Abstract Graphdatenbanken

Autor: Thi Minh Phuong Pham
Modul: Oberseminar Datenbanksysteme
Betreuer: Prof. Dr.-Ing. Thomas Kudrass

30. Juni 2017

1 Motivation

Graphdatenbanken sind in unserer heutigen Gegenwart in unterschiedlichen Lebensbereiche vertreten, darunter gehören Social Media, hierbei werden die Verbindungen zwischen Personen in Form von Knoten und Kanten miteinander verknüpft und abgebildet. In der Logistik werden mithilfe von Graphen die kürzesten und effizientesten Wege analysiert. Des Weiteren werden in Bereiche wie Chemie und Bioglogie anhand der Graphenstrukturen die Verbindungen zwischen Atomen untersucht. Mithilfe von Graphendatenbanken lassen sich somit reale Sachverhalte speichern, ohne künstliche Konstrukte zu erzeugen [9]. Graphdatenbanken gehören neben Key- Value Datenbanken, Spalten- und Dokumentenorientierte Datenbanken zu NOSQL- Datenbanken[4].

1.1 Definition

Graphen bestehen aus einer Menge von Knoten und Kanten, aus diesen werden Graphenstrukturen gebildet. Hierbei verbinden Kanten die Knoten untereinander und bilden die Beziehungen der Knoten ab. Die vernetzten Informationen können in Graphendatenbanken gespeichert werden[2].

1.2 Anwendung in der Praxis

Die Anwendung von Graphen in der Praxis sind vielfältig. So werden diese in Sozialen Netzwerken verwendet, darunter zählen Facebook, Instagram und Twitter. Die Graphenstrukturen bestehen in dem Fall aus Freunden oder Bekannten, die als Knoten abgebildet werden. Die Kanten bilden die Beziehungen zwischen den jeweilgen Freunden oder Bekannten ab[5].
Des Weiteren findet man Graphenstrukturen in technologischen Netzwerken wieder. In der Logistik werden verschiedene Algorithmen verwendet, mit dem Zweck Kosten einzusparen und die kürzesten Strecke zu ermitteln. Zu den Algorithmen gehört auch Hub & Spoke, dabei existiert ein zentralen Knotenpunkt, von der aus abgehende Zweige zu anderen Knotenpunkten führen. Hierbei sollen Leerfahrten und Leerflüge vermieden werden. Hinzu soll die Auslastung optimiert werden. Straßennetzwerke, Zug- und Flugnetzwerke verwenden ebensfalls Hub & Spoke und viele weitere Algorithmen[10].
In Biologischen Netzwerken sind ebensfalls Graphenstrukturen vorzufinden. So sind diese für die Umlauf der Netzwerke und Induktionsgesetze von Bedeutung. Bindungen zwischen den Atomen können näher betrachtet und beurteilt werden[8].

2 Graphen und deren Eigenschaften

2.1 Graphenaufbau

Graphen bestehen aus Knoten und Kanten, die jeweils Informationen enthalten. Ein Knoten ist ein Objekt. Die Kanten stellen die Beziehungen zwischen den Objekten dar. Ein Graph kann auch als Tupel G=(V,E) dargestellt werden, wobei V ein Knoten und E die jeweilige Kante ist. Eine Kante hat weiterhin die Eigenschaft, Kantengewichte zu besitzen. Der Graph wird damit als gewichteter Graph bezeichnet und speichert Informationen, wie groß die Distanz beziehungsweise die Transportdauer zwischen zwei Objekten ist[9].

2.2 Graphdatenmodell

Ein Property Graph Model bezeichnet einen Graphen, deren Knoten und Kanten zusätzliche Eigenschaften besitzen, sognannnte Properties. In den Kanten und Knoten sind nicht nur Label gepeichert, sondern auch Key/Value Beziehung[7]. Mehrere Eigenschaften können Knoten oder Kanten detailliert charakterisieren. Zur Bestimmung der eindeutigen Identität eines Knoten, kann dieser eine ID besitzen[3].

2.3 Graphoperationen

Die Grundoperationen in der Graphdatenbank umfassen die Create-, Read-, Update- und Delete-Operation. Mit dem Create-Operation wird ein Knoten erzeugt, mit Read wird es gelesen, mit Update wird der Knoten aktualisiert und mit Delete gelöscht[3]. Eines der wichtigsten Operation in der Graphdatenbanktheorie ist die Traversierung. Hiebei werden Graphen durchlaufen. Ziel ist es, ein Knoten zu passieren und mit einem bestimmten Wert zu initialisieren. Ausgehend von einzelnen Knoten oder Knotenmenge wird jeder Knoten und Kante einmal durchlaufen[3]. Die bekanntesten Algorithmen der Traversierung sind Breiten- und Tiefensuche. Mit der Breitensuche werden die Graphen durchsucht, indem zuerst alle Nachbarknoten des Startknoten durchsucht werden. Im Anschluss werden alle Nachbarn des Nachbarknoten durchlaufen. Im Gegensatz zu der Breitensuche, werden bei der Tiefensuche einen Graphen solange durchlaufen, bis ein Knoten keine ausgehende Kanten mehr hat[6].

2.4 Vergleich Relationale Datenbanken mit Graphdatenbanken

Im Gegensatz zu Relationale Datenbanken werden die Daten in Graphdatenbanken direkt vernetzt, gespeichert und gelesen. Dabei werden Knoten und Kanten miteinander verbunden, wobei die Kanten die Beziehungen darstellen.
Die Relationale Datenbank ist im Pendant dazu durch Relationen und Tupel gekennzeichnet. Jede Zeile ist ein Datensatz mit Attributswerten. Dabei werden die Beziehungen zwischen den Tabellen mit einem Schlüssel verbunden. Weiterhin legt ein Relationenschema die Anzahl und den Typ der Attribute für eine Tabelle fest. Komplexe Anfragen erfolgen durch rekursiv geschachtelte Join. Die gängige Abfragesprache ist hierbei SQL[2].
Bei Graphdatenbanken werden auf Joins vezichtet, da effiziente Traversierung, wie die Breiten- oder Tiefensuche, existieren. Des Weiteren gibt es keine einheitliche Abfragesprache, bekannte Abfragesprachen sind Cypher und GraphQL(SQL ähnliche Abfragesprache)[2].
Die Frage, welche von den beiden Datenbanksystemen man wählt, hängt von den Datensätzen ab. Möchte man eine Beziehungsdatenbank, so überwiegen die Vorteile einer Graphendatenbank. So verzichtet man hierbei auf Join Operationen und hat somit einen schnellen Zugriff auf die Daten. Hingegen dazu möchte man, dass die Daten unabhängig und SQL-fähig ist, so wäre eine Relationale Datenbank geeigneter[2].

3 Graphendatenbankmanagementsystem

In Graphdatenbankmanagementsysteme sind Datenbank, die Datensätze in Form von Knoten und Kanten verwalten. Sie werden häufig auch als Graphdatenbanken oder graphorientierte Datenbank genannt. Die bekanntesten Vertreter sind Neo4j, Titan und Giraph[1].

3.1 Neo4j

Neo4J ist eine in Java implementierte OpenSource Graphdatenbank. Des Weiteren ist es ein Natives Graphdatenkmanagementsystem und wurde seit 2004 entwickelt. Die erste Version 1.0 wurde 2010 veröffentlicht. Die Anfragsprache ist hierbei Cypher oder Gremlin[11].

3.2 Demonstrationsbeispiel Neo4j

Bei dem Demonstationsbeispiel wurde die Community-Variante von Neo4j vorgestellt, dabei wurden die grundlegenden Funktionen gezeigt. Die Modellierung und Speicherung von Daten in Form von Knoten und Kanten wurde schrittweise erläutert.

3.2.1 Cypher Query Language

- **Erstellen eines Knoten Melanie**:

```
CREATE (n:Person {name:"Melanie"}) RETURN n
```

- **Der Knoten Jack bekommt eine Eigenschaft**:

```
MATCH (n:Person {name:"Jack"}) SET n.age=34
```

- **Beziehungen zwischen zwei Personen erstellen**:

```
MATCH (a:Person {name:"Jack"}),(b:Person {name:"Jill"})
MERGE (a)-[r:MARRIED]->(b)
```

- **Beziehungen mit Property zwischen zwei Personen erstellen**:

```
MATCH (a:Person {name:"Melanie"}), (b:Person {name:"Jill"})
MERGE (a)-[r:FRIENDS {since:"2001"}]->(b)
```

- **Alle Personen finden, die einen Alter von 34 haben**:

```
MATCH (n:Person) WHERE n.age=34 RETURN n
```

- **Alle Personen finden, die mit Melanie befreundet sind**:

```
MATCH (a:Person{name:"Melanie"})-[:FRIENDS]->(b:Person)
RETURN a,b
```

- **Löschen von Knoten**:

```
MATCH (n:Person) DELETE n
```

3.2.2 Game Of Thrones Beispiel

- **CSV Datei laden**:

```
LOAD CSV WITH HEADERS
FROM
"https://www.macalester.edu/~abeverid/data/stormofswords.csv"
AS row
MERGE (src:Character {name: row.Source})
MERGE (tgt:Character {name: row.Target})
MERGE (src)-[r:INTERACTS]->(tgt)
ON CREATE SET r.weight = toInt(row.Weight)
```

- **Ausgabe der Zahl aller Charaktere**:

```
MATCH (c:Character) RETURN count(c)
```

- **Matche alle kürzesten Pfade zwischen Catelyn & Drogo**:

```
MATCH (catelyn:Character {name: "Catelyn"}),
(drogo:Character {name: "Drogo"})
MATCH p=shortestPath((catelyn)-[INTERACTS*]-(drogo))
RETURN p
```

Literaturverzeichnis

[1] DB-Engines: *Graph DBMS*. https://db-engines.com/de/article/Graph+DBMS. Version: 2017

[2] FHKöln: *Graphenmodell*. http://www.datenbanken-verstehen.de/lexikon/graphdatenbanken/. Version: 2011

[3] Gross, D. A.: *NoSQL-Datenbanken*. https://dbs.uni-leipzig.de/file/NoSQL_SS17_06_Graph. Version: 2017

[4] Ha, T. N.: *Charakteristika und Vergleich von SQL- und NoSQLDatenbanken*. https://dbs.uni-leipzig.de/file/seminar_1112_tran_ausarbeitung.pdf. Version: 2011

[5] Kaufmann, T. T. M. G. A. K. A. T. Moritz; Muhlbauer M. Moritz; Muhlbauer: *Hochperformante Analysen in Graph-Datenbanken*. https://db.in.tum.de/~muehlbau/papers/analysegraphdatenbanken.pdf. Version: 2017

[6] Lexicon, D. O.: *Breiten- und Tiefensuche*. http://lwibs01.gm.fh-koeln.de/wikis/wiki_db/index.php?n=Datenbanken.BreitenTiefensuche. Version: 2011

[7] Lexikon, D. O.: *Das Property-Graph-Modell*. http://wikis.gm.fh-koeln.de/wiki_db/Datenbanken/PropertyGraphModell. Version: 2011

[8] Nieselt, K. : *Einführung in die Bioinformatik*. https://ab.inf.uni-tuebingen.de/teaching/ss2012/ebi/SS12_EBI5_BiolNetzwerke.pdf. Version: 2012

[9] verstehen.de datenbank: *Graphdatenbanken*. http://www.datenbanken-verstehen.de/lexikon/graphdatenbanken/. Version: 2017

[10] wiki: *Hub and Spoke*. https://de.wikipedia.org/wiki/Hub_and_Spoke. Version: 2017

[11] wiki: *Neo4j*. https://de.wikipedia.org/wiki/Neo4j. Version: 2017