Technische Universität Dortmund

Fakultät Maschinenbau

Fachgebiet IT in Produktion und Logistik

Bachelorarbeit

Entwicklung eines Graphmining-Konzeptes unter Verwendung von MATLAB im Supply Chain Management

Katharina Langenbach

Studiengang	Maschinenbau
Matrikelnummer	196084
Thema ausgegeben am	04.01.2021
Arbeit eingereicht am	28.03.2021
Erstprüfer	Dr.-Ing. Dipl.-Inform. Anne Scheidler
Zweitprüfer	Joachim Hunker, M. Sc.

Inhaltsverzeichnis

1 Einleitung 1

2 Wissensentdeckung im Kontext der Supply Chain 3
2.1 Supply Chain 3
2.2 Daten und Datenstrukturen 5
2.3 Datenbanken 8
2.3.1 Relationale Datenbanken 9
2.3.2 Nichtrelationale Datenbanken 11

3 Verarbeitung von Graphstrukturen 14
3.1 Grundlagen der Wissensentdeckung 14
3.2 Graphmining Tools 17

4 Entwicklung eines Graphmining-Konzeptes in MATLAB zur Analyse von Daten aus der Supply Chain Domäne 19
4.1 Kontextbezogene Vorüberlegungen 21
4.2 Datenvorbereitung 23
4.3 Bewertung der Graphmining-Algorithmus-Ergebnisse 26

5 Prototypische Anwendung des Graphmining-Konzeptes 31
5.1 Vorstellung des Fallbeispiels 31
5.2 Anwendung 32
5.3 Diskussion und Fazit 43

6 Zusammenfassung und Ausblick 47

Literaturverzeichnis 50

Abbildungsverzeichnis 55

Tabellenverzeichnis 56

Abkürzungsverzeichnis 57

Eidesstattliche Erklärung 58

1 Einleitung

Supply Chains (SCs) sind in der heutigen Zeit komplexe Systeme, die stetigen Veränderungen unterworfen sind (Hunker *et al.*, 2020). Um diese Systeme überwachen und steuern zu können, werden in wachsendem Umfang Datenmengen erhoben und gespeichert (Tanase *et al.*, 2018). Innerhalb von SCs, bspw. im Supply Chain Management (SCM), werden dazu in der Regel relationale Datenbanken bzw. relationale Datenstrukturen verwendet (Drakopoulos *et al.*, 2015). Dabei zählen relationale Datenbanken aktuell zu den dominierenden Datenbanken (Garcia-Molina *et al.*, 2009; Hunker *et al.*, 2020).

Bei Systemen in der SC Domäne handelt es sich um Netzwerke (Junghanns *et al.*, 2016; Hunker *et al.*, 2020). Netzwerke lassen sich aufgrund ihrer Struktur als Graph interpretieren, sodass die Speicherung in Graphen naheliegend ist (Junghanns und Petermann, 2016; Junghanns *et al.*, 2016; Anikin *et al.*, 2019; Kovács *et al.*, 2019). Bei der Speicherung von Graphen stellen Graphdatenbanken eine rasch wachsende Technologie dar (Junghanns *et al.*, 2017; Ezhilchelvan *et al.*, 2020). Die Darstellung von Daten in Graphstrukturen basiert auf der Graphentheorie (Robinson, 2015). Dabei werden Daten über Knoten und Kanten abgebildet (Junghanns und Petermann, 2016; Ezhilchelvan *et al.*, 2020). Durch Graphstrukturen können bspw. Beziehungen zwischen heterogenen Daten, also z. B. Daten, die unterschiedliche Strukturen oder Formate aufweisen, einfacher und schneller als in relationalen Datenstrukturen dargestellt und abgefragt werden (Junghanns und Petermann, 2016; Junghanns *et al.*, 2017; Rostami *et al.*, 2019; Ezhilchelvan *et al.*, 2020). Um das angestrebte Wissen aus Daten für das SCM mit Methoden der Wissensentdeckung gewinnen zu können, müssen Daten effizient verarbeitet werden. Dabei wächst die Bedeutung des Data Minings (Bissantz und Hagedorn, 2009). Eine spezifische Form des Data Minings im Hinblick auf Daten in Graphstrukturen ist das Graphmining (Rehman *et al.*, 2012), bei welchem Daten aus Graphstrukturen direkt basierend auf Graphen analysiert werden und nicht zunächst in relationale Datenstrukturen überführt werden müssen (Klöckner, 2015). Zur direkten Verarbeitung von Graphen können unterschiedliche Programme, wie z. B. MATLAB, genutzt werden.

Um eine Graphanalyse durchführen zu können, ist im Kontext eines SCMs ein Konzept notwendig, das die Besonderheiten von Daten in Graphstrukturen berücksichtigt. Ziel dieser Arbeit ist die Entwicklung eines Graphmining-Konzeptes unter Verwendung des Programms MATLAB zur Verarbeitung von Daten in Graphstrukturen aus dem Bereich

der SC. Weitere Ziele sind, die Möglichkeiten der Verwendung von MATLAB zum Graphmining zu analysieren, die Anforderungen an die verwendeten Daten zu definieren und ein für SC Anwendungen generisches Vorgehen zu entwickeln, sowie seine Anwendbarkeit zu überprüfen. Um dies zu erreichen, ist insbesondere die Datenstruktur für Daten aus der SC Domäne zu untersuchen.

Zur Erstellung eines Graphmining-Konzeptes müssen zunächst Grundlagen der Anwendungsdomäne (SCs und SCM) und der Wissensentdeckung in diesem Kontext erarbeitet werden. Im Zuge dessen sind auch Daten, Datenstrukturen und unterschiedliche Datenbanktypen zu erläutern, um im späteren Verlauf der Arbeit die Vor- und Nachteile einzelner Datendarstellungen nutzen zu können. Gleiches gilt bezüglich der Erläuterung unterschiedlicher Programme zur Darstellung und Analyse von Graphstrukturen. Durch die Untersuchung unterschiedlicher Programme lassen sich Rückschlüsse auf die Anforderungen an Graphanalyse- und Graphmining-Tools ziehen. Dies ermöglicht zusammen mit grundlegenden Betrachtungen von Graphstrukturen in Kombination mit Anforderungen im SCM die Entwicklung eines Graphmining-Konzeptes. Dabei wird auf die Besonderheiten des verwendeten Tools, sowie auf die notwendige Datenstruktur bzw. die notwendige Datenaufbereitung eingegangen. Wurde dieser Schritt erfolgreich angewandt, kann mit der gezielten Datenverarbeitung bzw. Analyse begonnen werden. Auf diese Weise entsteht ein für SC Anwendungen einsetzbares Konzept. Um abschließend die Anwendbarkeit des entwickelten Konzeptes bewerten zu können, wird dieses prototypisch auf ein Anwendungsbeispiel angewandt. Die daraus abgeleiteten Erkenntnisse werden im Fazit diskutiert. Zudem wird das gesamte Konzept im Fazit kritisch hinterfragt, sodass sich Anhaltspunkte zur weiteren Entwicklung und Verbesserung des Konzeptes ergeben.

2 Wissensentdeckung im Kontext der Supply Chain

Im Folgenden werden die für die Untersuchung der Problemstellung benötigten Grundlagen erarbeitet. Dabei wird insbesondere auf die Anwendungsdomäne SC, sowie auf die Grundlagen von Daten und Datenbanken eingegangen.

2.1 Supply Chain

Der Begriff SC ist in der Literatur nicht einheitlich definiert. Dabei unterscheiden sich die Definitionen insbesondere im Umfang und der Art von Elementen, die in einer SC integriert sind. Das APICS Dictionary (Blackstone, 2010, S. 148) definiert eine SC z. B. folgendermaßen: „The global network used to deliver products and services from raw materials to end customers through an engineered flow of information, physical distribution, and cash." Dieser Definition lässt sich entnehmen, dass eine SC grundsätzlich aus Informations-, Waren- und Geldflüssen vom Rohmaterial über die Herstellung eines Produktes bis zum Endverbraucher besteht. Zudem verdeutlicht das Wort „network", dass es sich bei einer SC nicht, wie der Name vermuten lässt, um eine einzelne Kette, sondern um ein komplettes Netzwerk handeln kann. In einem solchen Netzwerk werden Verbindungen zwischen unterschiedlichen Herstellern und Dienstleistern aufgespannt. Eine weitere Definition einer SC nach Mentzer *et al.* (2001, S. 4) lautet: „[…] a supply chain is defined as a set of three or more entities (organizations or individuals) directly involved in the upstream and downstream flows of products, services, finances, and/or information from a source to a customer." Bei dieser Definition wird deutlich, dass Informations-, Waren- und Geldflüssen nicht nur in eine Richtung fließen. Das aufgespannte Netzwerk entwickelt sich somit nicht ausschließlich in Wertsteigerungsrichtung. Durch die nicht einheitliche Stromrichtung der Elemente steigt der Komplexitätsgrad des Netzwerkes. Die SC wird zusätzlich erweitert, wenn die Definition einer SC nach Chopra und Meindl (2004, S. 4) berücksichtigt wird: „A supply chain consists of all parties involved, directly or indirectly, in fulfilling a customer request." Demnach sind neben den von Mentzer *et al.* (2001) beschriebenen direkt involvierten auch indirekt involvierte Elemente zu berücksichtigen. Unter Elementen sind dabei bspw. unterschiedliche Unternehmen (Mentzer *et al.*, 2001; Stadtler *et al.*, 2015), Einzelpersonen (Mentzer *et al.*, 2001) oder

auch Wertsteigerungsstufen (Günther und Tempelmeier, 2003) zu verstehen. Je nach Definition werden unterschiedliche Mindestanforderungen an eine SC und an die enthaltenen Elemente gestellt. So sind bspw. nach Stadtler *et al.* (2015) mindestens zwei und nach Mentzer *et al.* (2001) mindestens drei Elemente notwendig, um eine SC aufzuspannen. In der allgemeinen Betrachtung von SCs ist zu beachten, dass SCs auf unterschiedlichen Ebenen aufgespannt werden können, bspw. innerhalb einer einzelnen Fabrik aber auch über mehrere Fabriken hinweg.

In der deutschsprachigen Literatur gibt es unterschiedliche Übersetzungen für den Begriff der SC. Arnold *et al.* (2008) verwenden als Übersetzung den Begriff Logistikketten. Logistikketten beschreiben sie dabei folgendermaßen: „In betrieblichen Logistikketten spiegeln sich die Zusammenhänge des Material- und Güterflusses im Beschaffungs-, Produktions- und Vertriebsbereich von Unternehmen wider“ (Arnold *et al.*, 2008, S. 160). Hierbei wird deutlich, dass der Begriff Logistikketten unspezifischer als SC in der englischsprachigen Literatur gefasst ist.

Insgesamt lässt sich den unterschiedlichen Definitionen einer SC entnehmen, dass Entscheidungen, die zunächst nur direkt auf ein einzelnes Element wirken, auch direkten oder indirekten Einfluss auf die anderen Elemente der SC haben. Es müssen somit nicht nur interne, sondern auch externe Auswirkungen von Entscheidungen berücksichtigt werden. Zusätzlich sind, bei Entscheidungen die SC betreffend, die teils heterogenen Interessenslagen der involvierten Parteien zu berücksichtigen, was die Komplexität der Entscheidungen zusätzlich erhöht (Kleijnen, 2005; Law, 2015; Stadtler *et al.*, 2015).

Im weiteren Verlauf dieser Arbeit wird die Definition einer SC nach Chopra und Meindl (2004) verwendet, da in dieser alle direkt und indirekt involvierte Parteien eingeschlossen sind. Dadurch wird die SC allumfassend abgebildet. Zudem werden in der Arbeit immer die Informations-, Waren- und Geldflüsse berücksichtigt.

Aufgrund des hohen Komplexitätsgrades von SCs ist ein SCM notwendig (Stadtler *et al.*, 2015). Dabei wird unter SCM „die unternehmensübergreifende Koordination und Optimierung der Material-, Informations- und Wertflüsse“ (Arndt, 2008, S. 47) verstanden. Auf diese Weise sollen die Prozesse in einer SC zeit- und kostenoptimal gestaltet werden (Christopher, 2016). Damit das SCM adäquate Entscheidungen für die Entwicklung der SC treffen kann, ist es notwendig, dass die Auswirkungen der Entscheidungen auf die SC schon im Voraus möglichst gut absehbar sind (Stadtler *et al.*, 2015). Um dies zu ermöglichen, werden viele Daten in der SC erhoben. Daten allein bieten keine ausreichende Entscheidungsunterstützung. Für eine solche Entscheidungsunterstützung ist die Zusammenführung, Aufbereitung,

und Analyse der Daten notwendig (Fayyad und Uthurusamy, 1996; Wirth und Hipp, 2000). Typische Fragestellungen im Bereich des SCMs beschäftigen sich u. a. mit dem Ressourcenmanagement, der Lieferantenstruktur und den Lieferzeiten (Hahn und Kaufmann, 2002; Hitt, 2011; Wannenwetsch, 2014).

2.2 Daten und Datenstrukturen

In der heutigen Zeit werden in allen Lebensbereichen, sowohl im Privatleben, in der Wissenschaft, als auch in der Industrie, immer mehr Daten und Informationen gesammelt (Steiner, 2017; Tanase *et al.*, 2018). Daten bilden die Grundlage der Wissensentdeckung und -gewinnung (Frawley *et al.*, 1992; Fayyad und Uthurusamy, 1996). Informationen und Wissen spielen in immer mehr Bereichen der SC eine entscheidende Rolle (Dippold *et al.*, 2005; Hildebrand *et al.*, 2015). So sind sowohl Bereiche der Produktion, als auch der Planung und Strategie involviert (Hildebrand *et al.*, 2015). Zudem hat sich gezeigt, dass je höher die Qualität eines Produkts und je innovativer es ist, desto stärker ist der Anteil an Informationen und Wissen ausgeprägt, der in den Entstehungsprozess eingebracht werden muss (Dippold *et al.*, 2005).

Es sind hierbei die Begriffe Wissen, Information und Daten zu unterschieden (Kudraß, 2015). Aus Daten können durch Zuordnung einer Bedeutung, also einer Interpretation, Informationen abgeleitet werden (Kudraß, 2015). Werden diese so gewonnenen Informationen anschließend vernetzt und in einen Kontext gesetzt, entsteht aus den Informationen Wissen (North, 2011). Diese Beziehungen zwischen Daten, Informationen und Wissen finden sich auch in der Wissenstreppe in Anlehnung an North (2016) in Abbildung 1 wieder. Zudem wird in der Wissenstreppe deutlich, dass aus Daten Handlungen abgeleitet werden können, indem das abgeleitete Wissen an passender Stelle angewendet wird.

„Daten sind das inhaltliche Element einer Information“ (Hildebrand *et al.*, 2015, S. 143) und bestehen aus nach Regeln geordneten Symbolen (Hildebrand *et al.*, 2015). Prinzipiell lassen sich strukturierte, unstrukturierte und semistrukturierte Daten unterscheiden (Kudraß, 2015). Damit sind Daten wertungsfrei und ohne Interpretation nicht weiter verwendbar. Aus diesem Grund ist es notwendig, dass die Daten zum richtigen Zeitpunkt, an der richtigen Stelle vorliegen, sodass durch kontextbezogenes Wissen wertsteigernde

Informationen aus ihnen gewonnen werden können (Dippold *et al.*, 2005). Damit unterscheiden sich Daten im Wesentlichen nicht von anderen Produktionsfaktoren, die auch zur richtigen Zeit am richtigen Ort verfügbar sein müssen (Dippold *et al.*, 2005).

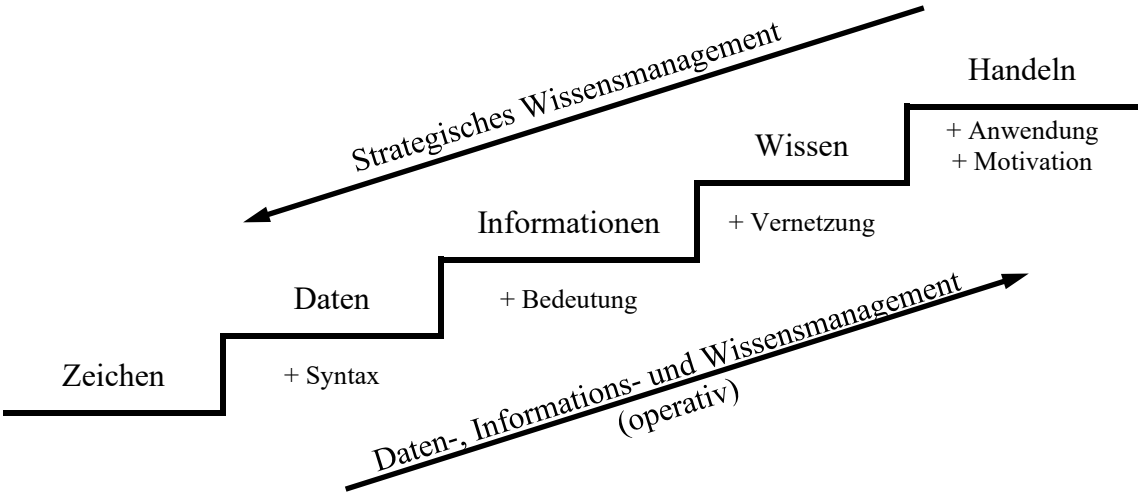

Abbildung 1: Wissenstreppe. Darstellung der Zusammenhänge zwischen Zeichen, Daten, Information, Wissen und Handeln in Anlehnung an North (2016)

Da wie in Kapitel 1 beschrieben die Bedeutung von Daten stetig wächst, ist u. a. auch auf die Datenqualität zu achten (Hazen *et al.*, 2014). Unter Datenqualität lässt sich dabei „[…] die Gesamtheit der Ausprägungen von Qualitätsmerkmalen eines Datenbestandes bezüglich dessen Eignung, festgelegte und vorausgesetzte Erfordernisse zu erfüllen“ (Hildebrand *et al.*, 2015, S. 88), verstehen. In der Arbeit wird im Folgenden davon ausgegangen, dass die verwendeten Daten den Datenqualitätsmerkmalen nach Cai und Zhu (2015) entsprechen.

Um Daten nutzen zu können, müssen diese zudem in einem geeigneten Format, in Datenstrukturen, gespeichert werden (Ottmann und Widmayer, 2012). Je nachdem welche Informationen aus den Daten gewonnen werden sollen, können einige Datenstrukturen Vorteile gegenüber anderen aufweisen (Ottmann und Widmayer, 2012). Dabei ist unter Datenstruktur die „Organisationsform für eine Menge von Daten“ (Ottmann und Widmayer, 2012, S. 24) zu verstehen. In Tabelle 1 sind einige häufig verwendete Datenstrukturen aufgeführt und kurz erläutert. Dabei ist zu beachten, dass die einzelnen Datenstrukturen zum Teil aufeinander aufbauen können.

Tabelle 1: Überblick über Datenstrukturen

Datenstruktur	**Erläuterung**	**Quelle**
Datenfeld / Array	Ein Array besteht aus einer festgelegten Anzahl von Index-Wert-Paaren. Alle Elemente des Arrays müssen vom gleichen Datentyp sein.	Mehta und Sahni, 2018; Will, 2018
Datensatz / Record	Ein Datensatz besteht aus mehreren zusammengehörigen Datenfeldern.	Mertens *et al.*, 2017
Verkettete Liste	In einer Liste können beliebig viele Elemente gespeichert werden. Dabei besteht ein Element aus den abgelegten Daten und einem Pointer auf das nächste Element. Somit bildet eine verkettete Liste eine Alternative zum Array.	Ottmann und Widmayer, 2012; Mehta und Sahni, 2018
Baum	Bäume bestehen aus Knoten, die über Kanten verbunden sind. Ein Baum kann als Graph ohne Kreisschlüsse angesehen werden. Dabei sind mehrere Spezifikationen von Bäumen wie z. B. orientierte, freie und binäre Bäume möglich. Über Bäume lassen sich z. B. hierarchische Informationen gut abbilden. Die Konstruktion eines Baumes kann auf eine Liste zurückgeführt werden.	Celko, 2014; Mehta und Sahni, 2018
Graph	Ein Graph besteht aus Knoten, die über (gerichtete und/ oder ungerichtete) Kanten verbunden sind. Durch die Verbindungen können Beziehungen zwischen Knoten abgebildet werden. Ein Graph kann als freier Baum interpretiert werden.	Celko, 2014; Mehta und Sahni, 2018
Stapelspeicher / Stack	In einem Stapelspeicher werden Elemente nacheinander abgespeichert. Dabei werden neu hinzugefügte Elemente immer an das gleiche Ende angehängt. Am gleichen Ende können Elemente auch nur entfernt werden. Ein Stapelspeicher funktioniert also nach dem Last-In-First-Out-Prinzip.	Mehta und Sahni, 2018; Will, 2018
Warteschlange / Queue	Eine Warteschlange funktioniert ähnlich wie der Stapelspeicher. Der Unterschied besteht darin, dass neue Elemente an einem Ende hinzugefügt und am anderen Ende entfernt werden. Eine Warteschlange funktioniert also nach dem First-In-First-Out-Prinzip.	Mehta und Sahni, 2018; Will, 2018

Bei der Betrachtung von SC-Strukturen spielen insbesondere Stamm-, Bestands- und Transaktionsdaten eine Rolle. Dabei sind Stammdaten „grundlegende Informationen […], die unabhängig von der Materialbewegung anfallen“ (van Bonn, 2013, S. 293). Es handelt sich dementsprechend um Daten, die sich kaum bis gar nicht ändern und feste Größen im System einer SC definieren. Ein Beispiel für Stammdaten sind Kundenstamm- und Standortdaten (van Bonn, 2013). Bestandsdaten sind Daten, die „Mengen- und Wertestrukturen in der Datenhaltung kennzeichnen“ (Scheidler, 2017, S. 17). Bestandsdaten ändern sich somit während Prozesse in der SC ablaufen. Die größte und letzte Hauptgruppe an Daten bilden die Transaktionsdaten (Scheidler, 2017). Mit Hilfe von Transaktionsdaten lassen sich Bewegungsvorgänge beschreiben (Scheidler, 2017). Bewegungsvorgänge, also Transaktionen, von Objekten beschreiben nach Corsten und Gössinger (2008) den Übergang zwischen einzelnen Akteuren. Mit Hilfe der Transaktionsdaten lassen sich also Änderungen an den Bestandsdaten nachvollziehen.

Stamm-, Bestands- und Transaktionsdaten weisen im Allgemeinen keinen einheitlichen Datentyp auf (Scheidler, 2017). Zudem ist bei der Zusammenführung solcher Daten aus unterschiedlichen Quellen auf die Eindeutigkeit von Bezeichnungen zu achten bzw. es muss ausreichend Kontextwissen über die Kodierung vorhanden sein.

2.3 Datenbanken

Wie in Abschnitt 2.2 beschrieben, gibt es eine Vielzahl unterschiedlicher Datenstrukturen. Daten können in Datenbanken gespeichert werden. In diesem Zusammenhang sind insbesondere die Begriffe Datenbank, Datenbanksystem (DBS) und Datenbankmanagementsystem (DBMS) zu unterscheiden. Unter einer Datenbank wird „[…] eine Sammlung von Daten, die einen Ausschnitt der realen Welt beschreiben“ (Kudraß, 2015, S. 20), verstanden. Die so abgelegten Daten stehen dabei zueinander in Beziehung (Schicker, 2014). Die Anzahl der beinhalteten Daten ist generell nicht limitiert (Preiß, 2007). Insgesamt lässt sich eine Datenbank als „nothing but a collection of organized data“ (Batra, 2018, S. 1) beschreiben. Dem gegenüber steht das DBMS. „Ein Datenbankmanagementsystem (DBMS) ist ein Softwaresystem […], das dem Benutzer das Erstellen und die Pflege einer Datenbank ermöglicht. Dies umfasst die Definition, die Erzeugung und Manipulation von Datenbanken“ (Kudraß, 2015, S. 21). Das DBMS ist von großer Bedeutung, da dabei auch das Datenmodell der Datenbank festgelegt wird (Kudraß, 2015). Das

Datenmodell bestimmt wiederum die Datenstruktur der gespeicherten Objekte, die möglichen Operationen, die auf den Daten ausgeführt werden können, und die Integritätsbedingungen, die die zulässigen Zustände definieren (Silberschatz *et al.*, 2011; Kudraß, 2015). Die Datenbank bildet zusammen mit dem DBMS ein DBS (Unland und Pernul, 2014; Kudraß, 2015; Saake *et al.*, 2018). Durch DBS können Daten effizient organisiert und verarbeitet werden (Kudraß, 2015).

Es lassen sich je nach Aufbau und genutzter Datenstruktur unterschiedliche Datenbankmodelle unterscheiden, z. B. relationale, objektorientierte, hierarchische und netzartige Datenbankmodelle (Schicker, 2014; Kudraß, 2015). Grundsätzlich lässt sich zwischen dem vorherrschenden relationalen und den nichtrelationalen Datenbankmodellen unterscheiden. Im Folgenden wird auf diese zwei Unterscheidungen eingegangen.

2.3.1 Relationale Datenbanken

Relationale Datenbanken zählen derzeit zu den wichtigsten Datenbankmodellen und bilden den defacto Standard bei Datenbankanwendungen (Garcia-Molina *et al.*, 2009; Schicker, 2014; Steiner, 2017; Batra, 2018; Tanase *et al.*, 2018).

Die Datenbasis einer relationalen Datenbank bilden Tabellen (Steiner, 2017). Die Tabellen werden als Relationen bezeichnet (Schicker, 2014). Die Tabellen stehen in Beziehung, wobei die Beziehungen in den Tabellen mit Hilfe von Primär- und Fremdschlüsseln abgespeichert werden (Schicker, 2014; Steiner, 2017). Die theoretische Grundlage für dieses Datenbankmodell bilden das Relationenkalkül und die relationale Algebra (Schicker, 2014).

Ein Überblick über wichtige Begriffe in Bezug auf relationale Datenbanken ist basierend auf Steiner (2017) in Tabelle 2 zusammengestellt.

Tabelle 2: Grundlegende Begriffe in Bezug auf das relationale Datenbankmodell

Begriff	**Erklärung**
Entität	Eine Entität beschreibt ein Themengebiet und umfasst Elemente, die die gleichen Merkmale aufweisen. Eine Entität kann somit als Tabellenname interpretiert werden. Dazugehörige Datensätze werden als Entitätsmenge bezeichnet.

Tabelle 2: Grundlegende Begriffe in Bezug auf das relationale Datenbankmodell (Fortsetzung)

Begriff	Erklärung
Relation	Eine Entität bildet zusammen mit der zugehörigen Entitätsmenge eine Relation, also eine Tabelle, die alle beigeordneten Werte beinhaltet.
Attribut	Ein Attribut entspricht einem Spaltennamen der Relation und beschreibt somit ein Merkmal eines abgespeicherten Datensatzes. Unter einem Attributwert lässt sich somit das dem Attribut zugeordnete Datum des Datensatzes verstehen.
Domäne	Synonym kann der Begriff Wertebereich verwendet werden. Die Domäne schränkt die zulässigen Attributwerte ein.
Nullwerte	Wenn ein Datensatz für ein Attribut keinen Attributwert besitzt, enthält der Datensatz an dieser Stelle einen Nullwert.

Eine Stärke des relationalen Datenbankmodells liegt in der hohen Flexibilität gegenüber Änderungen und Ergänzungen (Steiner, 2017). Zudem basieren relationale Datenbanksysteme auf standardisierten Prozessen und weisen eine einheitliche Abfragesprache auf (Batra, 2018). Um diese Abfragesprache nutzen zu können, ist aber zwangsläufig das relationale Format notwendig, was auch bedeutet, dass Daten, falls sie bspw. aus objektorientierten Quellen stammen, erst in das entsprechende Tabellenformat überführt werden müssen (Klöckner, 2015). Der Prozess des Umformens kann dabei komplex und aufwendig sein (Leavitt, 2010). Zudem ist die Abfragesprache nur für strukturierte Daten nutzbar (Leavitt, 2010). Eine andere Schwäche des relationalen Datenbankmodells ist, dass Daten teils nur redundant abgespeichert werden können (Schicker, 2014). Zudem wird das System der Tabellen durch jede neu hinzugefügte Tabelle immer schwerer überschaubar (Steiner, 2017). Ein weiterer Nachteil des relationalen Datenbankmodells liegt darin, dass bei Abfragen häufig Daten aus mehreren Tabellen zusammengeführt werden müssen, sodass die für die Abfrage benötigte Zeit bei komplexen Datenstrukturen hoch ist (Steiner, 2017). Zudem sind rekursive Abfragen nur schwer bis gar nicht mit relationalen Datenbanksystemen zu realisieren (Klöckner, 2015).

2.3.2 Nichtrelationale Datenbanken

Neben den dominierenden relationalen Datenbanken gibt es viele andere Datenbanken, die auf unterschiedlichen Datenbankmodellen beruhen. Im Folgenden werden exemplarisch hierarchische, Key-Value, spaltenorientierte Datenbankmodelle, Dokumentendatenbankmodelle und das Prinzip des Graphdatenbankmodells kurz vorgestellt.

Dabei werden zum einen die Verbindungen zwischen unterschiedlichen Datenbankmodellen aufgezeigt und gleichzeitig einige Kernmodelle aus dem Bereich der NoSQL-Datenbanken vorgestellt.

Hierarchische Datenbanken gehören zu den ältesten Datenbanken und ähneln in ihrem Aufbau den relationalen Datenbanken. Die Daten werden in einer Tabelle gespeichert, wobei die erste Spalte die Hierarchiestufe und die zweite Spalte die Daten der jeweiligen Hierarchiestufe abbildet (Steiner, 2017). Diese Struktur ist starr. Sie lässt sich nur mit viel Aufwand erweitern und an neue Anforderungen und Umstände anpassen (Steiner, 2017).

Spaltenorientierte Datenbanken ähneln ebenfalls relationalen Datenbanken, allerdings werden hier Daten spaltenweise auf dem Speicher abgelegt (Klöckner, 2015; Khasawneh *et al.*, 2020). Spaltenorientierte Modelle sind insbesondere dann effizient, wenn ganze Spalten ausgelesen werden sollen. Dabei kann direkt auf die relevante Spalte zugegriffen werden (Klöckner, 2015). Es muss also nicht wie bei relationalen Modellen aus jeder Zeile das entsprechende Datum ermittelt werden. Damit ist die spaltenorientierte Datenbank eine Spezialisierung der relationalen Datenbank für Anwendungen, für die nur auf wenige Spalten aber viele Zeilen zugegriffen werden muss.

Die Key-Value-Datenbank basiert auf Schlüssel-Wert-Paaren (Störl, 2015). Die Daten, die als Wert-Anteil gespeichert werden, können ein beliebiges Datenformat aufweisen und die darin abgelegten Daten können in einer strukturierten, semi- oder unstrukturierten Form vorliegen (Störl, 2015). Dies hat keinen Einfluss auf den Schlüssel, da der Schlüssel vollkommen unabhängig von der Datenstruktur des zugeordneten Werts ist, was einen Flexibilitätsgewinn darstellt (Klöckner, 2015). Im Gegensatz zu einer relationalen Datenbank müssen Daten so nicht in ein festes Schema übertragen werden.

Die Dokumenten-Datenbank ähnelt der Key-Value-Datenbank (Fasel, 2016). Auch bei Dokumenten-Datenbanken werden Schlüssel-Wert-Paare abgelegt (Meier und Kaufmann, 2016). Der Wert-Anteil wird dabei von Dokumenten gebildet (Meier und

Kaufmann, 2016; Khasawneh *et al.*, 2020). Unter Dokumenten ist dabei eine Datei mit einer inneren Struktur (Edlich *et al.*, 2011) zu verstehen. Damit bildet die Dokumenten-Datenbank eine Kombination aus der Flexibilität der Key-Value-Datenbank und der Strukturiertheit der relationalen Datenbank.

Das Graphdatenbankmodell basiert auf der mathematischen Graphen-Theorie (Celko, 2014; Robinson, 2015). Dabei werden Zusammenhänge über Knoten und Kanten abgebildet, wobei die Knoten den Entitäten eines relationalen Datenbankmodells entsprechen (Celko, 2014). Einem Knoten können beliebig viele Eigenschaften zugewiesen werden (Celko, 2014). Kanten, somit die Verbindungen zwischen Knoten, können, aber müssen nicht, gerichtet sein (Celko, 2014; Meier, 2016). Kanten können wiederum eigene Eigenschaften aufweisen bzw. ausdrücken (Edlich *et al.*, 2011; Celko, 2014).

In Tabelle 3 werden grundlegende Begriffe aus dem Bereich des Graphdatenbankmodells vorgestellt.

Tabelle 3: Grundlegende Begriffe in Bezug auf das Graphdatenbankmodell nach Celko (2014)

Begriff	**Erläuterung**
Null Graph	Der Graph besteht ausschließlich aus Knoten.
Vollständiger Graph	In dem Graphen ist jeder Knoten mit jedem anderen Knoten verbunden.
Verbundener Graph	In dem Graphen kann jeder Knoten durch einen Weg erreicht werden
Weg	Die Strecke an Kanten, die ein Satz von Knoten ohne Redundanz an durchlaufenden Kanten verbindet, wird als Weg bezeichnet.
Pfad	Ein Pfad ist ein Weg, wobei jeder Knoten nur einmal durchlaufen werden darf.
Kreis	Es wird im Kontext eines Graphen von einem Kreis gesprochen, wenn ein Pfad wieder zu seinem Ausgangspunkt zurückführt.

Graphen werden bspw. über eine Adjazenzmatrix und Adjazenzlisten aufgestellt (Celko, 2014; Mehta und Sahni, 2018). Dabei drückt eine Adjazenzmatrix „einen Graphen als Matrix aus und gibt an, welcher Knoten mit welcher Kante verbunden ist“ (Meier und

Kaufmann, 2016, S. 70). Auf diese Weise können sowohl symmetrische als auch asymmetrische Beziehungen umgesetzt werden. Zudem kann durch die Werte in der Matrix eine Gewichtung der Beziehungen vorgenommen werden.

Eine Adjazenzliste ist nach Mehta und Sahni (2018, S. 51) wie folgt definiert: „The adjacency list [..] consists of an array Adj of n linked lists, one for each vertex in G, such that Adj[υ] for vertex υ consists of all vertices adjacent to υ." Das bedeutet, dass zu jedem Knoten eine Liste erstellt wird, die die Verbindungen zu den anderen Knoten enthält. Diese Listen werden wiederum in einem Datenfeld abgespeichert. Auf diese Weise werden direkt die Verbindungen an jedem Knoten gespeichert, sodass die Effizienz von Abfragen gesteigert wird (Edlich *et al.*, 2011).

Das Graphdatenbankmodell fokussiert sich im Gegensatz zum relationalen Datenbankmodell auf die Beziehung zwischen den Daten anstatt nur auf die Daten selbst (Celko, 2014). Aus diesem Grund lassen sich komplexe Analysen, bei der die Beziehungen zwischen den Daten fokussiert werden, besser mit Graphdatenbankmodellen als mit relationalen Datenbankmodellen durchführen (Meier und Kaufmann, 2016). Die Abfragen für derartige Analysen in relationalen Datenbanken sind komplex und es besteht die Gefahr in einer endlosen Schleife gefangen zu sein (Celko, 2014). Auch wenn die Daten nicht in einem einheitlichen Format vorliegen, können Daten direkt in einen Graphen eingebettet werden (Celko, 2014). Dies ist bei Relationenstrukturen nicht möglich, da neue Daten gegen die bisherigen geprüft und an die vorliegende Relationenstrukturen angepasst werden müssen (Celko, 2014).

3 Verarbeitung von Graphstrukturen

Wie in Abschnitt 2.2 und Abschnitt 2.3.2 beschrieben, lassen sich netzwerkartige Strukturen durch Graphstrukturen und somit in Graphdatenbanken abbilden. Um aus den so abgespeicherten Daten einen Informationsgewinn ziehen zu können, müssen diese Daten allerdings noch in eine Graphdatenbank implementiert und anschließend basierend auf einer Fragestellung analysiert werden. Ein solches Vorgehen fällt in den Bereich der Wissensentdeckung.

3.1 Grundlagen der Wissensentdeckung

Wie bereits in Kapitel 1 beschrieben, werden immer mehr Daten erfasst. Bei diesen Daten handelt es sich vorwiegend um Rohdaten, die unverarbeitet abgespeichert werden (Frawley *et al.*, 1992). Um einen Nutzen aus diesen Daten ziehen zu können, müssen die Daten mit Hilfe von Computern analysiert werden (Frawley *et al.*, 1992). Aus diesem Grund nimmt die Wissensentdeckung bzw. Datenmustererkennung seit den 1990er Jahren einen immer größeren Stellenwert ein (Bissantz und Hagedorn, 2009).

Ein Stichwort, das in Bezug auf die Wissensentdeckung häufig fällt, ist Data Mining. Data Mining wird bereits in unterschiedlichen Bereichen praktiziert, bspw. im Marketing, bei der Auswertung von sozialen Netzwerken, in militärische Anwendungen, im Finanzsektor und in der Biologie und Medizin (Frawley *et al.*, 1992; Bissantz und Hagedorn, 2009). Data Mining wird allerdings auch häufig synonym mit „Knowledge Discovery in Databases“ (KDD), also der Wissensentdeckung in Datenbanken, verwendet (Jungermann, 2009). Unter Wissensentdeckung wird nach Frawley *et al.* (1992, S. 58) folgendes verstanden: „Knowledge discovery is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data.“ Data Mining wird nach Kleissner (1998, S. 296) wie folgt definiert: „Data mining is a new decision support analysis process to find buried knowledge in corporate data and deliver understanding to business professionals.“ Damit stellt das Data Mining einen Teil bzw. Unterabschnitt des KDD dar. Die Wissensentdeckung kann mehrere unterschiedliche Aufgaben erfüllen. Eine Hauptaufgabe ist die Gruppierung und Identifizierung von Daten (Frawley *et al.*, 1992). Als weitere Aufgaben können die Zusammenfassung von Daten durch Beschrei-

bung von Gemeinsamkeiten, die Differenzierung von Daten durch Beschreibung von Unterschieden in Daten und der Vergleich von Daten identifiziert werden (Frawley *et al.*, 1992). Nach Rehman *et al.* (2012) ist zusammenfassend das Hauptziel des Data Minings versteckte und nützliche Datenmuster aus einem sehr großen Datensatz zu extrahieren.

Wie schon in der Definition des Data Minings angedeutet, ist das Data Mining und darauf aufbauend die Wissensentdeckung, nicht als eine einzelne Aktion zu verstehen, sondern als Prozess (Kleissner, 1998; Kugran und Musilek, 2006). Die Durchführung einer Wissensentdeckung in Form von KDD kann in mehrere Abschnitte unterteilt werden. Im Folgenden wird eine Unterteilung nach Kleissner (1998) vorgestellt. Zunächst erfolgt die Datenauswahl, in der die zu analysierenden Daten ausgewählt und bereitgestellt werden. Anschließend erfolgt eine Bereinigung der Daten, in der bspw. Duplikate aus den Daten entfernt werden. Die darauffolgende Datenanreicherungs- und Kodierungs-Phase dient dazu, dass möglichst nur aussagekräftige Daten analysiert werden. Als letzte Phase des KDD kann das Data Mining verstanden werden. Wie das übergeordnete KDD lässt sich das Data Mining in mehrere Phasen unterteilen. Diese Phasen sind in Abbildung 2 grafisch dargestellt und im Folgenden erläutert. Die Hauptphasen des Data Minings sind die Studiendefinition, die Entdeckung, die Verfeinerung und abschließend die Vorhersage. In der Studiendefinition wird zunächst das Ziel der Datenuntersuchung festgelegt, um anschließend relevante Daten auswählen zu können. Zur Auswahl der Daten gehört unter anderem die Wahl der Datenquelle(n) und einer Strategie zum Umgang mit fehlenden Daten. Anschließend ist ein Mining Algorithmus wie z. B. Clusteranalyse, induktives Lernen oder Bayes (Bissantz und Hagedorn, 2009) zu wählen. In der Phase der Entdeckung wird der gewählte Algorithmus auf die ausgewählten Daten angewendet. In der Verfeinerungsphase werden die Ergebnisse der vorherigen Phase ausgewertet und bewertet. Es kann auch sein, dass Parameter der vorherigen Phase angepasst werden müssen, sodass ein iterativer Prozess entsteht. In der Verfeinerungsphase wird mehr Verständnis für die Daten entwickelt. In der letzten Phase werden basierend auf den gewonnenen Erkenntnissen Vorhersagen getroffen. Diese Vorhersagen können anschließend auf Daten angewendet werden, die nicht im Data Mining Prozess involviert waren. Damit fasst Kleissner (1998) den Bereich des Data Minings weiter als bspw. Fayyad und Uthurusamy (1996), die unter Data Mining nur die Anwendung von speziellen Algorithmen verstehen.

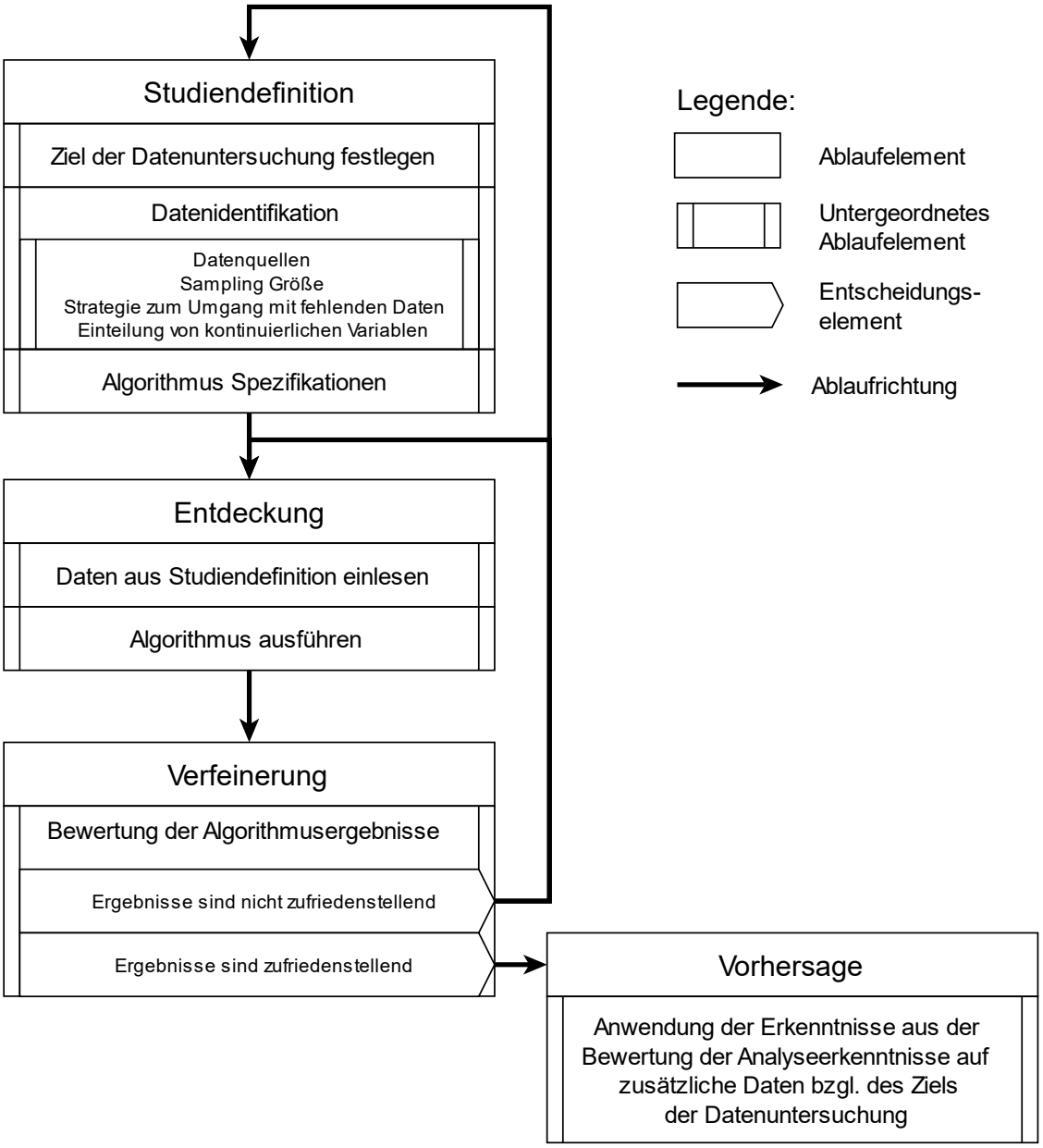

Abbildung 2: Data Mining Prozess. Ablauf eines Data Mining Prozesses nach Kleissner (1998)

Da die meisten Datenbanksysteme ein relationales Datenmodell aufweisen, werden auch Data Mining Prozesse meist auf diesen durchgeführt (Chen *et al.*, 1996; Džeroski, 2003; Scheidler, 2017). Insbesondere bei komplexen, stark verknüpften Daten, wie sie bspw. in SCs vorliegen, ist das Data Mining auf Basis von relationalen Datenbanken schwierig und langwierig. Daher bietet sich die Verarbeitung von Daten auf Basis von Graphstrukturen an (Drakopoulos *et al.*, 2015). Data Mining auf der Basis von Graphstrukturen wird auch als Graphmining bezeichnet und bildet somit eine Spezifikation des Data Minings (Rehman *et al.*, 2012).

3.2 Graphmining Tools

Das Vorgehen beim Graphmining unterscheidet sich nicht vom Vorgehen vom Data Mining auf relationalen Datenbanken. Grundsätzlich werden beim Graphmining auch Ansätze der Klassifizierung, der Gruppierung und Entscheidungsbäume verwendet (Rehman *et al.*, 2012) bspw. unter Verwendung von Nachbarschaftssuchen (Angelova, 2009). Um die Nutzung von Graphdatenbanken beim Data Mining, also das Graphmining, zu unterstützen und zu vereinfachen, wurden unterschiedliche Tools entwickelt, die auch Graphmining Prozesse unterstützen. Im Folgenden werden drei Tools exemplarisch kurz vorgestellt.

RapidMiner ist eine Open Source Data Mining-Plattform (Kotu und Deshpande, 2014). Zudem bietet RapidMiner die Möglichkeiten zum maschinellen Lernen, Text Mining, voraussagender Analytik und Geschäftsanalyse (Geetha und Nasira, 2014). Bei RapidMiner können Operatoren zur Verarbeitung von Daten kombiniert werden, die jeweils eine einzelne Aktion ausführen (Ristoski *et al.*, 2015). Innerhalb von RapidMiner wird eine XML Repräsentierung der Daten verwendet, um sicherzustellen, dass ein standardisiertes Format bei der Übertragung zwischen den einzelnen Operatoren verwendet wird (Han *et al.*, 2008). RapidMiner bietet darüber hinaus Möglichkeiten der Erweiterung und Einbindung in andere Strukturen (Geetha und Nasira, 2014; Ristoski *et al.*, 2015).

Der Konstanz Information Miner (KNIME) ist ein Tool, das die interaktive Analyse von Daten ermöglicht (Berthold *et al.*, 2009). Das Tool ist eine Open-Source-Software, wodurch die Einbindung in bereits vorhandene Strukturen erleichtert wird (Berthold *et al.*, 2009; Haun *et al.*, 2010). Die zu analysierenden Daten können aus unterschiedlichen Quellen, wie bspw. unterschiedlichen Datenbanksystemen mit unterschiedlichen Datenstrukturen, stammen und auch parallel verarbeitet werden (Berthold *et al.*, 2009; Feltrin, 2015). In KNIME werden Daten zwischen unterschiedlichen Knoten transferiert (Berthold *et al.*, 2009). An diesen Knoten können jeweils vordefinierte Aktionen durchgeführt werden, die durch die Verbindung von unterschiedlichen Knoten kombiniert und angepasst werden können (Berthold *et al.*, 2009; Haun *et al.*, 2010). Knoten können dabei als Verarbeitungszentren interpretiert werden. Durch diese Art der Verarbeitung, können Daten bzw. die verarbeiteten Varianten dieser, an den unterschiedlichen Knoten abgegriffen werden (Berthold *et al.*, 2009). Innerhalb von KNIME werden Daten in „DataTable“-Strukturen gespeichert (Berthold *et al.*, 2009). Dies bedeutet, dass auch Daten, die aus einem Graphdatenbanksystem stammen, in ein Tabellenformat überführt werden.

MATLAB ist eine hauptsächlich von Ingenieuren und Wissenschaftlern genutzte Programmierumgebung und -sprache, die auf die numerische Lösung von Problemen ausgelegt ist (MathWorks, 2020b). Es handelt sich dabei primär um eine lineare Algebra Software (Sahu *et al.*, 2017). Ein großer Vorteil von MATLAB liegt in der Möglichkeit der parallelen Durchführung von Prozessen (MathWorks, 2020b). Zudem ist es möglich MATLAB über die Database-Toolbox mit dem Neo4j Datenbanksystem zu verbinden (MathWorks, 2020a), das zu den populärsten Graphdatenbanksystemen gehört (Celko, 2014; Sahu *et al.*, 2017). Neo4j verfügt über eine Programmierschnittstelle (Angles, 2012), sodass mithilfe von MATLAB die in dem Datenbanksystem gespeicherten Daten manipuliert und analysiert werden können. Auf diese Weise können die Funktionen von MATLAB und Neo4j kombiniert werden.

Neo4j bietet die Möglichkeit Daten aus unterschiedlichen Quellen und mit unterschiedlichen Datenstrukturen in einen Graphen zu überführen (Neo4j, 2020). Auf einen in Neo4j vorliegenden Graphen kann über MATLAB zugegriffen werden (MathWorks, 2020a). Innerhalb von MATLAB können Beziehungen zwischen Knoten analysiert werden (MathWorks, 2020a). Zusätzlich ist es möglich Knoten und Kanten zu erstellen, zu löschen oder zu modifizieren (MathWorks, 2020a). Der so bearbeitete bzw. analysierte Graph kann zurück in die Neo4j Datenbank exportiert werden (MathWorks, 2020a). In MATLAB können eigene Analyseroutinen verfasst werden (MathWorks, 2020b). Die Ergebnisse können direkt in MATLAB dargestellt werden (MathWorks, 2020a, 2020b). Mit Hilfe von weiteren Toolboxen innerhalb von MATLAB können auch bspw. zusätzlich statistische Analysen von Daten zugeführt werden (MathWorks, 2020a).

Da MATLAB darüber hinaus mit anderen Datenbanktypen wie bspw. spaltenorientierten (Apache Cassandra) und Dokumenten- (MongoDB) sowie relationalen Datenbanken verbunden werden kann (MathWorks, 2020a), ist es möglich auf eine große Bandbreite von Datenquellen zuzugreifen.

Die Verarbeitung von großen Datensätzen kann über unterschiedliche Tools erfolgen, bspw. mittels eines kompletten Frameworks, eines Datenbanksystems und/ oder mittels Bibliotheken (Durand *et al.*, 2018). Die Verwendung und Einbindung von solchen Graph-Tools sind allerdings noch nicht ausreichend erforscht und erfolgt (Drakopoulos *et al.*, 2015).

4 Entwicklung eines Graphmining-Konzeptes in MATLAB zur Analyse von Daten aus der Supply Chain Domäne

Wie in Abschnitt 2.3.2 beschrieben, fokussieren Graphdatenbanken die Beziehungen zwischen den Daten. Übertragen auf die SC Domäne entspricht dies einer Fokussierung auf die Transaktionsdaten. Da die Transaktionsdaten zu den wichtigsten Daten in einer SC zählen (siehe Abschnitt 2.2), ist die Wahl von Graphdatenbanken als Grundlage eines Data Mining-Prozesses sinnvoll. Damit kann ein solcher Prozess als Graphmining bezeichnet werden (siehe Abschnitt 3.1). Um einen Graphmining-Prozess in MATLAB strukturiert durchführen zu können, wird ein Konzept benötigt. Ein zu diesem Zweck entwickeltes Konzept wird im Folgenden vorgestellt.

Bei der Erstellung eines Graphmining-Konzeptes für die Umsetzung in MATLAB zur Analyse von Daten aus der SC Domäne wurde sich an der allgemeinen Struktur des Data Minings, die in Abschnitt 3.1 vorgestellt wurde, orientiert. In Abbildung 2 sind die für das Data Mining benötigten Prozesskomponenten dargestellt. Diese Komponenten wurden bei der Erstellung des Konzeptes berücksichtigt. Die einzelnen Aspekte des Data Minings wurden unter Berücksichtigung der Charakteristika des Graphminings angepasst. Zusätzlich wurden bei der Entwicklung des Konzeptes unterschiedliche Kompetenzen wie kontextbezogenes Wissen, Programmierung und Datenbewertung beachtet. Bei der Nutzung von Graphmining ist zu beachten, dass das Graphmining nur als Teil des übergeordneten KDD zu verstehen ist (siehe Abschnitt 3.1) und somit nicht als alleinstehender Prozess durchgeführt werden darf.

Wie in Abbildung 3 deutlich wird, unterteilt sich das Graphmining-Konzept in vier Hauptbestandteile: die kontextbezogenen Vorüberlegungen, die Datenvorbereitung, die Ausführung des Graphmining-Algorithmus und die Bewertung der Graphmining-Algorithmus-Ergebnisse. Dabei werden im ersten Teil alle Entscheidungen getroffen, für die Domänenwissen und Wissen über die allgemeinen Zusammenhänge bzgl. der zu beantwortenden Fragestellung benötigt werden. Im zweiten Teil werden die zur Verfügung gestellten Daten auf die Verarbeitung im Graphmining-Algorithmus vorbereitet, sodass der im dritten Teil angewendete Algorithmus verwertbare Ergebnisse liefert.

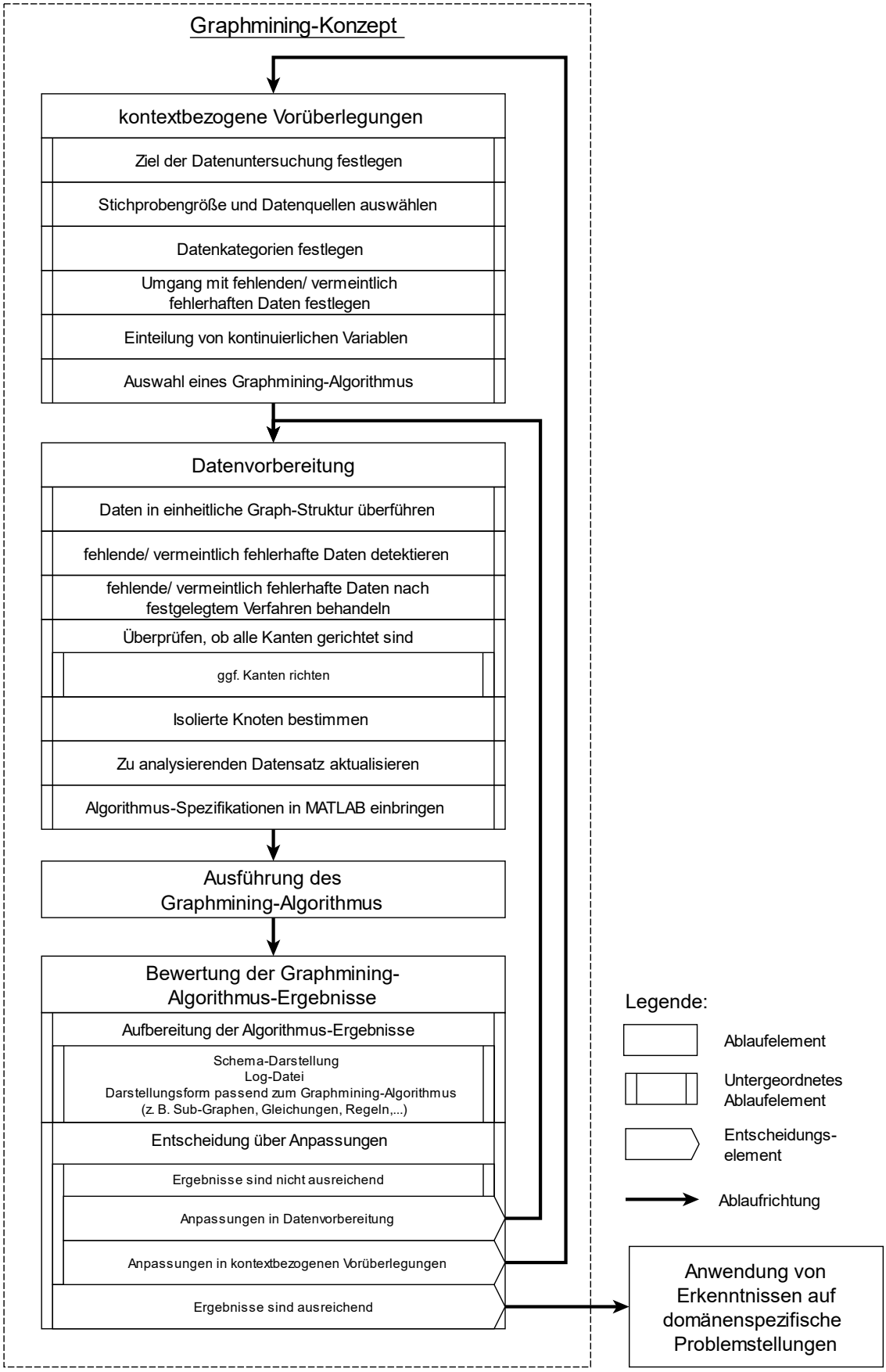

Abbildung 3: Graphmining-Konzept zur Anwendung in MATLAB. Ablaufdiagram zur Durchführung eines Graphmining-Prozesses

Diese Ergebnisse werden anschließend so aufbereitet, dass Analysten basierend darauf einschätzen können, ob die Graphmining-Algorithmus-Ergebnisse ausreichen, um das Ziel der Datenuntersuchung zu erfüllen. Es kann notwendig sein, dass entsprechend der Bewertung der Algorithmus-Ergebnisse Anpassungen an vorherigen Schritten vorgenommen werden müssen. Die Bestandteile des Konzeptes inklusive ihrer Unteraufgaben werden nacheinander den Verlaufspfeilen entsprechend durchlaufen. Dadurch ergibt sich ein iterativer Prozess, der zu einem besseren Verständnis der verwendeten Daten führt.

In den folgenden Abschnitten werden einzelne Aspekte des Konzeptes näher erläutert. Dabei wird jedoch nicht genauer auf die Ausführung von Graphmining-Algorithmen eingegangen, da das Konzept allgemein für domänenspezifische Aufgaben konzipiert ist.

4.1 Kontextbezogene Vorüberlegungen

Die kontextbezogenen Vorüberlegungen zeichnen sich dadurch aus, dass sie stark abhängig von der jeweiligen Problemstellung sind. Sie müssen somit von einer Person bzw. Personengruppe durchgeführt werden, die über domänenspezifisches Wissen verfügt. Zudem muss kontextbezogenes Wissen bzgl. der zu untersuchenden Fragestellung vorhanden sein. Aus einer Problemstellung können Fragestellungen abgeleitet werden, mit deren Hilfe die Problemstellung gelöst werden kann. Eine solche Fragestellung soll anschließend mit Hilfe des Graphminings beantwortet werden. Basierend auf der ausgewählten Fragestellung und somit dem Ziel der Datenuntersuchung können passenden Datenquellen und die zugehörige Stichprobengröße gewählt werden. Dabei können Datenquellen unabhängig von ihrer Datenstruktur in Betracht gezogen werden, da alle Daten in eine gemeinsame Graphdatenbank überführt werden. Jedoch sollten die Zusammenhänge zwischen den unterschiedlichen Daten bekannt gemacht werden. Entscheidend für die Auswahl ist die Relevanz der Daten für die jeweilige Fragenstellung. Anschließend muss eine Kategorisierung der Daten vorgenommen werden. Dies dient zum einen einem besseren Verständnis der Daten, sodass diese besser zusammengefügt werden können. Zum anderen unterstützt die Kategorisierung die Entwicklung eines Schemas, wie mit fehlenden bzw. vermeintlich fehlerhaften Daten umzugehen ist. Das Schema dient dazu, dass nur sinnstiftende Daten in den späteren Graphmining-Algorithmus eingebracht werden. Dabei ist zu beachten, dass alle Daten, die in das übergeordnete KDD (siehe Abschnitt 3.1) eingebracht werden, eine gute Datenqualität aufweisen (siehe Abschnitt 2.2). Trotz der

vorausgesetzten Datenqualität ist es möglich, dass z. B. durch das Verwenden von unterschiedlichen Datenquellen, nicht immer alle zusammengehörigen Daten vorhanden sind. Die bereits vorhandene Datenqualität stellt jedoch sicher, dass vertrauenswürdige Daten in den Graphmining-Prozess eingebracht werden. Da es sich bei einem Großteil der Daten um Transaktionsdaten handelt (siehe Abschnitt 2.2), kann die Entwicklung eines Schemas zum Umgang mit fehlenden bzw. vermeintlich fehlerhaften Daten komplex sein. Die Transaktionsdaten verbinden in der später zu untersuchenden Graphstruktur unterschiedliche Knoten, sodass Anpassungen von Transaktionsdaten direkt Einfluss auf weitere Daten haben. Zudem muss beachtet werden, dass die meisten Daten nicht als einzelnes Element in dem Graphen vorliegen. Knoten und Kanten weisen beliebig viele Eigenschaften auf (siehe Abschnitt 2.2 und Abschnitt 2.3.2). Somit wird bei der Veränderung eines einzelnen Datums auch indirekt Einfluss auf das zugehörige Objekt (Knoten bzw. Kante) genommen. Das hier festgelegte Schema zum Umgang mit fehlenden bzw. vermeintlich fehlerhaften Daten wird im weiteren Verlauf des Konzeptes in der Datenvorbereitung angewendet.

Innerhalb der Datenquellen können Daten, die kontinuierliche Daten abbilden, vorhanden sein. Kontinuierliche Daten können jedoch nur in diskretisierter Form abgespeichert werden. Da die Diskretisierung insbesondere bei unterschiedlichen Datenquellen nicht zwangsläufig aufeinander abgestimmt ist, muss ein Verfahren zum Umgang mit kontinuierlichen Variablen festgelegt werden. Auf diese Weise wird sichergestellt, dass die verwendeten Daten in einem vergleichbaren Rahmen vorliegen.

Die abschließende Entscheidung, die im Bereich der kontextbezogenen Vorüberlegungen getroffen werden muss, bezieht sich auf die Auswahl eines Graphmining-Algorithmus (siehe Abschnitt 3.2). Die Auswahl des Graphmining-Algorithmus wird maßgeblich von der gewählten Fragestellung und dem damit verbundenen Ziel der Datenuntersuchung bestimmt. Somit wird Wissen bzgl. des mit der Fragestellung untersuchten Gebiets und der Anwendbarkeit einzelner Graphmining-Algorithmen benötigt.

Sind alle Unterabschnitte der kontextbezogenen Vorüberlegungen durchlaufen, ist die durch das Graphmining zu untersuchende Fragestellung genauer beschrieben. Zudem wurden relevante Daten bestimmt und zur optimalen Verarbeitung kategorisiert. Durch die Wahl eines Graphmining-Algorithmus werden zudem Vorgaben für die weitere Verarbeitung der Daten festgelegt. Bei all diesen Überlegungen sind stets die domänenspezifischen Eigenschaften von Daten und Fragestellungen zu beachten.

4.2 Datenvorbereitung

Die Datenvorbereitung bildet den Verbindungsschritt zwischen den kontextbezogenen Vorüberlegungen und der eigentlichen Anwendung des Graphmining-Algorithmus. Das Ziel der Datenvorbereitung ist, die nach den Vorüberlegungen bereitgestellten Daten für den Algorithmus vorzubereiten. Dies bedeutet, dass die konkret durchzuführenden Schritte je nach Datenlage und Wahl des Algorithmus in den Vorüberlegungen voneinander abweichen können.

Da in dem entwickelten Konzept Graphmining durchgeführt wird, ist es notwendig, dass die zu analysierenden Daten in eine Graphstruktur überführt werden (siehe Abschnitt 3.1 und Abschnitt 3.2). Basierend auf den in den kontextbezogenen Vorüberlegungen gewählten Datenquellen kann es zu unterschiedlichen Szenarien bei der Zusammenführung von Daten kommen. Ziel der Datenzusammenführung ist es, alle Daten in einer für das Graphmining bereitgestellten Datenbank, im Folgenden als Graphmining-Datenbank bezeichnet, zusammenzufassen. Auf diese Weise wird sichergestellt, dass durch die Analyse der Daten keine anderweitig genutzte Datenbank beeinträchtigt wird. Mögliche Szenarien bei der Datenzusammenführung sind in Abbildung 4 aufgeführt. Insgesamt lassen sich fünf Szenarien unterscheiden. Dabei erfolgt eine Differenzierung bzgl. der Anzahl an Datenquellen, der Unterscheidung, ob eine einheitliche Graphstruktur vorliegt, und ob ausschließlich Graphstrukturen eingebunden werden sollen. Liegt eine Situation wie in Szenario 1 oder Szenario 3 vor, können die Daten direkt in die Graphmining-Datenbank überführt werden. Dadurch, dass alle Daten bereits in einer Graphstruktur vorliegen, sind keine weiteren Datenstruktur-Umwandlungsprozesse notwendig. Häufig liegen Daten allerdings in anderen Datenstrukturen vor. Dies liegt u. a. daran, dass bspw. relationale Datenbanken deutlich verbreiteter als Graphdatenbanken (siehe Abschnitt 2.3 und Abschnitt 2.3.1) sind. Somit kann ein zusätzlicher Zwischenschritt zur Verarbeitung von Daten, die nicht in Graphstrukturen vorliegen, notwendig sein. Der Aufwand der Umwandlung kann stark variieren, je nachdem, wie viele unterschiedliche Datenstrukturen in Graphstrukturen umgewandelt werden müssen. In den Szenarien 2 und 4 in Abbildung 4 müssen Daten mit einer gleichen Datenstruktur in die Graphstruktur überführt werden. Dabei ist bei Szenario 4 jedoch Wissen aus den kontextbezogenen Vorüberlegungen notwendig, um die Verbindung zwischen den Daten aus den unterschiedlichen Quellen korrekt übernehmen zu können. Dies gilt auch insbesondere in Szenario 5. Szenario 5 stellt den aufwendigsten Anwendungsfall dar.

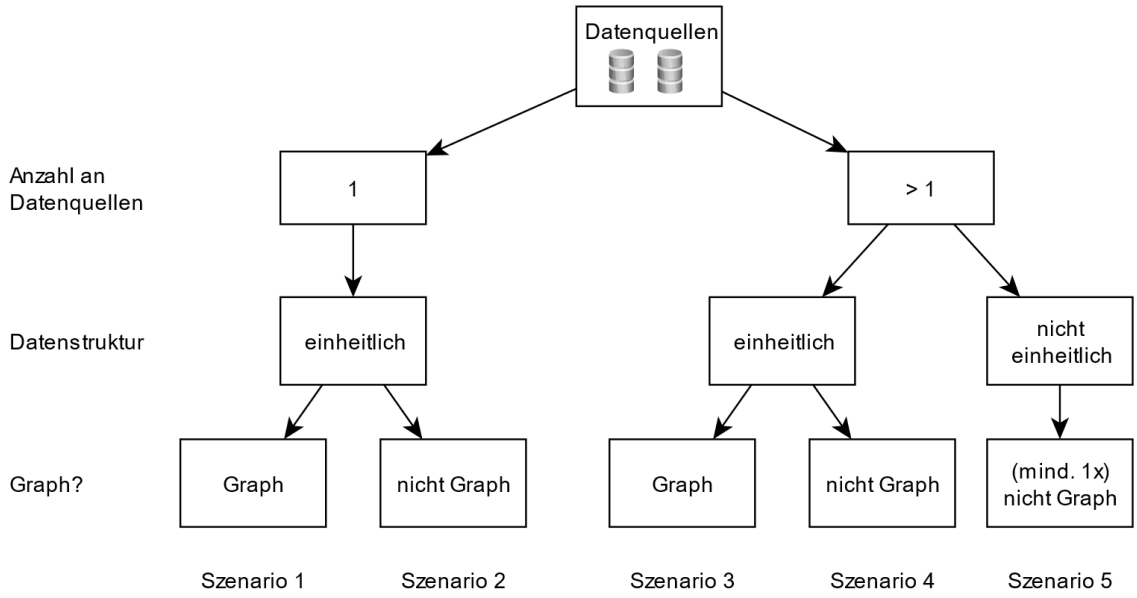

Abbildung 4: Datenstrukturkombinationen bei Datenquellen. Mögliche Szenarien von Datenstrukturkombinationen bei der Einbindung von Daten für den Graphmining-Prozess

Dies liegt daran, dass die Verbindung zwischen Daten aus unterschiedlichen Datenquellen korrekt erstellt und zusätzlich unterschiedliche Datenstrukturen in Graphstrukturen werden müssen. Je nach verwendeter Graphdatenbank, kann nicht sichergestellt werden, dass diese Umwandlungsprozesse direkt auf der Graphdatenbank durchgeführt werden können. Aus diesem Grund empfiehlt es sich die Datenstrukturumwandlung in MATLAB durchzuführen, da MATLAB mit vielfältigen Datenstrukturen arbeiten kann (siehe Abschnitt 2.2 und Abschnitt 3.2).

Sind alle Daten in eine Graphstruktur eingebunden, müssen fehlende bzw. vermeintlich fehlerhafte Daten detektiert werden. Dies ist notwendig, um Verfälschungen des Graphmining-Algorithmus-Ergebnisses zu verhindern. Wurden fehlende bzw. vermeintlich fehlerhafte Daten entdeckt, wird mit ihnen nach dem zuvor gewählten Schema verfahren (siehe Abschnitt 4.1). Auf diese Weise entsteht ein Datensatz, der komplett in einer Graphstruktur vorliegt und bereits auf die verwertbaren Daten reduziert wurde.

Anschließend wird die Graphstruktur bzgl. ihrer Kanten untersucht. Durch eine Graphstruktur wird nicht per se vorgeschrieben, dass Kanten gerichtet sind (siehe Abschnitt 2.2). Die Beziehungen in SCs zeichnen sich insbesondere bei der Betrachtung von Informations-, Waren- und Geldflüsse durch eine vorgegebene Richtung aus (siehe Abschnitt 2.1). Die Kantenüberprüfung hat somit den Grund, dass Beziehungen zwischen den Knoten

eindeutig definiert sein sollen. Am Beispiel von Warenflüssen lässt sich dies veranschaulichen. Ein Knoten, fungiert als Absender der Waren, während ein weiterer Knoten als Empfänger auftritt. Ist die Richtung dieser Transaktion nicht gegeben, kann der Warenfluss nicht korrekt abgebildet werden. Sind also Sender und Empfänger durch die Richtung der Kante nicht klar gekennzeichnet, lassen sich aus den Daten schwieriger sinnvolle Informationen ableiten. Bei der Anwendung des Konzeptes wird deshalb überprüft, ob alle Kanten gerichtet sind. Falls Daten wie zuvor beschrieben, zunächst noch in eine Graphstruktur überführt werden müssen, ist darauf zu achten, dass direkt gerichtete Graphen erzeugt werden. Für die Fälle, dass Daten aus Graphstrukturen mit einbezogen werden (siehe Szenarien 1, 3 und evtl. 5 in Abbildung 4), müssen die einzelnen Kanten überprüft werden. Falls dabei Kanten entdeckt werden, die nicht gerichtet sind, muss dies korrigiert werden.

Eine weitere Besonderheit, die in Graphen auftreten kann, sind isolierte Knoten. Solche Knoten weisen keine Kanten mit anderen Knoten auf. Unabhängig vom angewendeten Algorithmus sind dies interessante Objekte. Einerseits können sie ungenutzte Ressourcen beinhalten. Andererseits ist es aber möglich, dass es sich bei ihnen um Störgrößen handelt. Aus diesen Gründen ist es sinnvoll isolierte Knoten bereits im Vorhinein zu detektieren. Zudem kann der zu analysierende Datensatz je nach gewähltem Algorithmus um die isolierten Knoten reduziert werden.

Die beschriebenen Schritte werden alle innerhalb von MATLAB durchgeführt. Wurden die einzelnen Schritte der Datenvorbereitung bis zu diesem Punkt durchlaufen, kann der zu analysierende Datensatz in der Graphmining-Datenbank aktualisiert werden. Auf diese Weise wird sichergestellt, dass die Graphmining-Datenbank stets nur die akut relevanten Daten enthält.

Damit in der nächsten Phase des Konzeptes der Graphmining-Algorithmus durchgeführt werden kann, müssen auf den jeweiligen Algorithmus angepasste Algorithmus-Spezifikationen in MATLAB eingebracht werden. Zur Durchführung des gewählten Algorithmus können die bereits in MATLAB vordefinierten Funktionen genutzt werden. Die Funktionen können durch zusätzlichen MATLAB-Code ergänzt und an die jeweilige Fragestellung angepasst werden (siehe Abschnitt 3.2). Dabei ist auf einen modularen Aufbau der Programmsegmente zu achten, damit diese variabel für weitere Graphanalysen ange-

wendet werden können. Dadurch wird die Anpassbarkeit des Programmcodes erhöht, sodass die evtl. notwendigen Modifikationen im späteren Verlauf des Graphmining-Prozesses erleichtert werden.

Anschließend kann der zuvor ausgewählte Graphmining-Algorithmus auf die aufbereiteten Daten angewendet werden.

4.3 Bewertung der Graphmining-Algorithmus-Ergebnisse

Die Bewertung der Graphmining-Algorithmus-Ergebnisse erfolgt nach der Durchführung des eigentlichen Algorithmus (siehe Abbildung 3). Damit Analysten die Ergebnisse des Algorithmus korrekt einordnen können, ist zunächst eine Aufbereitung der Ergebnisse notwendig. Darüber hinaus sind allerdings auch Informationen über die allgemeinen Zusammenhänge der Daten sowie Informationen bzgl. der angepassten Daten notwendig. Allgemeine Informationen sind notwendig, damit Ergebnisse korrekt in den Sachbezug eingeordnet werden können. Zudem kann durch eine Kombination von allgemeinen Informationen über die verwendeten Daten, Informationen über die Anpassung der Daten und zusätzlich den Ergebnissen des Algorithmus, genauer detektiert werden, an welchen Stellen des Prozesses noch Anpassungen notwendig sind. Dabei werden alle Aufbereitungen von Ergebnissen direkt in MATLAB erzeugt. Auf diese Weise werden alle Informationen an einer Stelle gebündelt.

Als allgemeine Information über den Zusammenhang der zu analysierenden Daten wird eine Schema-Darstellung verwendet. Diese zeigt die grundsätzliche Vernetzung zwischen unterschiedlichen Datengruppen auf. In Abbildung 5 ist eine beispielhafte Darstellung aufgeführt. In dem dargestellten Fall liegen vier unterschiedliche Knotentypen (blau, rot, grün und gelb) und drei unterschiedliche Kantentypen vor. Die unterschiedlichen Knotentypen weisen prinzipiell immer die dargestellten Verbindungen über die gerichteten Kanten (Typ 1-3) untereinander auf. Auf diese Weise können generelle Zusammenhänge ohne konkrete Daten ausgedrückt werden. Im Rahmen einer SC könnten die drei Kantentypen Waren-, Geld- und Informationsflüsse darstellen (siehe Abschnitt 2.1). Die unterschiedlichen Knotentypen könnten Lieferanten, Hersteller, Verkäufer und Kunden repräsentieren.

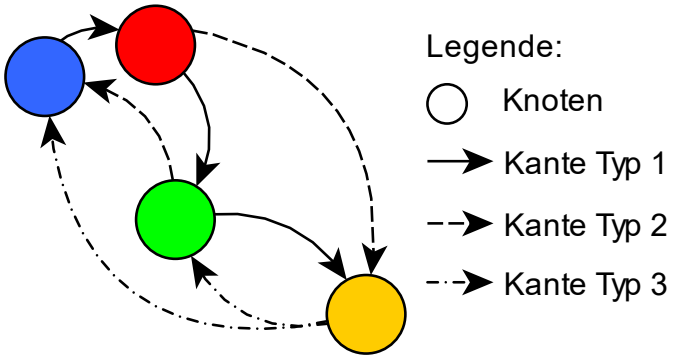

Abbildung 5: Beispielhafte Schema-Darstellung. Grundlegende Vorschrift der Verknüpfung zwischen unterschiedlichen Kanten- und Knotentypen

Zusätzlich werden Anpassungen an den Daten in einer Log-Datei zusammengefasst. Wie in Abbildung 6 gezeigt ist, werden in einer solchen Log-Datei die Ausgangswerte und die veränderten Werte von angepassten Daten aufgeführt, sowie der Grund der Anpassung. Dies ermöglicht es, die Auswirkungen von angepassten Daten auf das Gesamtergebnis besser abzuschätzen. Genauso werden ergänzte Daten und exkludierte Daten aufgeführt. Auf diese Weise können alle Anpassungen am Datensatz an einer Stelle nachverfolgt werden. Ebenfalls werden die isolierten Knoten in die Log-Datei eingetragen, da diese stets als potenziell interessante Objekte anzusehen sind. Um besser einschätzen zu können, welcher Anteil an Daten manipuliert oder ausgeschlossen wurden, wird zudem die ursprüngliche Gesamtzahl an Knoten und Kanten angegeben.

Die Darstellungsform der Graphmining-Algorithmus-Ergebnisse kann nicht pauschal festgelegt werden, da sie stark vom verwendeten Algorithmus abhängt. Mögliche Darstellungen umfassen bspw. Subgraphen, Gleichungen, Tabellen und Regeln. Diese Arten der Visualisierung bieten zusammen mit den zuvor beschriebenen Datenaufbereitungen eine Entscheidungsunterstützung, ob die bisher durchgeführten Graphmining-Schritte ausreichend zur Beantwortung der Fragestellung sind. Allgemein gilt es zu beachten, dass unterschiedliche Darstellungsformen unterschiedliche Fokusse setzen. Deshalb kann es sinnvoll sein, die Ergebnisse des Graphmining-Algorithmus auf mehrere Weisen zu visualisieren. MATLAB bietet dabei eine große Bandbreite an Visualisierungs- und Exportmöglichkeiten. Durch die gemeinsame Berücksichtigung der unterschiedlichen Visualisierungen können bisher noch nicht bekannte Zusammenhänge unter den Daten aufgedeckt werden.

```
Log Datei

Projekt: XXX

Durchlaufnr.: XX

Datum:      XX.XX.XXXX
Uhrzeit:    XX.XX.XXXX

Anzahl Knoten: XXX
Anzahl Kanten: XXX

Veraenderte Daten
ID          ursprungsWert       angepassterWert     Grund
...         ...                 ...                 ...
...         ...                 ...                 ...
...         ...                 ...                 ...
...         ...                 ...                 ...
...         ...                 ...                 ...
...         ...                 ...                 ...
...         ...                 ...                 ...
...         ...                 ...                 ...

Ergaenzte Daten
ID          Wert
...         ...
...         ...
...         ...

Exkludierte Daten
ID          Grund
...         ...
...         ...
...         ...
...         ...

Isolierte Knoten
ID          Typ
...         ...
...         ...
...         ...

Legende: Grund
     1       1       1
Grund1  Grund2  Grund3
```

Abbildung 6: Schematische Log-Datei. Zusammenfassende Darstellung über allgemeine Informationen, Änderungen und Anpassungen von Daten während des Graphmining-Prozesses

Häufig sind nach dem ersten Durchlauf des Graphminings Anpassungen am Graphmining-Prozess notwendig, um das Ziel der Datenanalyse zu erreichen. Dies wird in Abbildung 3 durch die sich rückbeziehenden Pfeile aus den Anpassungsblöcken deutlich. Dabei sind direkte Anpassungen in den ersten zwei Hauptblöcken des Graphmining-Konzeptes möglich.

Zum einen können Anpassungen an den kontextbezogenen Vorüberlegungen vorgenommen werden. Dabei ist besonders die Klassifizierung der Daten und das darauf aufbauende Schema bzgl. des Umgangs mit fehlenden bzw. vermeintlich fehlerhaften Daten kritisch zu hinterfragen. Die hier festgelegte Klassifizierung hat Einfluss auf den gesamten Graphmining-Prozess. Da an dieser Stelle Entscheidungen insbesondere auf Grundlage

von Domänenverständnis und Erfahrung getroffen werden, findet sich in diesem Bereich großes Optimierungspotential. Dies gilt insbesondere unter der Berücksichtigung der bereits gewonnen Erkenntnisse aus mind. einem bereits durchgeführten Durchlauf des (noch nicht abgeschlossenen) Graphmining-Prozesses. Eine weitere Anpassungsmöglichkeit liegt in der Wahl des Graphmining-Algorithmus selbst. Unterschiedliche Algorithmen weisen unterschiedliche Stärken und Schwächen auf, sodass es zur Beantwortung einer Frage notwendig sein kann, dass mehrere Algorithmen angewendet werden, sodass sich deren Stärken ergänzen. Die Zielformulierung bzw. die zu beantwortende Fragestellung ist ein zusätzlicher Punkt, der zu hinterfragen ist. So kann es notwendig sein, die Frage bzw. die Zielformulierung präziser zu definieren. Dies kann dazu führen, dass die ursprüngliche Fragestellung in mehrere Subfragestellungen unterteilt werden muss, um auf einzelne Aspekte Antworten zu liefern. Damit einhergehend kann es notwendig sein, die zu betrachtenden Datenquellen oder Stichproben anzupassen. Werden in den kontextbezogenen Vorüberlegungen Anpassungen vorgenommen, hat das direkten Einfluss auf alle weiteren folgenden Schritte.

Die zweite Stelle, an der Anpassungen vorgenommen werden können, ist die Datenvorbereitung. Da die Vorgaben, wie Daten vorzubereiten sind, zum großen Teil aus den kontextbezogenen Vorüberlegungen stammen, ist in diesem Block insbesondere die Parametrisierung der Graphmining-Algorithmen zu betrachten. Bspw. kann es notwendig sein, Grenzwerte zu verändern oder andere Parametrisierungen vorzunehmen. Um dabei ein effizientes Vorgehen zu unterstützen, ist der modulare Aufbau der Programmstruktur wichtig.

Werden die gewonnen Erkenntnisse auf Basis der Graphmining-Ergebnisse nach Durchführung des iterativen Vorgehens als zufriedenstellend eingestuft, können die gewonnen Erkenntnisse auf Daten außerhalb des KDD- und somit auch außerhalb des Graphmining-Prozesses angewendet werden. Auf die Weise können die Erkenntnisse aus dem Graphmining-Prozess auch als Unterstützung für Entscheidungen innerhalb des SCM genutzt werden.

Allgemein lässt sich festhalten, dass insbesondere im Bereich der Bewertung der Ergebnisse der Graphmining-Algorithmen deutlich wird, dass kein Schritt im Graphmining-Prozess gänzlich unabhängig von den anderen Schritten ist. Die kontextbezogenen Vorüberlegungen beeinflussen als Art Weichenstellung den gesamten Prozess. Dies spiegelt

sich bspw. im festgelegten Umgang mit fehlenden Daten wider. Die Definition der Vorgehensweise erfolgt im Vorfeld und wird später in der Datenvorbereitung umgesetzt. Die Wahl des Graphmining-Algorithmus in den kontextbezogenen Vorüberlegungen beeinflusst u. a. die Implementierung in den MATLAB-Programmcode und im späteren Verlauf die Darstellung der Ergebnisse. Die Wahl des Algorithmus bestimmt somit, auf welche Aspekte der Daten die Fokussierung liegt. Aufgrund dieser engen Verbindung aller Teilaspekte des Konzeptes, ist die Durchführung des Graphminings in Form eines iterativen Prozesses von großer Bedeutung.

5 Prototypische Anwendung des Graphmining-Konzeptes

Im folgenden Kapitel wird das in Kapitel 4 erarbeitete Graphmining-Konzept prototypisch auf Teile des NorthWind-Datensatzes (GitHub, 2015) angewendet. Dabei soll die Praktikabilität des Konzeptes überprüft werden, sodass Rückschlüsse auf dessen Stärken und Schwächen gezogen werden können. Zudem wird insbesondere auf die Umsetzung in MATLAB eingegangen. Bei der Umsetzung werden MATLAB Version 2020a (MathWorks Deutschland, 2021) und Neo4j Desktop 1.3.4 sowie eine Neo4j Datenbank mit der Version 3.5.17 (Neo4j Graph Database Platform, 2020) verwendet.

5.1 Vorstellung des Fallbeispiels

Der prototypischen Anwendung des Konzeptes liegt ein Teil des NorthWind-Datensatzes zugrunde. Der NorthWind-Datensatz beschreibt ein Produkt-Verkaufs-System und besteht aus mehreren Comma-separated-values-Dateien (.csv-Dateien). Bei den für diese Arbeit ausgewählten Teilen des Datensatzes handelt es sich um Daten bzgl. der Produkte (z. B. Produkt-Kategorie, Produktname und Stückpreis), der Lieferanten (z. B. Firmenname und Adresse), der Angestellten (z. B. Name und Stelleneinordnung) und der Aufträge (z. B Auftragsnummer, Preis, Menge und Lieferdatum) (siehe Abbildung 7).

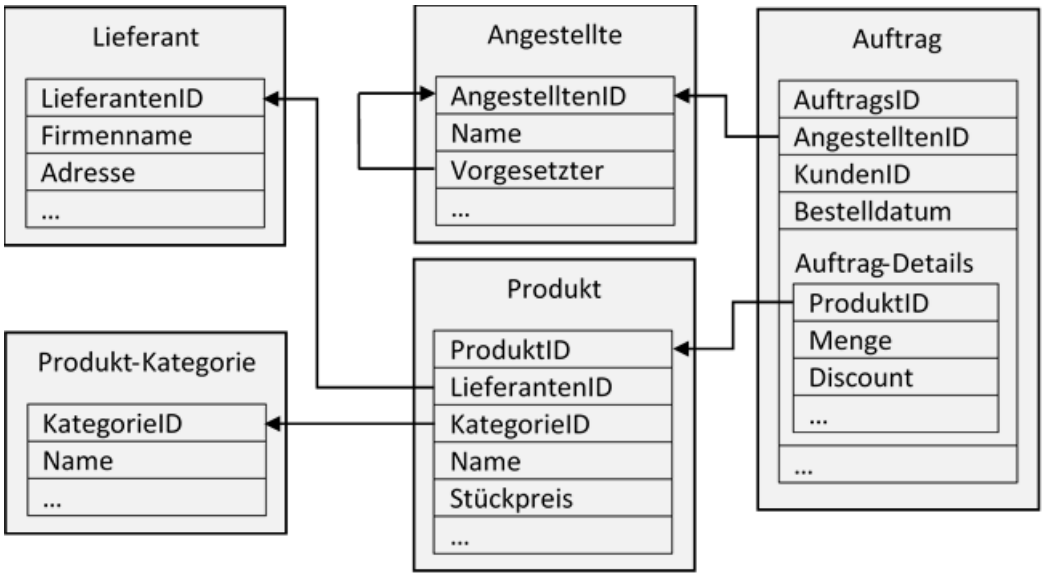

Abbildung 7: Struktur der NorthWind .csv-Dateien. Darstellung der Zusammenhänge der unterschiedlichen Tabellen mit NorthWind-Daten

Diese Auswahl an Daten ermöglicht es einen Ausschnitt einer SC darzustellen. Zudem wird durch das Schaubild in Abbildung 7 der netzartige Charakter des Produkt-Verkaufs-Systems deutlich. Bspw. beschreibt ein Teil der Auftrags-Tabelle, der in Abbildung 7 als Auftrag-Details bezeichnet wird, ausschließlich die Verbindungen zwischen einem Auftrag und den zugehörigen Produkten. Aus diesem Grund bietet sich eine Überführung der Daten in eine Graphstruktur an, um so die Daten effektiver analysieren zu können.

5.2 Anwendung

Im Folgenden wird das in Kapitel 4 vorgestellte Graphmining-Konzept auf die reduzierten NorthWind-Daten angewendet. Dabei werden die Hauptphasen 1) kontextbezogene Vorüberlegungen, 2) Datenvorbereitung, 3) Ausführung des Graphmining-Algorithmus und 4) Bewertung der Graphmining-Algorithmus-Ergebnisse durchlaufen (siehe Abbildung 3).

Kontextbezogene Vorüberlegungen:

Zunächst müssen die kontextbezogenen Vorüberlegungen getroffen werden. Hierfür muss das Ziel der Datenuntersuchung festgelegt werden. In dieser prototypischen Anwendung des Konzeptes soll untersucht werden, welche Angestellten für die Generierung des meisten Umsatzes verantwortlich sind. Um diese Frage zu beantworten werden Daten des NorthWind-Datensatzes ausgewählt (siehe Abschnitt 5.1).

Aus der Betrachtung von Abbildung 7 geht hervor, dass unterschiedliche Datenkategorisierungen vorgenommen werden können. Grob lässt sich zwischen Objekten, den späteren Knoten, und Verbindungen, den späteren Kanten unterscheiden. Die Objekttypen lassen sich dadurch als *Lieferant*, *Produkt*, *Angestellte* und *Auftrag* (siehe Abbildung 7) definieren. Die Zeile einer Tabelle bildet ein Objekt. Verweise auf andere Tabellen werden den Verbindungen zugeordnet. Zusätzlich ergänzt der Tabellenabschnitt zu den Auftrag-Details die Verbindung zwischen Aufträgen und Produkten. Die sich daraus ergebenden Verbindungstypen sind X *beinhaltet* Y, X *liefert* Y, X ist *Teil_von* Kategorie Y, X *verkauft* Y und X ist Y *unterstellt*.

Um die Fragestellung zu beantworten, werden primär die Daten bzgl. der Angestellten, der Aufträge und der Produkte benötigt. Es werden jedoch nicht alle Daten bzgl. eines

Objektes bzw. einer Verbindung benötigt. Als erste Überlegung wird deshalb angenommen, dass die *AngestelltenID* und die *AuftragsID* mit den Verbindungen zu den jeweiligen Produkten unverzichtbar sind. Bei den Verbindungen zwischen den Aufträgen und den Produkten sind Daten bzgl. der Menge, des Stückpreises und des zugesprochenen Nachlasses unverzichtbar. Dies liegt darin begründet, dass ohne diese Daten der Umsatz nicht zu berechnen ist.

Im ersten Durchlauf des Graphmining-Prozesses wird davon ausgegangen, dass keine fehlerhaften Daten vorliegen. Diese Einschätzung kann nach der Bewertung der Graphmining-Algorithmus-Ergebnisse überarbeitet werden. Dadurch entfällt im späteren Verlauf in der Datenvorbereitung die Suche nach fehlerhaften Daten. Fehlen Daten, die nach ersten Überlegungen nicht zwangsläufig notwendig zur Beantwortung der Frage sind, wird das zugehörige Objekt bzw. die zugehörige Verbindung trotzdem in die Analyse mit einbezogen. Fehlen notwendige Daten, wird der entsprechende Datensatz in Form eines Knotens oder einer Kante nicht berücksichtigt.

Da in diesem Fallbeispiel keine kontinuierlichen Daten vorliegen, kann der Schritt der Einteilung von kontinuierlichen Variablen übersprungen werden.

Wie in der Festlegung der Datenkategorien beschrieben, werden durch die vorgegebene Fragestellung insbesondere die Verbindungen zwischen Angestellten, Aufträgen und Produkten untersucht. Aus diesem Grund empfiehlt sich die Anwendung einer Nachbarschaftssuche.

Datenvorbereitung:

Sind alle kontextbezogenen Vorüberlegungen festgelegt, kann in die Datenvorbereitung übergegangen werden. Zur Umsetzung wird MATLAB in Kombination mit Neo4j genutzt. Eine Kombination aus diesen beiden Programmen wurde gewählt, da MATLAB eine direkte Verbindung zu einer Neo4j Datenbank herstellen kann (siehe Abschnitt 3.2). Zunächst wird eine Datenbank in Neo4j angelegt, die im weiteren Verlauf mit zur Datenanalyse genutzt wird. Die Datenbank enthält zunächst keine Daten. Diese Datenbank wird im Folgenden als NorthWindDB bezeichnet. Als nächstes wird in MATLAB ein neues Programm erstellt, in dem der Programmcode für das Graphmining verfasst wird. Dieses Hauptprogramm wird als Graphmining-Code bezeichnet.

Zunächst müssen alle Daten in eine einheitliche Graphstruktur überführt werden. Es liegt ein Fall wie in Szenario 2 in Abbildung 4 vor. Es wird also eine einzelne Datenquelle

betrachtet, deren Datenstruktur kein Graph ist. Die NorthWind-Daten liegen in .csv-Dateien, also in relationalen Strukturen, vor. Um diese in eine Graphstruktur zu überführen, werden die Datenkategorien aus den kontextbezogenen Vorüberlegungen genutzt, sowie das Wissen über die verwendete Datenquelle (siehe Abbildung 7). Zudem ist es notwendig, über MATLAB eine Verbindung zu der NorthWindDB herzustellen. Damit die Verbindung aufgebaut werden kann, ist es notwendig, dass die NorthWindDB aktiv ist.

Über die Verwendung der Query-Language Cypher werden die .csv-Dateien geladen und in eine Graphstruktur überführt. Es empfiehlt sich, die .csv-Dateien in dem Import-Ordner der NorthWindDB zu speichern. Dies erleichtert den Zugriff auf die Dateien. Bei der Überführung der Daten in eine Graphstruktur müssen die Verknüpfungen, aus denen die Kanten generiert werden und die Objekte, aus denen die Knoten entstehen, genau definiert werden. Die Cypher-Anweisungen werden über MATLAB an die NorthWindDB übergeben. Dazu wird eine Query-Anweisung in einer Text-Datei geschrieben. Da mit MATLAB nur einzelne Anweisungen an die NorthWindDB übergeben werden können, werden die Query-Anweisung mittels eines Separators getrennt. Dafür wurde die Zeichenkombination „--“ gewählt. Wird die Query-Anweisung in MATLAB eingelesen, können die durch den Separator getrennten Anfragen in ein Array (siehe Abschnitt 2.2) abgelegt werden, sodass ein Eintrag im Array einer Anweisung entspricht. Auf die Weise können die Anweisungen nacheinander vom Graphmining-Code aufgerufen werden. Bei der Umwandlung der Datenstruktur in Graphstrukturen, wird darauf geachtet, dass alle Kanten gerichtet sind. Die NorthWindDB besteht nun aus den Knotentypen *Lieferant*, *Produkt*, *ProduktKategorie*, *Auftrag* und *Angestellte* sowie den Kantentypen *untersteht*, *verkauft*, *beinhaltet*, *Teil_von* und *wird_geliefert_von*. Sind die Daten in die NorthWindDB überführt, können die in eine Graphstruktur überführten Daten genauer untersucht werden.

Zunächst werden fehlende Daten detektiert. Dabei erfolgt erneut ein Rückbezug auf die kontextbezogenen Vorüberlegungen. Da davon ausgegangen wird, dass keine fehlerhaften Daten vorliegen, kann dieser Aspekt vernachlässigt werden. Aufgrund des gewählten Umgangs mit fehlenden Daten, müssen unterschiedliche Knoten- und Kantentypen überprüft werden. Die als unverzichtbar eingestuften Daten sind in den Knotentypen *Auftrag* und *Angestellte* sowie in den Kanten *verkauft* und *beinhaltet* enthalten. Geht von einem *Angestellte*-Knoten keine *verkauft*-Kante aus, wird dieser Knoten von der Analyse ausgeschlossen. Ebenso wird mit *Auftrag*-Knoten verfahren, falls diese keine *beinhaltet*-

Kante oder eine eintreffende *verkauft*-Kante aufweisen. Ebenfalls werden *beinhaltet*-Kanten ausgeschlossen, wenn sie keine Daten zu Menge, Stückpreis oder Nachlass beinhalten. Dafür werden alle relevanten Knoten und Kanten im Graphmining-Code auf die beschriebenen Eigenschaften überprüft.

Die Knoten- bzw. Kanten-IDs, der für die Analyse ausgeschlossenen Knoten und Kanten, werden zusammen mit dem Grund des Ausschlusses in den Arrays *ausgeschlossene_Knoten* und *ausgeschlossene_Kanten* gespeichert. Zur einfacheren Handhabung der Arrays wird der Grund in kodierter Form und nicht als ausgeschriebener Ausdruck abgespeichert.

Ausgeschlossene Kanten und Knoten müssen anschließend nach dem vorgegebenen Schema aus dem Graphen entfernt werden. Dafür werden die betroffenen Knoten und Kanten mittels MATLAB aus der NorthWindDB gelöscht. Falls ein Knoten entfernt werden muss, werden gleichzeitig auch die zum Knoten zugehörigen Kanten gelöscht. Dies hat den Grund, dass Kanten ohne Start- bzw. Endpunkt nicht für die Analyse verwendbar sind. Die dabei betroffenen Kanten werden dem *ausgeschlossene_Kanten*-Array hinzugefügt.

Basierend auf dem Konzept muss nun bei den Kanten des Graphen überprüft werden, ob diese gerichtet sind. Der zu analysierende Graph wurde zuvor auf der Basis von Daten, die in relationalen Strukturen vorlagen, mit gerichteten Kanten mit dem Graphmining-Code erstellt. Aus diesem Grund muss nicht zusätzlich überprüft werden, ob alle Kanten gerichtet sind. Dementsprechend kann dieser Schritt übersprungen werden. Um isolierte Knoten zu detektieren, werden alle Knoten auf eintreffende und herauslaufende Kanten überprüft. Weist ein Knoten weder eine eintreffende noch eine herauslaufende Kante auf, wird die Knoten-ID in dem Array *isolierte_Knoten* abgespeichert. Zudem wird der Knotentyp des isolierten Knotens für die spätere Erstellung der Log-Datei abgespeichert. Die Typenbezeichnung erleichtert eine Interpretation des isolierten Knotens. Da die bis zu diesem Schritt durchgeführte Datenvorbereitung zu einer Veränderung der Daten führen kann, ist es notwendig, die Daten neu aus der NorthWindDB zu laden. So wird sichergestellt, dass nicht bereits exkludierte Daten in die Analyse mit einfließen.

Als abschließender Schritt der Datenvorbereitung werden Algorithmus-Spezifikationen in Graphmining-Code vorgenommen. Um einen modularen Programmaufbau zu gewährleisten, wird der Algorithmus in eine Funktion ausgelagert, die bei der Ausführung des Graphmining-Algorithmus nur noch aufgerufen werden muss. Mit Hilfe des Algorithmus soll ermittelt werden, welcher Angestellte wie viel Umsatz generiert hat. Dafür werden

die jeweiligen Verkäufe der Angestellten ermittelt. Um dies programmiertechnisch umzusetzen, ist es notwendig, klar zu definieren, welche Pfade zu ermitteln sind und wie mit den gewonnenen Daten umgegangen werden soll. In dieser prototypischen Umsetzung werden zunächst alle Angestellten ermittelt. Zu jedem Angestellten werden anschließend die zugeordneten verkauften Aufträge bestimmt. Von einem *Auftrag*-Knoten gehen *beinhaltet*-Kanten aus, die den Aufträgen die beinhalteten Produkte zuordnen. Diese Kanten werden ausgelesen. Aus den darin beinhalteten Daten bzgl. Menge, Stückpreis und Nachlass wird der Umsatz des jeweiligen Angestellten bestimmt. Dabei sind die *Produkt*-Knoten nicht von Bedeutung, da alle benötigten Daten bereits in der *beinhaltet*-Kante enthalten sind.

Bei der Durchführung der einzelnen Schritte ist insbesondere auf die verwendeten Datentypen und -strukturen zu achten. Durch die Vernetzung von MATLAB und Neo4j liegen Daten teilweise nicht direkt in den benötigten Datentypen bzw. -strukturen vor. Aus diesem Grund müssen Daten innerhalb des Graphmining-Codes in passende Strukturen überführt werden. Dies ist durch in MATLAB gegebene Umwandlungsfunktionen gut realisierbar.

Ausführung des Graphmining-Algorithmus:

Wenn die Umsetzung des Algorithmus im Graphmining-Code definiert wurde, kann der Algorithmus durchgeführt werden.

Bewertung der Graphmining-Algorithmus-Ergebnisse:

Basierend auf dem verwendeten Graphmining-Konzept, werden zur Bewertung der Graphmining-Algorithmus-Ergebnisse zusätzlich Darstellungen benötigt. Zur allgemeinen Einordnung unterschiedlicher Daten werden die Schema-Darstellung der analysierten Graphstruktur sowie eine Log-Datei generiert, die die Anpassungen an den Daten dokumentiert.

Die Schema-Darstellung wird im Graphmining-Code mit Hilfe eines gerichteten Graphens in MATLAB dargestellt. Die dafür benötigten Daten werden aus dem Ergebnis der Abfrage des Graphen-Schemas der NorthWindDB extrahiert. Dafür werden innerhalb des Graphmining-Codes Knoten- und Kanteninformationen zusammengeführt. So kann ein gerichteter Graph erstellt und zur späteren Nutzung abgespeichert werden.

Die Log-Datei kann mit Hilfe der Arrays *ausgeschlossene_Knoten*, *ausgeschlossene_Kanten*, *isolierte_Knoten* erstellt werden. Zusätzlich verwendete Informationen betreffen die Durchlaufnummer, die Anzahl von eingebundenen Knoten und Kanten sowie eine Legende zur Dekodierung der Ausschlussgründe. Dabei werden die relevanten Daten von MATLAB in eine Text-Datei geschrieben. Das grundsätzliche Format der Log-Datei wird von der Funktion *write_Log* vorgeschrieben. Auf diese Weise werden bei jedem Graphmining-Prozess vergleichbare Log-Dateien erzeugt. Zudem wird der Arbeitsaufwand zur Erstellung einer Log-Datei minimiert.

Für die Darstellung der Algorithmus-Ergebnisse bietet sich in diesem Fall eine Tabelle an. In dieser werden die *AngestelltenID*, die Anzahl der verkauften Aufträge sowie der dadurch generierte Umsatz aufgeführt. Da die Frage beantwortet werden soll, welcher Angestellte für die Generierung des meisten Umsatzes verantwortlich ist, empfiehlt es sich zusätzlich die Tabelle zu sortieren. Des Weiteren wird ein Tortendiagramm erstellt, das die Verhältnisse zwischen den Umsätzen der einzelnen Angestellten klarer verdeutlicht.

In den Abbildungen 8, 9, 10 und 11 sind die aufbereiteten Graphmining-Algorithmus-Ergebnisse dargestellt. Die Schema-Darstellung in Abbildung 8 verdeutlicht die Zusammenhänge zwischen den unterschiedlichen Knoten- und Kantentypen. Es wird deutlich, dass zur Beantwortung der Fragestellung nur ein Subgraph verwendet wurde. Dieser besteht aus den Knoten *Angestellte*, *Auftrag* und *Produkt* sowie den Kanten *verkauft* und *beinhaltet*.

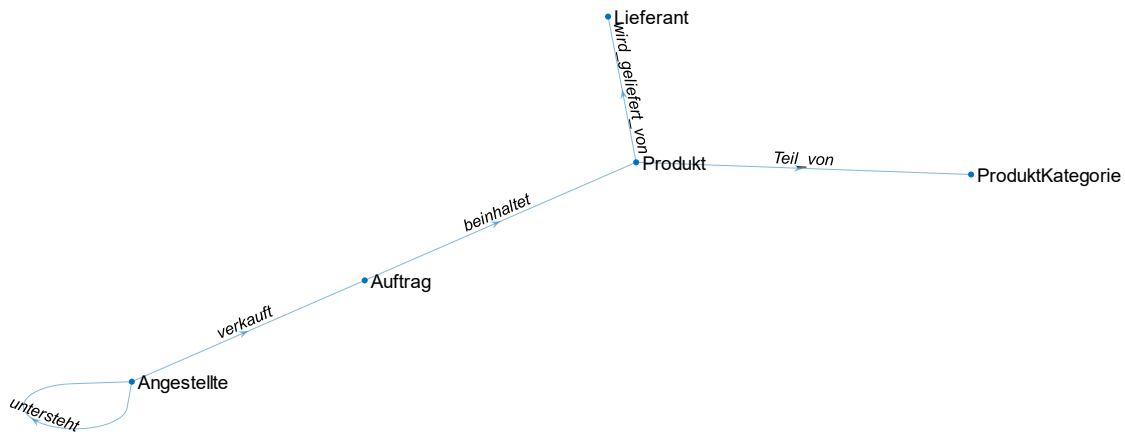

Abbildung 8: Schema-Darstellung. Darstellung der allgemeinen Zusammenhänge der NorthWind Daten

	AngestelltenID	Anzahl_Auftraege	Umsatz
1	4	420	3.0549e+06
2	1	345	2.9920e+06
3	3	321	2.4168e+06
4	2	241	1.8617e+06
5	7	176	1.4812e+06
6	8	260	1.3814e+06
7	5	117	9.9261e+05
8	9	107	8.8441e+05
9	6	168	7.3853e+05

Abbildung 9: Umsatz-Tabelle. Nach Umsatz sortierte Tabelle zur Darstellung der Umsätze der einzelnen Angestellten

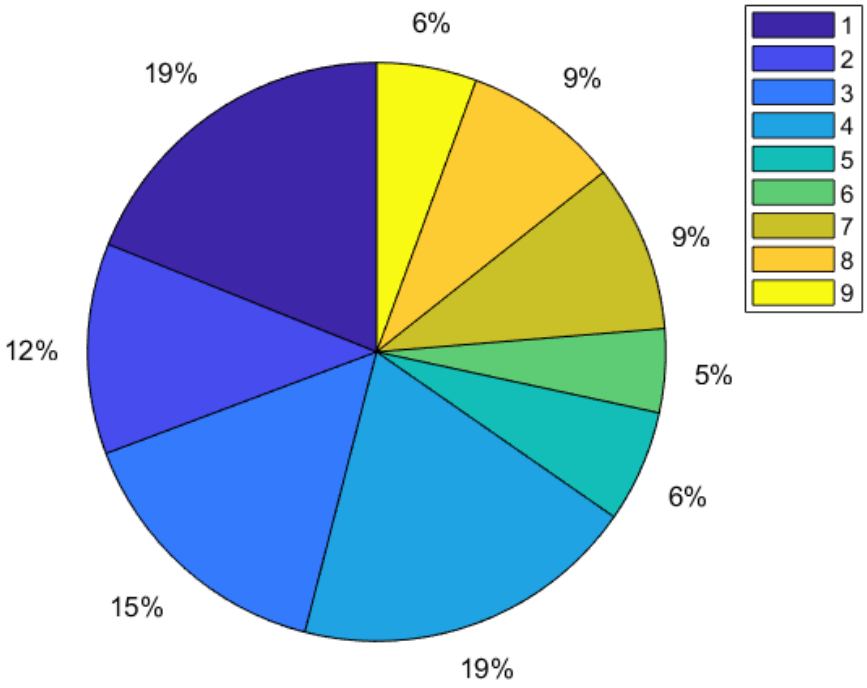

Abbildung 10: Umsatz-Tortendiagramm. Visualisierung der Umsatzanteil der unterschiedlichen Angestellten

Die Graphmining-Algorithmus-Ergebnisse in Form der Tabelle und des Tortendiagramms (siehe Abbildung 9 und Abbildung 10) legen nahe, dass der Angestellte 4 den meisten Umsatz generiert. Bei Betrachtung der Log-Datei (Abbildung 11) wird ersichtlich, dass keine Angestellten von der Analyse ausgeschlossen wurden. Somit wurden alle Angestellte in der Analyse berücksichtigt. Es zeigen sich große Unterschiede in den Umsätzen. So generiert der Angestellte 4 etwa den vierfachen Umsatz in Vergleich zu dem Angestellten 6. Bei Betrachtung der Schema-Darstellung (Abbildung 8) wird deutlich, dass auch innerhalb der Angestellten Abhängigkeiten bestehen. Dies wird durch die Beziehung *untersteht* deutlich.

```
Log Datei

Durchlaufnr.:  1

Projekt:  NorthWind Umsatz

Datum:  25-Mar-2021
Uhrzeit:  15 : 10

Anzahl Knoten:        2278
Anzahl Kanten:        9376

Veraenderte Daten
ID     ursprungsWert   angepassterWert        Grund

Ergaenzte Daten
ID       Wert

Exkludierte Daten
ID      Grund

Isolierte Knoten
ID        Typ

Legende: Grund
                    1               1                 1                              1                         1                          1
  Mengenangabe fehlt  Nachlass fehlt  Stueckpreis fehlt  Angestellte: keine Auftraege   Auftrag: keine Produkte   Auftrag: nicht verkauft
```

Abbildung 11: Log-Datei. Darstellung der Log-Datei für den Graphmining-Durchlauf 1

Durch die Entdeckung dieses Zusammenhangs kann nicht ausgeschlossen werden, dass Angestellte nur indirekt als Vorgesetzte an Verkäufen beteiligt sind. Da dieser Zusammenhang noch nicht berücksichtigt wurde, kann die zugrunde liegende Fragestellung noch nicht vollständig beantwortet werden. Es lässt sich allerdings schon als Teilergebnis formulieren, dass der Angestellte 4 den höchsten direkten Umsatz generiert.

Iteration:

Da durch die Ergebnisse des ersten Graphmining-Durchlaufes die Fragestellung noch nicht zufriedenstellend beantwortet werden kann, muss ein zweiter Graphmining-Durchlauf umgesetzt werden. Dabei wird nun auch berücksichtigt, dass ein Angestellter als Vorgesetzter indirekt an Umsätzen beteiligt sein kann. Wird diese Überlegung weitergeführt, wird deutlich, dass ein Angestellter gleichzeitig die Position eines Unterstellten und eines Vorgesetzten einnehmen kann. Mit diesen Überlegungen muss der Umgang mit fehlenden Daten überarbeitet werden. Als Ausschlusskriterium für die Analyse entfällt die Bedingung, dass von einem *Angestellte*-Knoten eine *verkauft*-Kante ausgehen muss. Als Anpassung werden nun auch die internen Personalhierarchien berücksichtigt. Ein *Angestellte*-Knoten wird in diesem Durchlauf nur dann von der Analyse ausgeschlossen, wenn weder eine *verkauft*-Kante von ihm ausgeht noch eine *untersteht*-Kante eintrifft. Eine weitere Anpassung erfolgt in der Datenvorbereitung. Dort werden die Algorithmus-Spezifikationen angepasst. Bei der Berechnung der Gesamt-Umsätze müssen die indirekten Umsätze der unterstellten Angestellten berücksichtigt werden. Dafür werden drei weitere Spalten in der tabellarischen Darstellung der Graphmining-Algorithmus-Ergebnisse zur Verfügung gestellt (Anzahl von indirekt betreuten Aufträgen; durch die indirekten Aufträge generierter Umsatz; Gesamtumsatz). Zusätzlich haben die neu getroffenen Annahmen Einfluss auf die zur Verfügung gestellten Visualisierungen. Zunächst werden in der tabellarischen Darstellung der Ergebnisse die indirekten Umsatzgrößen mit aufgeführt. Da bei der Berechnung der indirekten Umsatzgrößen die Personalstruktur berücksichtigt wurde, muss sich dies auch in der Visualisierung der Ergebnisse wiederfinden. Zur Veranschaulichung der Zusammenhänge wird eine zusätzliche Grafik erzeugt, die die hierarchischen Strukturen unter den Angestellten in Form eines Graphen abbildet.

Werden diese Anpassungen im Graphmining-Code vorgenommen, kann der zweite Durchlauf gestartet werden. Zur Bewertung der Ergebnisse werden nun eine Log-Datei (Abbildung 14), eine Schema-Darstellung, eine angepasste tabellarische Aufstellung der

Umsätze (Abbildung 12), ein Tortendiagramm sowie eine Darstellung der Personalstruktur (Abbildung 13) erzeugt. Das Tortendiagramm bzgl. der prozentualen Beteilung der Angestellten am Gesamtumsatz durch ihre direkten Umsätze ändert sich ebenso wie die Schema-Darstellung im Vergleich zum ersten Durchlauf (siehe Abbildung 8 und Abbildung 10) nicht.

Die Veränderungen in der Log-Datei (Abbildung 14) sind minimal. Es ändert sich die Durchlaufnummer, die Zeit der Durchführung sowie die Legende zum Grund des Ausschlusses von Knoten bzw. Kanten. Wie in Durchlauf 1 werden keine Knoten bzw. Kanten von der Analyse ausgeschlossen. Es werden auch in diesem Durchlauf keine isolierten Knoten identifiziert.

	1	2	3	4	5	6
	AngestelltenID	Anzahl_Auftraege	Umsatz_direkt	Anzahl_indirekte_Auftraege	Umsatz_indirekt	Gesamtumsatz
1	2	241	1.8617e+06	1914	4.0967e+06	5.9585e+06
2	5	117	9.9261e+05	451	3.1041e+06	4.0967e+06
3	4	420	3.0549e+06	0	0	3.0549e+06
4	1	345	2.9920e+06	0	0	2.9920e+06
5	3	321	2.4168e+06	0	0	2.4168e+06
6	7	176	1.4812e+06	0	0	1.4812e+06
7	8	260	1.3814e+06	0	0	1.3814e+06
8	9	107	8.8441e+05	0	0	8.8441e+05
9	6	168	7.3853e+05	0	0	7.3853e+05

Abbildung 12: Umsatz-Tabelle. Nach Gesamtumsatz sortierte Tabelle zur Darstellung der Umsätze der einzelnen Angestellten

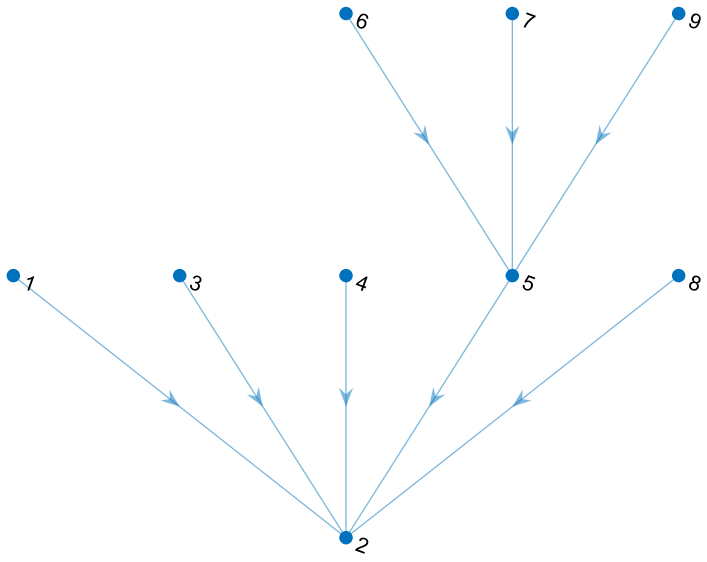

Abbildung 13: Personalstruktur. Darstellung der Personalstruktur durch Visualisierung des Ablaufs der Berichts-Kette zwischen den Angestellten

```
Log Datei

Durchlaufnr.:  2

Projekt:  NorthWind Umsatz

Datum:  25-Mar-2021
Uhrzeit:  16 : 9

Anzahl Knoten:        2278
Anzahl Kanten:        9376

Veraenderte Daten
ID     ursprungsWert   angepassterWert        Grund

Ergaenzte Daten
ID       Wert

Exkludierte Daten
ID      Grund

Isolierte Knoten
ID        Typ

Legende: Grund
                  1                1                   1                                               1                         1                          1
  Mengenangabe fehlt   Nachlass fehlt   Stueckpreis fehlt   Angestellte: weder Auftraege noch Vorgesetzter   Auftrag: keine Produkte    Auftrag: nicht verkauft
```

Abbildung 14: Log-Datei. Darstellung der Log-Datei für den Graphmining-Durchlauf 2

Jedoch geht aus der tabellarischen Darstellung der Umsätze deutlich hervor, dass unter Einbezug der indirekten Umsätze die Angestellten 2 und 5 einen größeren Einfluss auf die Generierung von Umsätzen haben. Der Grund dafür wird durch Betrachtung der hierarchischen Personalstruktur deutlich (siehe Abbildung 13). In Abbildung 13 wird deutlich, dass die Angestellten 2 und 5 Vorgesetzte sind. Zusätzlich ist der Angestellte 2 u. a. der Vorgesetzte vom Angestellte 5. Es wird insgesamt deutlich, dass der Angestellte 2 indirekt an allen Aufträgen beteiligt ist.

Durch die zusätzlich gewonnenen Erkenntnisse aus dem zweiten Durchlauf des Graphmining-Prozesses, lässt sich die Fragestellung, welcher Angestellte für die Generierung des größten Umsatzes verantwortlich ist, zufriedenstellend beantworten. Durch das Graphmining wurde die Notwendigkeit der Berücksichtigung von Personalstrukturen entdeckt. So liefert der Graphmining-Prozess eine mehrschichtige Antwort auf die Frage. Durch den direkten Verkauf von Aufträgen generiert der Angestellte 4 den größten Umsatz. Werden die hierarchischen Ebenen der Personalstruktur (siehe Abbildung 14) betrachtet, werden drei Ebenen deutlich (Ebene 1: Angestellte 6, 7, 9; Ebene 2: Angestellte 1, 3, 4, 5, 8 und Ebene 3: Angestellte 2). Betrachtet man hierbei die direkten Umsätze, generieren die Angestellten 7, 4 und 2 die größten Umsätze. Dadurch, dass den Angestellten 2 und 5 andere Angestellte unterstellt sind, sind sie insgesamt für einen großen Teil der Umsätze mitverantwortlich. Insbesondere ist dabei der Angestellte 2 hervorzuheben, der indirekt an allen Umsätzen beteiligt ist.

5.3 Diskussion und Fazit

Im folgenden Abschnitt erfolgen eine Diskussion und Bewertung des entwickelten Graphmining-Konzeptes. Dabei wird insbesondere auf die prototypische Anwendung des Konzeptes auf den angepassten NorthWind-Datensatz (siehe Abschnitt 5.1) eingegangen.

Das in Kapitel 4 entwickelte Graphmining-Konzept bietet einen strukturierten Ablauf zur Durchführung von Graphmining. Damit bildet das Konzept eine gute Ergänzung zu bereits vorhandenen Data Mining Prozessen (siehe Abschnitt 3.1). Es wird insbesondere durch die Entwicklung des Konzeptes eine Lücke in der Forschung bzgl. des Graphminings in der SC Domäne geschlossen. Es wurde gezeigt, dass sich Graphdatenbanken besser für die Speicherung von Daten aus der SC Domäne eignen und somit auch das Graphmi-

ning dem Data Mining auf relationalen Datenbanken vorzuziehen ist. Durch die Entwicklung des Konzeptes in Anlehnung an den Data Mining Prozess nach Kleissner (1998) (siehe Abschnitt 3.1) wurde sichergestellt, dass alle wichtigen Bestandteile eines Graphmining-Prozesses beinhaltet sind.

Durch die prototypische Anwendung des Konzeptes auf den NorthWind-Datensatz, wurde die Durchführbarkeit des Konzeptes gezeigt. Dabei ist allerdings zu beachten, dass es sich bei dem Beispiel um ein Minimal-Beispiel handelt. Minimal-Beispiel bedeutet, dass der Umfang der analysierten Daten gering ist (2278 Knoten und 9376 Kanten). Zudem wurde nur eine einzige Datenquelle verwendet. Dadurch kann die Leistungsfähigkeit des Konzeptes in Bezug auf große Datenmengen nicht vollständig überprüft werden. Dadurch, dass nur eine einzelne Datenquelle verwendet wurde, war zudem der Zusammenhang zwischen den verwendeten Daten klar ersichtlich. Somit konnten die Daten im Testfall gut zusammengeführt werden.

Die Wahl des NorthWind-Datensatzes ist in Bezug auf die Anwendungsdomäne des Konzeptes, als Mininmal-Beispiel sinnvoll, da durch die Daten ein Produkt-Verkaufs-System aufgespannt wird (siehe Abschnitt 5.1). Aus diesem Grund lassen sich die Schlüsse auf Basis der Mininmal-Anwendung qualitativ auf größere Beispiele übertragen.

Die zu untersuchende Fragestellung und damit das Ziel der Datenuntersuchung beschäftigt sich mit einem einfachen und intuitiven Zusammenhang zwischen den Daten. Für eine erste prototypische Anwendung des Konzeptes sind sowohl die Fragestellung als auch die Daten von hinreichender Komplexität. Schon bei der gewählten einfachen Untersuchung zeigte sich, dass eine Anpassung der kontextbezogenen Vorüberlegungen in einem zweiten Durchlauf des Graphmining-Prozesses notwendig (siehe Abschnitt 5.2) war. Daraus lassen sich zwei Schlüsse ziehen. Zum einen müssen selbst vermeintlich einfache Fragestellungen mit einem iterativen Prozess untersucht werden. Das Verständnis der Daten wächst mit den Graphmining-Durchläufen, sodass bessere Annahmen getroffen werden können. Zum anderen wird deutlich, dass mehrere unterschiedliche Visualisierungen von Graphmining-Ergebnissen benötigt werden, um fundiert Schlüsse ziehen zu können. So wurden in Abschnitt 5.2 die Anpassungen des Umgangs mit fehlenden Daten sowie des Algorithmus unter Berücksichtigung der Log-Datei, der Schema-Darstellung und der Algorithmus-Ergebnis-Darstellung getroffen. Die alleinige Darstellung der Graphmining-Algorithmus-Ergebnisse, hätte keine Ansatzpunkte dafür gegeben, dass

auch hierarchische Strukturen im Personalwesen zu berücksichtigen sind. Durch das entwickelte Konzept konnten diese beiden Aspekte innerhalb des Graphmining-Prozesses abgebildet werden.

Die Lösung von einfachen Problemstellungen wurde anhand der Anwendung des Konzeptes demonstriert. Da durch den iterativen Aufbau des Prozesses komplexe Problemstellungen in einfache Unterprobleme unterteilt werden können, lässt dies den Schluss zu, dass das Konzept auch zur Bearbeitung von komplexeren Problemen geeignet ist.

Insgesamt lässt sich festhalten, dass das Konzept sowohl Stärken als auch Schwächen aufweist. Zum einen bietet es Orientierung bei der Anwendung von Graphmining-Prozessen. Dabei wird berücksichtigt, dass zu analysierende Daten häufig aus unterschiedlichen Quellen stammen und in unterschiedlichen Datenstrukturen vorliegen können (siehe Abschnitt 2.2 und Abschnitt 2.3). Zudem ist es möglich Anpassungen auf Grundlage von gewonnen Erkenntnissen vorzunehmen, sodass sich einer qualitativ hochwertigen Lösung schrittweise genähert werden kann. Das Konzept gliedert sich in vier Hauptphasen: 1) kontextbezogene Vorüberlegungen, 2) Datenvorbereitung, 3) Ausführung des Graphmining-Algorithmus und 4) Bewertung der Graphmining-Algorithmus-Ergebnisse. Diese Unterteilung bietet den Vorteil, dass Programmierung (Phasen 2 und 3) und kontextbezogene Überlegungen und Bewertungen (Phasen 1 und 4) voneinander getrennt werden. Auf diese Weise wird die Anwendung von Graphmining erleichtert, da unterschiedliche Kompetenzgebiete nicht vermischt werden.

Jedoch sind viele Unterpunkte des Konzeptes allgemein formuliert, sodass der Anwender des Konzeptes eigenständig Lösungen finden muss. Dies liegt zum einen daran, dass das Konzept noch nicht ausreichend an Fallbeispielen erprobt wurde. Durch solche Anwendungen kann das Konzept verbessert und verfeinert werden. Zum anderen können viele Teile des Konzeptes nicht unabhängig von der zu beantwortenden Fragestellung konkretisiert werden. Insbesondere die Auswahl eines geeigneten Graphmining-Algorithmus ist komplex und kann nicht ohne Berücksichtigung der Anwendungsdomäne und der zugrundeliegenden Fragestellung erfolgen. Auf der anderen Seite ist das Konzept in dieser Form nicht auf einen Typ von Problemstellung festgelegt, sondern kann für eine Vielzahl von Anwendungen genutzt werden. Insbesondere in der Anwendungsdomäne SC ist dies von Vorteil, da dort ein großes Spektrum von Problemstellungen mit unterschiedlichen Charakteristika behandelt wird. Zudem lassen sich Wechselwirkungen zwischen unter-

schiedlichen Prozessschritten feststellen. Bspw. werden in den kontextbezogenen Vorüberlegungen Datenquellen und Stichproben gewählt. Diese Daten werden zur Beantwortung der Fragestellung analysiert. Die Daten werden unter Berücksichtigung von Kontextwissen in der Problemdomäne gewählt. Darauf basierend kann anschließend ein Graphmining-Algorithmus gewählt werden. Andererseits können Graphmining-Algorithmen Anforderungen an Daten und ihre Struktur stellen, sodass für die gewünschte Anwendung eines speziellen Algorithmus Daten eventuell angepasst werden müssen.

Das Graphmining-Konzept ist für die Anwendung in MATLAB konzipiert. Es zeigte sich jedoch, dass die grundlegenden Schritte unabhängig von dem verwendeten Anwendungsprogramm sind. Das Anwendungsprogramm spielt eine übergeordnete Rolle bei der konkreten Anwendung des Konzeptes. Dabei sind die unterschiedlichen Programmspezifika zu beachten. Insbesondere der Umgang mit verschiedenen Datenstrukturen und deren Umwandlung ineinander muss berücksichtigt werden. MATLAB bietet diesbezüglich eine gute Plattform für die Anwendung des Graphmining-Prozesses, da Datentypen und -strukturen leicht ineinander überführt werden können. Zusätzlich hat sich auch gezeigt, dass die Anbindung an Neo4j eine Erleichterung bei der Durchführung von Abfragen direkt auf der zu analysierenden Datenbank bietet. Dadurch wird das Einbinden, Manipulieren und Abfragen von Daten erleichtert. Jedoch ist bei der Verwendung von MATLAB zu beachten, dass MATLAB nicht auf die Verarbeitung von Graphstrukturen spezialisiert ist. Dadurch erhöht sich an einigen Stellen der Aufwand für Daten in Graphstrukturen. Auf der anderen Seite verbessert MATLAB die Möglichkeiten der Einbindung von Graphmining in andere bereits bestehende Prozesse, die mit MATLAB realisiert wurden.

Insgesamt bietet das entwickelte Konzept eine gute Anleitung zur Durchführung eines Graphmining-Prozesses in MATLAB im Rahmen einer KDD zur Bearbeitung von Fragestellungen aus der SC Domäne. Insbesondere durch die Nutzung eines in der Industrie weitverbreiteten Anwendungsprogrammes bietet das Konzept einen Mehrwert für die Nutzung von Graphmining in der industriellen Anwendung. Eine Verbesserung der praktischen Anwendbarkeit wurde durch die Struktur des Konzeptes erreicht. Darin werden unterschiedliche Kompetenzbereich durch die einzelnen Hauptphasen des Konzeptes getrennt.

6 Zusammenfassung und Ausblick

In der Wissenschaft und der Wirtschaft werden immer mehr Daten erhoben und abgespeichert, so auch in der SC Domäne. Die SC Domäne zeichnet sich durch ihren vernetzten Charakter aus. Insbesondere stehen Transaktionsdaten im Fokus des SCM. Aus diesem Grund ist es sinnvoll Daten in Graphstrukturen in Graphdatenbanken zu speichern. In Graphstrukturen wird ein Fokus auf die Verbindung zwischen den Daten gelegt. Verbindungsdaten entsprechen in der SC den Transaktionsdaten. Um Wissen aus den in der SC anfallenden Daten ableiten zu können, kann im Rahmen von KDD Graphmining durchgeführt werden. In dieser Arbeit wurde ein Konzept für das Graphmining in MATLAB entwickelt.

Um ein solches Konzept entwickeln zu können, war es zunächst notwendig, Grundlagen der Anwendungsdomäne, also der SC, (siehe Abschnitt 2.1) zu erarbeiten. Damit in Verbindung stehen die Daten, die in einer SC anfallen (siehe Abschnitt 2.2). Um die Daten beschreiben zu können, müssen Daten und Datenstrukturen beschrieben werden. Dies erfolgte in Abschnitt 2.2. Die bspw. in einer SC anfallenden Daten müssen in einer strukturierten Form, die auch die Verbindung unter den Daten verdeutlicht, abgespeichert werden. Das Abspeichern passiert in Datenbanken. Datenbanken weisen je nach zugrundeliegender Datenstruktur unterschiedliche Charakteristika auf. Diese wurden in Abschnitt 2.3 beschrieben. Durch die Überlegungen in Kapitel 2 wurde deutlich, dass sich Graphen für die Abbildung von SC Daten gut eignen.

Die in einer SC erhobenen Daten dienen einem besseren Verständnis der Prozesse in SCs. Wie Daten dafür weiterverarbeitet werden müssen, wurde in Kapitel 3 erläutert. Um ein besseres Verständnis zu generieren, also Wissen aus den Daten abzuleiten, ist es notwendig die Daten weiter zu verarbeiten und zu analysieren. Dies kann in einem KDD Prozess realisieret werden (siehe Abschnitt 3.1). Ein solcher Prozess beinhaltet u. a. das Data Mining. Dabei wurde insbesondere das Data Mining Vorgehen nach Kleissner (1998) berücksichtigt (siehe Abschnitt 3.2). Da, wie in Kapitel 2 herausgestellt, Daten aus der SC Domäne aufgrund ihrer Charakteristik, gut als Graph darstellbar sind, kann Graphmining als eine Spezifikation des Data Minings verwendet werden (siehe Abschnitt 3.1 und Abschnitt 3.2). Mögliche Tools, die beim Graphmining zur Anwendung kommen können, wurden anschließend in Abschnitt 3.2 kurz beschrieben. Anschließend konnte, angelehnt

an den Data Mining Prozess nach Kleissner (1998) (siehe Abbildung 2), ein Graphmining-Konzept für die Umsetzung in MATLAB entwickelt werden (siehe Kapitel 4). Dabei ließen sich vier Hauptphasen (kontextbezogene Vorüberlegungen (siehe Abschnitt 4.1), Datenvorbereitung (siehe Abschnitt 4.2), Ausführung des Graphmining-Algorithmus und Bewertung der Graphmining-Algorithmus-Ergebnisse (siehe Abschnitt 4.2) identifizieren. Zudem wurde deutlich, dass die unterschiedlichen Phasen nicht unabhängig voneinander zu betrachten sind, da sie sich gegenseitig beeinflussen.

Um die Praktikabilität des entwickelten Konzeptes zu überprüfen, wurde das Konzept anschließend in Kapitel 5 prototypisch auf einen Datensatz angewendet. Der verwendete Datensatz bildete dabei eine Minimal-Darstellung einer SC (siehe Abschnitt 5.1). Durch die Anwendung des Konzeptes auf ein Beispiel war es dann möglich, zu ersten Einschätzungen über die Stärken und Schwächen des Konzeptes zu gelangen (siehe Abschnitt 5.3). Insgesamt lässt sich festhalten, dass erfolgreich ein nutzbares Konzept für die Durchführung von Graphmining innerhalb MATLABs entwickelt und erprobt wurde.

Basierend auf der Diskussion und dem Fazit in Abschnitt 5.3 lassen sich zahlreiche Ansatzpunkte für weitere Untersuchungen und Anwendungen des Graphmining-Konzeptes ableiten.

Aus dem Fazit ergibt sich u. a. die Notwendigkeit, das Konzept auf eine größere Datenmenge aus unterschiedlichen Datenquellen anzuwenden. In diesem Zusammenhang wäre auch von Interesse, eine Automatisierungsmöglichkeit von Einlesevorgängen bei Daten aus unterschiedlichen Datenquellen zu untersuchen. Zudem sollten komplexere Fragestellungen betrachtet werden, sodass die Effizienz und Notwendigkeit des iterativen Charakters des Konzeptes überprüft werden kann bzw. Verbesserungen an den Anpassungsmöglichkeiten vorgenommen werden können. In diesem Zusammenhang wäre eine Bewertung des Konzeptes durch einen Analysten aus dem SCM von Interesse.

Beim Graphmining werden für gewöhnlich große Mengen an Daten analysiert. In diesem Zug kann eine parallele Verarbeitung von Daten zur Verbesserung der Laufzeit beitragen. Hierbei wäre zu untersuchen, inwieweit das parallel computing in MATLAB dafür nutzbar ist. Zudem ergibt sich daraus die Fragestellung, wie sich Graphen aufteilen lassen. Bei der Verarbeitung von großen Datenmengen ist auch die Datenqualität von Bedeutung. In dieser Arbeit wurde davon ausgegangen, dass Datenqualitätsstandards eingehalten wurden. Um die Auswirkungen der Datenqualität auf Graphmining-Erkenntnisse besser

einschätzen zu können bzw. um Erkenntnisse darüber zu erlangen, welche Datenqualitätsmerkmale von entscheidender Bedeutung sind, wären noch weitere Untersuchen notwendig.

Die Auswahl der Graphmining-Algorithmen erfolgt bisher nur auf Grundlage von Erfahrungen und persönlichen Einschätzungen. Aus diesem Grund wäre es wünschenswert, dass ein Konzept zur Auswahl von Graphmining-Algorithmen entwickelt wird. Dadurch würde die Anwendbarkeit von Graphmining auch für Analysten mit wenig Erfahrung erleichtert. Daran anknüpfend wäre es möglich die Graphmining-Algorithmen aus dem definierten Auswahlkonzept konkret in MATLAB umzusetzen. Die Kombination aus einem Graphmining-Algorithmus-Auswahl-Konzept, entsprechend vordefinierten Algorithmen und einem Graphmining-Konzept würde den Vorteil bieten, dass Graphmining auch mit wenig theoretischen Vorkenntnissen auf einem hohen Qualitätsniveau durchgeführt werden könnte.

Da Datenanalyse mittels Graphmining nicht nur mit Hilfe von MATLAB durchgeführt werden kann, wäre eine Übertragung des Konzeptes auf andere Programme, wie bspw. RapidMiner, von Interesse.

Da für Graphmining-Untersuchungen häufig Daten zunächst in eine Graphstruktur überführt werden müssen, wären vergleichende Untersuchungen, bei denen Data Mining auf den Daten in ihrer ursprünglichen Datenstruktur durchgeführt wird, von Interesse. Dabei könnte der Aufwand des verwendeten Data Minings im Vergleich zum Graphmining untersucht werden. Von besonderem Interesse wäre dabei, ob die gleichen Erkenntnisse gewonnen werden würden.

Literaturverzeichnis

Angelova, R. (2009), „Graph-based Classification and Clustering of Entities in Hetrerogeneous Networks“, Dissertation, Max-Planck-Institut für Informatik, Universität des Saarlandes, Saarbrücken, 2009.

Angles, R. (2012), „A Comparison of Current Graph Database Models“, in *2012 IEEE 28th International Conference on Data Engineering workshops (ICDEW 2012), 1.-5. April 2012, Arlington, VA, USA*, IEEE, Piscataway, NJ, S. 171–177.

Anikin, D., Borisenko, O. und Nedumov, Y. (2019), „Labeled Property Graphs: SQL or NoSQL?“, in Prokhorov, S. (Hg.), *2019 Ivannikov Memorial Workshop, 13.-14. September 2019, Velikiy Novgorod, Russia*, Conference Publishing Services, IEEE Computer Society, Los Alamitos, California, Washington, Tokyo, S. 7–13.

Arndt, H. (2008), *Supply Chain Management: Optimierung logistischer Prozesse, Gabler Lehrbuch,* 4., aktualisierte und überarb. Aufl., Gabler, Wiesbaden.

Arnold, D., Isermann, H., Kuhn, A., Tempelmeier, H. und Furmans, K. (2008), *Handbuch Logistik*, Springer Berlin Heidelberg, Berlin, Heidelberg.

Batra, R. (2018), *SQL Primer: An Accelerated Introduction to SQL Basics*, Apress, Berkeley, CA.

Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinl, T., Ohl, P., Thiel, K. und Wiswedel, B. (2009), „KNIME - the Konstanz information miner“, *ACM SIGKDD Explorations Newsletter*, 11. Jg., Nr. 1, S. 26–31.

Bissantz, N. und Hagedorn, J. (2009), „Data Mining (Datenmustererkennung)“, *WIRTSCHAFTSINFORMATIK*, 51. Jg., Nr. 1, S. 139–144.

Blackstone, J. H. (Hg.) (2010), *APICS dictionary,* 13. Aufl., APICS, Alexandria, VA.

Cai, L. und Zhu, Y. (2015), „The Challenges of Data Quality and Data Quality Assessment in the Big Data Era“, *Data Science Journal*, 14. Jg., Nr. 0, S. 2.

Celko, J. (2014), *Joe Celko's Complete Guide to NoSQL: What Every SQL Professional Needs to Know About Nonrelational Databases*, Elsevier.

Chen, M.-S., Han, J. und Yu, P. S. (1996), „Data mining: an overview from a database perspective“, *IEEE Transactions on Knowledge and Data Engineering*, 8. Jg., Nr. 6, S. 866–883.

Chopra, S. und Meindl, P. (2004), *Supply Chain Management: Strategy, Planning and Operation,* 2. Aufl., Pearson Education Inc, Upper Saddle River, NJ.

Christopher, M. (2016), *Logistics & supply chain management, Always learning,* fifth edition, Pearson Education, Harlow, England, New York.

Corsten, H. und Gössinger, R. (2008), *Einführung in das Supply Chain Management, Lehr- und Handbücher der Betriebswirtschaftslehre,* 2., vollst. überarb. und wesentlich erw. Aufl., Oldenbourg, München.

Dippold, R., Meier, A., Schnider, W. und Schwinn, K. (2005), *Unternehmensweites Datenmanagement*, Vieweg+Teubner Verlag, Wiesbaden.

Drakopoulos, G., Baroutiadi, A. und Megalooikonomou, V. (2015), „Higher order graph centrality measures for Neo4j“, in *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA): Date: 6-8 July 2015, 7/6/2015 - 7/8/2015, Corfu, Greece*, IEEE, Piscataway, NJ, S. 1–6.

Durand, G. C., Ma, J., Pinnecke, M. und Saake, G. (2018), „Piecing together large puzzles, efficiently: Towards scalable loading into graph database systems“, in Klassen, G. und Stefan Conrad (Hg.), *Grundlagen von Datenbanken*, Wuppertal, S. 95–100.

Džeroski, S. (2003), „Multi-relational data mining“, *ACM SIGKDD Explorations Newsletter*, 5. Jg., Nr. 1, S. 1–16.

Edlich, S., Friedland, A., Hampe, J., Brauer, B. und Brückner, M. (2011), *NoSQL: Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken,* 2., aktualisierte und erw. Aufl., Hanser, München.

Ezhilchelvan, P., Mitrani, I. und Webber, J. (2020), „Modeling the Gradual Degradation of Eventually-Consistent Distributed Graph Databases“, *Queueing Models and Service Management*, 3. Jg., Nr. 2, S. 235–253.

Fasel, D. (2016), „Übersicht über NoSQL-Technologien und -Datenbanken“, in Fasel, D. und Meier, A. (Hg.), *Big Data: Grundlagen, Systeme und Nutzungspotenziale*, *Edition HMD*, Springer Vieweg, Wiesbaden, S. 109–137.

Fayyad, U. und Uthurusamy, R. (1996), „Data mining and knowledge discovery in databases“, *Communications of the ACM*, 39. Jg., Nr. 11, S. 24–26.

Feltrin, L. (2015), „KNIME an Open Source Solution for Predictive Analytics in the Geosciences. Software and Data Sets“, *IEEE Geoscience and Remote Sensing Magazine*, 3. Jg., Nr. 4, S. 28–38.

Frawley, W. J., Piatetsky-Shapiro, G. und Matheus, C. J. (1992), „Knowledge Discovery in Databases: An Overview“, *AI Magazine*, 13. Jg., Nr. 3, S. 57.

Garcia-Molina, H., Ullman, J. D. und Widom, J. (2009), *Database Systems: The Complete Book, Pearson international edition,* 2. ed., internat. ed., Pearson/Prentice Hall, Upper Saddle River, NJ.

Geetha, A. und Nasira, G. M. (2014), „Artificial Neural Networks’ Application in Weather Forecasting – Using RapidMiner“, *International Journal of Computational Intelligence and Informatics*, 4. Jg., Nr. 3, 177-182.

GitHub (2015), „neo4j-documentation/developer-resources. northwind.zip“, verfügbar unter https://github.com/neo4j-documentation/developer-resources/blob/gh-pages/data/northwind/northwind.zip (Zugriff am 15. März 2021).

Günther, H.-O. und Tempelmeier, H. (2003), *Produktion und Logistik, Springer-Lehrbuch,* 5., verb. Aufl., Springer, Berlin.

Hahn, D. und Kaufmann, L. (2002), *Handbuch Industrielles Beschaffungsmanagement: Internationale Konzepte - Innovative Instrumente - Aktuelle Praxisbeispiele,* 2., überarbeitete und erweiterte Auflage, Gabler Verlag, Wiesbaden, s.l.

Han, J., Rodriguez, J. C. und Beheshti, M. (2008), „Diabetes Data Analysis and Prediction Model Discovery Using RapidMiner“, in *Proceedings of the 2008 Second International Conference on Future Generation Communication and Networking, 12/13/2008 - 12/15/2008, Hainan, China*, IEEE, Piscataway, NJ, S. 96–99.

Haun, S., Nürnberger, A., Kötter, T., Thiel, K. und Berthold, M. R. (2010), „CET: A Tool for Creative Exploration of Graphs“, in Balcázar, J. L., Bonchi, F., Gionis, A. und Sebag, M. (Hg.), *Machine learning and knowledge discovery in databases: European conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010; proceedings*, *Lecture notes in computer science Lecture notes in artificial intelligence*, Bd. 6323, Springer, Berlin, S. 587–590.

Hazen, B. T., Boone, C. A., Ezell, J. D. und Jones-Farmer, L. A. (2014), „Data quality for data science, predictive analytics, and big data in supply chain management: An introduction to the problem and suggestions for research and applications“, *International Journal of Production Economics*, 154. Jg., S. 72–80.

Hildebrand, K., Gebauer, M., Hinrichs, H. und Mielke, M. (2015), *Daten- und Informationsqualität*, Springer Fachmedien Wiesbaden, Wiesbaden.

Hitt, M. A. (2011), „Relevance of Strategic Management Theory and Research for Supply Chain Management“, *Journal of Supply Chain Management*, 47. Jg., Nr. 1, S. 9–13.

Hunker, J., Scheidler, A. A. und Rabe, M. (2020), „A systematic classification of database solutions for data mining to support tasks in supply chains“, in Kersten, W.,

Blecker, T. und Ringle, C. (Hg.), *Data Science and Innovation in Supply Chain Management: How Data Transforms the Value Chain*, epubli, 395-425.

Jungermann, F. (2009), *Information Extraction with RapidMiner*, Duisburg.

Junghanns, M. und Petermann, A. (2016), „Verteilte Graphanalyse mit Gradoop“, *JavaSPEKTRUM*, Nr. 5, S. 56–60.

Junghanns, M., Petermann, A., Neumann, M. und Rahm, E. (2017), „Management and Analysis of Big Graph Data: Current Systems and Open Challenges“, in Zomaya, A. Y. und Sakr, S. (Hg.), *Handbook of Big Data technologies*, Springer, Cham, S. 457–505.

Junghanns, M., Petermann, A., Teichmann, N., Gómez, K. und Rahm, E. (2016), „Analyzing extended property graphs with Apache Flink“, in Arora, A., Roy, S. und Mehta, S. (Hg.), *Proceedings of the 1st ACM SIGMOD Workshop on Network Data Analytics, 01.07.2016, San Francisco California*, ACM, New York, NY, S. 1–8.

Khasawneh, T. N., AL-Sahlee, M. H. und Safia, A. A. (2020), „SQL, NewSQL, and NOSQL Databases: A Comparative Survey“, in *2020 11th International Conference on Information and Communication Systems (ICICS), 07.04.2020 - 09.04.2020, Irbid, Jordan*, IEEE, S. 13–21.

Kleijnen, J. P. (2005), „Supply chain simulation tools and techniques: a survey“, *International Journal of Simulation and Process Modelling*, 1. Jg., Nr. 1/2, S. 82.

Kleissner, C. (1998), „Data mining for the enterprise“, in *Proceedings of the Thirty-First Hawaii International Conference on System Sciences, 6-9 Jan. 1998, Kohala Coast, HI*, IEEE Computer Society, Los Alamitos, Calif., S. 295–304.

Klöckner, K. (2015), *Im Vergleich: NoSQL vs. relationale Datenbanken*, Siegen.

Kotu, V. und Deshpande, B. (2014), „Chapter 13 - Getting Started with RapidMiner“, in Kotu, V. und Deshpande, B. (Hg.), *Predictive Analytics and Data Mining: Concepts and Practice with RapidMiner*, Elsevier Science, Burlington, S. 371–406.

Kovács, T., Simon, G. und Mezei, G. (2019), „Benchmarking Graph Database Backends—What Works Well with Wikidata?“, *Acta Cybernetica*, 24. Jg., Nr. 1, S. 43–60.

Kudraß, T. (2015), „Datenbanken: Grundlagen und Überblick“, in Kudraß, T. und Brinkhoff, T. (Hg.), *Taschenbuch Datenbanken*, 2., neu bearbeitete Auflage, Hanser, München, S. 19–43.

Kugran, L. A. und Musilek, P. (2006), „A survey of Knowledge Discovery and Data Mining process models“, *The Knowledge Engineering Review*, 21. Jg., Nr. 1, S. 1–24.

Law, A. M. (2015), *Simulation modeling and analysis,* Fifth edition, international student edition, McGraw Hill Education, New York, NY.

Leavitt, N. (2010), „Will NoSQL Databases Live Up to Their Promise?“, *Computer*, 43. Jg., Nr. 2, S. 12–14.

MathWorks (2020a), „Database Toolbox. User's Guide“, verfügbar unter https://de.mathworks.com/help/pdf_doc/database/database.pdf (Zugriff am 3. März 2021).

MathWorks (2020b), „Documentation“, verfügbar unter https://de.mathworks.com/help/matlab/index.html (Zugriff am 3. März 2021).

MathWorks Deutschland (2021), „TU Dortmund - MATLAB Access for Everyone“, verfügbar unter https://de.mathworks.com/academia/tah-portal/tu-dortmund-31486497.html (Zugriff am 15. März 2021).

Mehta, D. P. und Sahni, S. (2018), *Handbook of data structures and applications, Chapman & Hall/CRC computer information science series,* Second edition, CRC Press Taylor & Francis Group, Boca Raton, London, New York.

Meier, A. (2016), „Datenmanagement mit SQL und NoSQL“, in Fasel, D. und Meier, A. (Hg.), *Big Data: Grundlagen, Systeme und Nutzungspotenziale*, *Edition HMD*, Springer Vieweg, Wiesbaden, S. 17–38.

Meier, A. und Kaufmann, M. (2016), *SQL- & NoSQL-Datenbanken*, Springer, Berlin, Heidelberg.

Mentzer, J. T., DeWitt, W., Keebler, J. S., Min, S., Nix, N. W., Smith, C. D. und Zacharia, Z. G. (2001), „Defining Supply Chain Management“, *Journal of Business Logistics*, 22. Jg., Nr. 2, S. 1–25.

Mertens, P., Bodendorf, F., König, W., Schumann, M., Hess, T. und Buxmann, P. (2017), *Grundzüge der Wirtschaftsinformatik, Lehrbuch,* 12., grundlegend überarbeitete Auflage, Springer Gabler, Berlin.

Neo4j (2020), „The Neo4j Operations Manual v4.2“, verfügbar unter https://neo4j.com/docs/pdf/neo4j-operations-manual-4.2.pdf (Zugriff am 8. März 2021).

Neo4j Graph Database Platform (2020), „Neo4j Desktop Download. Launch and Manage Neo4j Databases“, verfügbar unter https://neo4j.com/download/ (Zugriff am 15. März 2021).

North, K. (2011), *Wissensorientierte Unternehmensführung: Wertschöpfung durch Wissen, Lehrbuch,* 5., aktualisierte und erweiterte Auflage, Gabler, Wiesbaden.

North, K. (2016), „Die Wissenstreppe“, in North, K. (Hg.), *Wissensorientierte Unternehmensführung: Wissensmanagement gestalten*, *Lehrbuch*, 6., aktualisierte und erweiterte Auflage, Springer Gabler, Wiesbaden, S. 33–65.

Ottmann, T. und Widmayer, P. (2012), *Algorithmen und Datenstrukturen,* 5. Aufl., Spektrum Akad. Verl., Heidelberg.

Preiß, N. (2007), *Entwurf und Verarbeitung relationaler Datenbanken*, R. Ouldenbourg Verlag, München, Wien.

Rehman, S. U., Khan, A. U. und Fong, S. (2012), „Graph mining: A survey of graph mining techniques“, in Fong, S. (Hg.), *2011 Seventh International Conference on Digital Information Management (ICDIM 2012), 8/22/2012 - 8/24/2012, Macau, Macao*, IEEE, Piscataway, NJ, S. 88–92.

Ristoski, P., Bizer, C. und Paulheim, H. (2015), „Mining the Web of Linked Data with RapidMiner“, *Journal of Web Semantics*, 35. Jg., S. 142–151.

Robinson, I. (2015), *Graph databases: New opportunities for connected data,* Second edition, O'Reilly, Sebastopol, CA.

Rostami, M. A., Kricke, M., Peukert, E., Kühne, S., Wilke, M., Dienst, S. und Rahm, E. (2019), „BIGGR: Bringing Gradoop to Applications“, *Datenbank-Spektrum*, 19. Jg., Nr. 1, S. 51–60.

Saake, G., Heuer, A. und Sattler, K.-U. (2018), *Datenbanken: Konzepte und Sprachen, mitp Professional,* 6. Auflage, MITP, Frechen.

Sahu, S., Mhedhbi, A., Salihoglu, S., Lin, J. und Özsu, M. T. (2017), „The ubiquity of large graphs and surprising challenges of graph processing“, *Proceedings of the VLDB Endowment*, 11. Jg., Nr. 4, S. 420–431.

Scheidler, A. A. (2017), „Methode zur Erschließung von Wissen aus Datenmustern in Supply-Chain-Datenbanken“, Dissertation, Technische Universität Dortmund, Dortmund, 2017.

Schicker, E. (2014), *Datenbanken und SQL: Eine praxisorientierte Einführung mit Anwendungen in Oracle, SQL Server und MySQL, Informatik & Praxis,* 4., überarbeitete Auflage, Springer Vieweg, Wiesbaden.

Silberschatz, A., Korth, H. F. und Sudarshan, S. (2011), *Database System Concepts,* 6th edition, McGraw-Hill, New York, NY.

Stadtler, H., Kilger, C. und Meyr, H. (2015), *Supply Chain Management and Advanced Planning*, Springer Berlin Heidelberg, Berlin, Heidelberg.

Steiner, R. (2017), *Grundkurs Relationale Datenbanken*, Springer Fachmedien, Wiesbaden.

Störl, U. (2015), „NoSQL-Datenbanksysteme“, in Kudraß, T. und Brinkhoff, T. (Hg.), *Taschenbuch Datenbanken*, 2., neu bearbeitete Auflage, Hanser, München, S. 368–393.

Tanase, G., Suzumura, T., Lee, J., Chen, C.-F., Crawford, J., Kanezashi, H., Zhang, S. und Vijitbenjaronk, W. D. (2018), *System G Distributed Graph Database*.

Unland, R. und Pernul, G. (2014), *Datenbanken im Einsatz*, DE GRUYTER, Berlin, München, Boston.

van Bonn, B. (2013), „Basisdaten der Logistikplanung Logistikplanung Basisdaten“, in Clausen, U. und Geiger, C. (Hg.), *Verkehrs- und Transportlogistik*, 2. Aufl., Springer Vieweg, Berlin, S. 291–298.

Wannenwetsch, H. (2014), „Vernetztes Supply Chain Management“, in Wannenwetsch, H. (Hg.), *Integrierte Materialwirtschaft, Logistik und Beschaffung*, *Springer-Lehrbuch*, 5., neu bearb. Aufl., Springer Vieweg, Berlin, S. 499–517.

Will, T. T. (2018), *C++: Das umfassende Handbuch,* 1. Aufl., Rheinwerk, Bonn.

Wirth, R. und Hipp, J. (2000), *CRISP-DM: Towards a standard process model for data mining*.

Abbildungsverzeichnis

Abbildung 1: Wissenstreppe 6
Abbildung 2: Data Mining Prozess 16
Abbildung 3: Graphmining-Konzept zur Anwendung in MATLAB 20
Abbildung 4: Datenstrukturkombinationen bei Datenquellen 24
Abbildung 5: Beispielhafte Schema-Darstellung 27
Abbildung 6: Schematische Log-Datei 28
Abbildung 7: Struktur der NorthWind .csv-Dateien 31
Abbildung 8: Schema-Darstellung 37
Abbildung 9: Umsatz-Tabelle 38
Abbildung 10: Umsatz-Tortendiagramm 38
Abbildung 11: Log-Datei 39
Abbildung 12: Umsatz-Tabelle 41
Abbildung 13: Personalstruktur 41
Abbildung 14: Log-Datei 42

Tabellenverzeichnis

Tabelle 1: Überblick über Datenstrukturen 7
Tabelle 2: Grundlegende Begriffe in Bezug auf das relationale Datenbankmodell 9
Tabelle 3: Grundlegende Begriffe in Bezug auf das Graphdatenbankmodell 12

Abkürzungsverzeichnis

.csv-Datei	Comma-separated-values-Datei
DBMS	Datenbankmanagementsystem
DBS	Datenbanksystem
KDD	Knowledge Discovery in Databases
KNIME	Konstanz Information Miner
SC	Supply Chain
SCM	Supply Chain Management