoSQL-Systeme verwenden nicht-relationale Datenmodelle, die Daten in vielfältigen Formaten wie JSON, BSON, Dokumenten, Graphen oder Schlüssel-Wert-Paaren speichern können.

Architektur

Die meisten NoSQL-Systeme sind verteilt aufgebaut, d.h. die Daten werden auf mehrere Server verteilt, um die Skalierbarkeit und Leistung zu verbessern. Horizontale Skalierung ist möglich, indem weitere Server hinzugefügt werden, um mit wachsenden Datenmengen und Lasten umzugehen.

Schema

Im Gegensatz zu relationalen Datenbanken mit ihren fixen Schemata bieten NoSQL-Systeme flexible Schemata, die sich mit den Daten weiterentwickeln können. Das ermöglicht eine dynamische Anpassung an neue Datenstrukturen und Anwendungsanforderungen.

Replikation

NoSQL-Systeme unterstützen Dateireplikation, um Datenredundanz zu gewährleisten und die Verfügbarkeit im Falle von Serverausfällen zu erhöhen. Replikationsstrategien können je nach System variieren.

Mehrbenutzerbetrieb

NoSQL-Systeme müssen den Mehrbenutzerbetrieb unterstützen, da sie in der Regel in webbasierten Anwendungen und anderen Szenarien mit mehreren gleichzeitigen Benutzern eingesetzt werden

Konsistenz

NoSQL-Systeme bieten unterschiedliche Konsistenzmodelle, wobei **zumindest verzögerte Konsistenz** gewährleistet sein muss. Im Gegensatz zu relationalen Datenbanken mit strikter ACID-Konsistenz, bieten NoSQL-Systeme oft „Eventual Consistency“, die eine gewisse Verzögerung bei der Datenaktualisierung zulässt, um die Leistung und Skalierbarkeit zu verbessern.

Nachstehend einige Beispiele von NoSQL Speicherstrukturen..

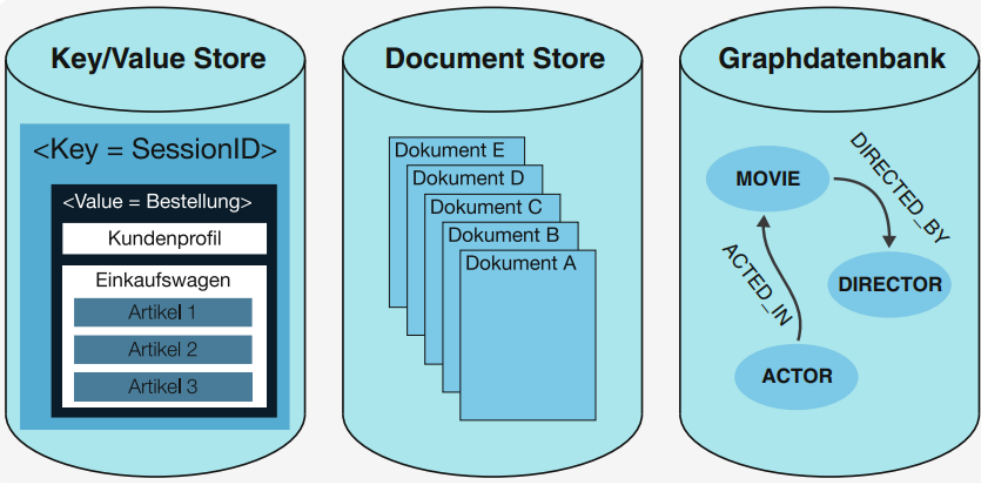


Abbildung 4: Arten von Speicherstrukturen

Key-Value Modell

Schlüssel-Wert-Datenbanken sind die einfachsten Systeme und speichern die Informationen in Form einer ID und einer Wertzuweisung. Ein Beispiel könnte ein Webshop sein der als Key die Session-ID verwendet und als Wert die Artikel referenziert. Der Schlüssel muss allerdings eindeutig sein. Durch den einfachen Aufbau können keine komplexen Operationen wie eine Produktbildung von Datensätzen durchgeführt werden.

Der Aufbau ist in nachstehender Abbildung zu sehen.

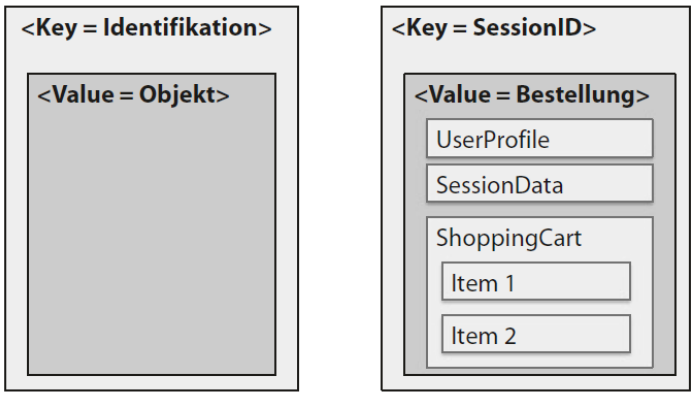


Abbildung 5. Aufbau Key/Value

Ein Key/Value Store kann einfach mit dem `set` Befehl einen Wert zuweisen. Hier wird für den Schlüssel User: U17547 der Vorname, Nachname, usw. gespeichert.

```
SET User:U17547:vorname Max
SET User:U17547:nachname Mueller
SET User:U17547:email max.mueller@blue_planet.net
SET User:U17547:pwhash D75872C818DC63BC1D87EA12
SET User:U17548:vorname Mina
```

Dokumentendatenbank

Die Informationen werden in JSON-ähnlichen Dokumenten gespeichert, die keine fixen Attribute besitzen. Dieser flexible Ansatz ermöglicht eine Vielzahl von Anwendungsmöglichkeiten, wie z. B. Kataloge, Benutzerprofile oder Content-Management-Systeme (CMS). Ein großer Vorteil ist die Möglichkeit, das Schema selbst zu erweitern und neue Attribute hinzuzufügen. Jedoch kann dies bei der Anforderung von Normalformen oder referenzieller Integrität zu Herausforderungen führen.

Das nachstehende JSON-Dokument ist eine gültige Darstellung eines Benutzerprofils mit verschachtelten Objekten für die Adresse:

```
{
   vorname : „Daniel“,
   nachname : „Fasel“,
   email : „df@scigility.com“,
   addresse : {
                    strasse : „Rte du soleil 22“,
                    ort : „Fribourg“,
                    plz : 1700,
                    land : „Switzerland“
                },
   sprache : „deutsch“
 }
```

Datenbanken die Dokument-Stores verwenden sind MongoDB oder CouchDB.

Graphen-Datenbank

Graphenmodelle werden überall dort verwendet, wo netzwerkartige Strukturen analysiert werden müssen. Ein Graph besteht aus einem Knoten und mindestens einer Kante, die wieder einem anderen Knoten zugeordnet ist. Die Anzahl der Graphen bestimmt den Weg von einer Information zur anderen. Viele Problemstellungen werden mittels Graphentheorie gelöst.

Grundlegende Konzepte

Ein Graph besteht aus zwei fundamentalen Elementen:

- Knoten: Repräsentieren einzelne Entitäten innerhalb des Netzwerks
- Kanten: Verbinden Knoten miteinander und symbolisieren die Beziehung zwischen ihnen

Die Anzahl der Kanten, die mit einem Knoten verbunden sind, wird als Grad des Knotens bezeichnet. Je nach Art der Verbindung können Graphen gerichtet oder ungerichtet sein. In gerichteten Graphen fließt die Information entlang der Kanten in eine bestimmte Richtung, während sie in ungerichteten Graphen in beide Richtungen fließen kann.

Anwendungen

Die Graphentheorie bietet eine Vielzahl von Algorithmen und Methoden zur Analyse von Graphenmodellen. Diese finden Anwendung in zahlreichen Bereichen, wie:

- Routingprotokolle: Bestimmung des effizientesten Pfads für die Datenübertragung in Netzwerken
- Logistik: Optimierung von Lieferrouten und Transportvorgängen
- Soziale Netzwerkanalyse: Untersuchung von Beziehungen und Interaktionen in sozialen Netzwerken
- Bioinformatik: Modellierung von Proteinstrukturen und Genregulationsprozessen
- Verkehrsmanagement: Optimierung von Verkehrsströmen und Reduzierung von Staus

Beispiel: die Kante SPIELT ist gerichtet und zeigt vom Knoten DARSTELLER auf den Knoten FILM.

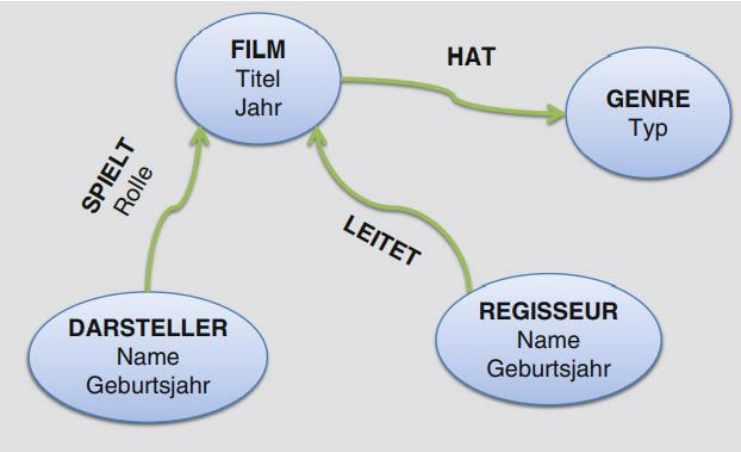


Abbildung 6: Graphen-System

Graphen-Datenbanken werden überall da eingesetzt, wo viele semantische Zusammenhänge ausgedrückt werden können. Als Abfragesprache dient z.B.: Cypher.

Nachstehende Abfrage zeigt das Ergebnis, in welchen Filmen spielte Tom Hanks mit.

```
MATCH (film:Film)-[:GESPIELT_IN]->(schauspieler:Schauspieler) WHERE schauspieler.name = "Tom Hanks" RETURN film
```

In-Memory-Datenbanken

NoSQL-Datenbanken in Kombination mit In-Memory-Technologien bieten erhebliche Vorteile für Anwendungen, die schnelle Zugriffszeiten und Echtzeit-Analysen erfordern. Im Folgenden wird die Funktionsweise und die Anwendungsgebiete dieser Kombination näher erläutert.

Funktionsweise

In-Memory-Datenbanken speichern Daten im Arbeitsspeicher (RAM) des Computersystems, anstatt sie auf Festplatten oder anderen Speichermedien abzulegen. Dies ermöglicht deutlich schnellere Lese- und Schreibvorgänge im Vergleich zu traditionellen Festplatten-basierten Datenbanken. Diese Systeme können problemlos an wachsende Datenmengen und Anforderungen angepasst werden.

Vorteil

Die Verwendung von NoSQL mit In-Memory-Datenbanken bietet folgende Vorteile:

- deutlich schnellere Zugriffszeiten und geringere Latenzzeiten im Vergleich zu Festplatten-basierten Datenbanken
- ermöglicht die Analyse von Datenströmen in Echtzeit, z. B. für Betrugserkennung, Systemüberwachung oder personalisierte Empfehlungen
- einfache Skalierung auf neue Server, um steigende Anforderungen zu bewältigen

Beispiele dafür sind, Redis, Apache Ignite oder Memcached.