Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik

Eignung von Graphdatenbanken zur Verwaltung multiskaler und domänenspezifischer Datenstrukturen in einer Logistikanwendung

Exposé zur Masterarbeit

eingereicht von: Dominique Hüneburg

Gutachter: Prof. Dr. Ulf Leser

Inhaltsverzeichnis

1 Motivation

Das Forschungsgebiet der Logistik beschäftigt sich unter anderem mit fachspezifischen Optimierungsproblemen und dem Thema Big Data. Nicht nur die Fortschritte auf dem Gebiet der Datensammlung und -verarbeitung hatten immer größer werdende Datenmengen zur Folge. Auch die Globalisierung und Erschließung neuer Verkehrswege führt dazu, dass Logistiknetzwerke stetig wachsen. Insbesondere sehen sich Nutzer und Entwickler von Logistikanwendungen täglich mit dem Problem konfrontiert, dass immer komplexere Fragestellungen auf immer größer werdenden Datensätzen in immer kürzerer Zeit gelöst werden müssen. Häufig fallen diese Fragestellungen in die Kategorie der Optimierungsaufgaben. Zur Lösung dieser Probleme werden laufend Optimierungsalgorithmen entwickelt und verbessert. Mit der Entwicklung solcher Algorithmen beschäftigt sich auch das Forschungsprojekt, aus dem die Frage nach der Eignung von Graphdatenbanken zur Verwaltung multiskaler und domänenspezifischer Datenstrukturen in einer Logistikanwendung hervorging. Insbesondere werden in diesem Projekt Algorithmen entwickelt, die nicht nur auf einer Menge von realen Daten mit einer für die Logistik typischen Struktur arbeiten sollen, sondern auf einer sogenannten multiskalen Datenstruktur. Diese wird ebenfalls im Rahmen des Projektes entwickelt.

Multiskale Datenstrukturen sind Datenstrukturen, die dieselben Daten auf unterschiedlichen Granularitätsstufen, d.h. unter Verwendung verschiedener Skalen darstellen können. Zum Beispiel lassen sich Daten chemischer Prozesse, die stündlich beobachtet werden, aggregieren zu Daten, aus denen die Entwicklung nur noch tageweise hervorgeht. Damit verändert sich die betrachtete Zeitskala und die Daten sind weniger granular. Der Vorteil der aggregierten Daten ist die geringere Menge, der Nachteil ist Informationsverlust. Eine multiskale Datenstruktur dient der gemeinsamen Verwaltung von Originaldaten und aggregierten Daten. Im Forschungsprojekt betrifft diese Aggregation geographische Positionsdaten.

Das Ziel von Optimierungsalgorithmen, die auf einer multiskalen Datenstruktur arbeiten, ist eine deutlich schnellere Berechnung der Problemlösung mit trotzdem möglichst guter Qualität. Auf Basis der Ergebnisse kann die Lösung anschließend noch verfeinert werden. Dieses Vorgehen soll dem Umstand begegnen, dass in der Logistik alltäglich durchgeführte Berechnungen auf wachsenden Datenmengen immer mehr Zeit kosten.

2 Zielstellung

Zu den im Forschungsprojekt entwickelten Algorithmen und der multiskalen Datenstruktur wurde bisher keine persistente Datenhaltung konzipiert. Auch gilt es noch zu evaluieren, ob die Ergebnisse des Projektes in eine konkrete bestehende Logistikanwendung eingebracht werden können. Die Masterarbeit wird sich mit der Frage beschäftigen, ob Graphdatenbanken für diese Zwecke geeignet sind.

Graphdatenbanken gehören zu den sogenannten *NoSQL-Datenbanken*, wobei der Ausdruck *NoSQL* sowohl als Akronym für *Not only SQL* als auch für *No to SQL* verstanden werden kann (RWE13). Treffender ist allerdings die Bezeichnung *nichtrelationale Datenbanken* (VW15). Graphdatenbanken wurden zur Speicherung netzwerkartiger Datenstrukturen entwickelt. Mit ihnen lassen sich Entitäten und deren einzelne Verbindungen untereinander als Knoten und Kanten eines Graphen verwalten, wodurch sich das Netzwerk effizient traversieren lässt. Die Entitäten und Verbindungen sind durch beliebig viele, individuell definierbare Eigenschaften näher beschreibbar. Graphdatenbanken sind daher nicht domänenspezifisch. Fragestellungen, die typisch sind für eine netzwerkartige Datenstruktur, lassen sich weniger aufwändig beantworten als in anderen Datenbankkonzepten. Graphdatenbanken werden unter anderem eingesetzt in der Robotik zur Modellierung von Abläufen in der Sensordatenerfassung und -verarbeitung (HVK16) sowie als Wissensdatenbanken für komplexe Zusammenhänge in der Chemie (HMV17) und Biologie (YKK17).

Neben Graphdatenbanken existiert eine Reihe weiterer nichtrelationaler Datenbankkonzepte, darunter Schlüssel-Werte-Speicher, spaltenorientierte und dokumentenorientierte Datenbanken (VW15). Den nichtrelationalen Datenbanken stehen relationale Datenbanken gegenüber als das in den vergangenen Jahren kommerziell am weitesten verbreitete moderne Konzept (SSH13). Sie werden auch in der bestehenden Logistikanwendung, in die die Ergebnisse des Forschungsprojektes eingebracht werden sollen, verwendet.
Ein Datenbankkonzept kann auf unterschiedliche Weise implementiert werden. Eine solche Implementierung ist ein *Datenbankmanagementsystem* (DBMS). Ein DBMS ist eine Software zum Anlegen, Verwalten, Bearbeiten und Abfragen einer Datenbank (EN09), (RC95). Vertreter von Managementsystemen für Graphdatenbanken sind Neo4j (Neo17) und OrientDB (Ori17).

Gegenstand der Masterarbeit wird unter anderem die Überführung der im Forschungsprojekt entwickelten multiskalen Datenstruktur in ein Graphdatenbankmodell sein. Ebenso soll das zu entwerfende Graphdatenbankmodell Teile der Datenstruktur der bestehenden Logistikanwendung, in die die Ergebnisse des Projektes in Zukunft integriert werden sollen, abbilden. Um die Güte und Praxistauglichkeit des Modells beurteilen zu können, wurden konkrete Anwendungsfälle für die Graphdatenbank definiert. Diese sollen durch ein zu implementierendes *Application Programming Interface* (API) von der Graphdatenbank unterstützt werden. Anhand der Anwendungsfälle und auf Basis von Testdaten, deren Import von der zu implementierenden Graphdatenbank ebenfalls unterstützt werden soll, können anschließend Testfälle für die Graphdatenbank und die implementierte Schnittstelle definiert werden. Die Ergebnisse der Tests werden die Eignung von Graphdatenbanken zur Verwaltung multiskaler und domänenspezifischer Datenstrukturen in der konkreten Logistikanwendung messbar machen. Die Anwendungsfälle ergeben sich aus drei Kategorien:

Basis-Anwendungsfälle beinhalten grundlegende Operationen, die die Datenbank durchführen können muss. Dazu zählen zum Beispiel das Einfügen und Löschen von Knoten

und Kanten.

Multiskale Anwendungsfälle ergeben sich aus Anforderungen der multiskalen Datenstruktur, die zurzeit mit objektorientierter Programmierung umgesetzt sind.

Logistik-Anwendungsfälle ergeben sich aus Aufgaben, die in der konkreten Logistikanwendung bereits mit objektorientierter Programmierung und einer relationalen Datenbank umgesetzt sind, aber bei einem Wechsel auf eine Graphdatenbank auch von dieser erledigt werden können sollten. Diese Anwendungsfälle dienen der Sicherstellung der Kompatibilität.

Die Anwendungsfälle jeder Kategorie werden im Folgenden näher erläutert. Dabei sind einige als *optional* gekennzeichnet. Optionale Anwendungsfälle sind für eine Beurteilung der Eignung von Graphdatenbanken nicht zwingend erforderlich, da sie entweder eine Erweiterung eines anderen Anwendungsfalls darstellen und daher abschätzbar sind (s. Logistik-Anwendungsfälle) oder nicht den hauptsächlichen Zweck der bestehenden und zu vergleichenden Implementierung betreffen (s. multiskale Anwendungsfälle).

2.1 Basis-Anwendungsfälle

Abbilden relevanter Datenstrukturen Sowohl die multiskale Datenstruktur als auch das Datenmodell der Logistikanwendung sind aus objektorientierter Sicht definiert. In dieser Form können die Datenstrukturen nicht eins zu eins in einer Graphdatenbank abgelegt werden, da eine Graphdatenbank wie beschrieben aus Knoten und Kanten und nicht aus Objekten besteht. Es ist erforderlich die objektorientierte Definition auf ein graphbasiertes Modell abzubilden.
Bei der Abbildung der Datenstrukturen in eine Graphdatenbank sind sowohl für die multiskale Datenstruktur als auch für die Logistikanwendung zunächst die Teile zu identifizieren, die für eine Realisierung aller weiteren Anwendungsfälle relevant sind. Die Implementierung einer entsprechenden Graphdatenbank soll repräsentativ und nicht allumfassend erfolgen. Zudem muss bei der Auswahl des zu implementierenden Modells aus verschiedenen möglichen Ansätzen beachtet werden, welche Anforderungen die weiteren Anwendungsfälle an das Modell stellen.

Elementare Netzwerkoperationen Entsprechend der Grundfunktionalität, die die multiskale Datenstruktur in ihrer objektorientierten Implementierung bietet, sollen Operationen auf der Graphdatenbank implementiert werden, die diese entsprechend für das graphbasierte Modell realisieren. Umgesetzt werden sollen elementare Netzwerkoperationen, die typisch sind für einen Graphen. Dazu zählen das Einfügen und Löschen von Knoten und Kanten in einen Graph sowie das Ermitteln der Gesamtzahl von Knoten oder Kanten im Graph. Des Weiteren sollen knoten- und kantenspezifische Operationen implementiert werden, wie das Ermitteln aller ein- oder ausgehenden Kanten eines Knoten und des Start- und Endknoten einer Kante.

2.2 Multiskale Anwendungsfälle

Speichern von Optimierungsergebnissen/Lösungen In einem Logistiknetzwerk werden Transporte im Hinblick auf ihre Kosten optimiert. Ein im Forschungsprojekt entwickelter Algorithmus berechnet auf Basis der multiskalen Datenstruktur eine Lösung für ein gegebenes Transportproblem. Eine solche Lösung, die einer Menge von Transportaufträgen jeweils einen günstigen Pfad im Netzwerk zuordnet, soll in der zu implementierenden Graphdatenbank gespeichert werden können. Bisher wird im Projekt keine persistente Datenhaltung für Optimierungsergebnisse realisiert. Die dauerhafte Speicherung der Lösungen ist jedoch sinnvoll, weil sie als Ausgangspunkt für die Berechnung neuer Lösungen auf einem leicht veränderten Netzwerk dienen können und sich so spätere Optimierungsprobleme in kürzerer Zeit lösen lassen. Zu diesem Zweck soll die Graphdatenbank außerdem die Anforderung unterstützen, den Graphen aus der Datenbank zu laden, der zu einer bestimmten Lösung gehört, um ihn als Basis für weitere Berechnungen verwenden zu können.
Im weiteren Projektverlauf sollen die Informationen, die der Algorithmus bisher in einer Lösung bereithält, erweitert werden um die Zuordnung von Kosten zu den einzelnen im Netzwerk befindlichen Transportkanten. Auch diese Informationen sollen in der zu implementierenden Graphdatenbank gespeichert werden können.

Zoomen Die multiskale Datenstruktur hält in ihrer im bisherigen Projektverlauf entwickelten objektorientierten Implementierung Funktionalität bereit, um in der vom Optimierungsalgorithmus aktuell betrachteten graphischen Darstellung des Logistiknetzwerkes, der aktuellen Projektion, zu zoomen. Eine Projektion besteht aus Knoten und Kanten unterschiedlicher Granularitätsstufen. An einer bestimmten Stelle in den Graphen hinein zu zoomen bedeutet, einen einzelnen Knoten sowie seine ein- und ausgehenden Kanten zu verfeinern, also stattdessen die zugehörigen Knoten und Kanten auf einer niedrigeren Granularitätsstufe zu betrachten. Umgekehrt bedeutet aus dem Graphen heraus zu zoomen, eine bestimmte Menge Knoten zusammenzufassen, also den aggregierten Knoten und die zugehörigen Kanten auf einer höheren Granularitätsstufe zu betrachten. Um diese Vorgänge realisieren zu können, kann in der multiskalen Datenstruktur von einem Knoten direkt auf den entsprechenden Knoten in der nächst höheren Granularitätsstufe zugegriffen werden. Zudem sind zu jeder aggregierten Kante die Kanten der niedrigsten Granularitätsstufe bekannt, aus denen sich die Kante zusammensetzt. Diese Informationen sollen auch in der Graphdatenbank zur Verfügung stehen. Darüber hinaus soll die Graphdatenbank die aggregierten Tarife zuvor verwendeter Projektionen speichern. Diese werden in der objektorientierten Implementierung jeweils aufwändig neu berechnet, so dass die Datenbank hier möglicherweise zu einem Geschwindigkeitsgewinn führen wird. Außerdem soll es durch die Graphdatenbank möglich sein, zu einer gesamten Projektion diejenigen Projektionen zu speichern, die sich aus der aktuellen durch einen einzelnen Zoomschritt erzeugen lassen.

Speichern sinnvoller Hubs für ein Quelle-Senke-Paar (optional) Bei der Berechnung eines günstigen Pfades im Logistiknetzwerk zur Ausführung eines Transportauftrages mit

einem bestimmten Quelle-Senke-Paar können zunächst anhand verschiedener Kriterien bestimmte Hub-Standorte von vornherein von der Benutzung ausgeschlossen werden. Zu einem Quelle-Senke-Paar existiert also eine Menge von Hub-Standorten, die bei der Berechnung günstiger Routen sinnvollerweise zu betrachten sind. Um diese Informationen nicht für jedes Optimierungsproblem neu berechnen zu müssen, sollen sie in der Graphdatenbank gespeichert werden können.

2.3 Logistik-Anwendungsfälle

Die bestehende Logistikanwendung ermöglicht sowohl operative als auch strategische Transportplanung. Um die Kompatibilität der Implemementierung der Graphdatenbank mit der Logistikanwendung sicherzustellen, wurden zwei Anwendungsfälle der operativen Transportplanung (Finden von Feiertagsanpassungen und Standardtransporten) und zwei Anwendungsfälle der strategischen Transportplanung (Routen und Warenbedarf auf Transportkanten) ausgewählt.

Finden von Standardtransporten In der operativen logistischen Planung sollen für konkrete Transportaufträge passende Routen gefunden und entsprechend Transportmittel gebucht werden, um die Aufträge zu realisieren. Sobald ein neuer Transportauftrag vorliegt, wird zunächst anhand verschiedener Auswahlkriterien eine günstige Route festgelegt. Anschließend werden anhand der Informationen im Transportauftrag zu den einzelnen Standorten, die diese Route betreffen, konkrete Ankunfts- und Abfahrtzeiten an konkreten Tagen berechnet, die von den jeweiligen Transportmitteln eingehalten werden müssen. Sobald diese zeitliche Ausprägung der Route vorliegt, soll festgestellt werden, ob für einzelne Abschnitte bereits ein Transportmittel für den passenden Zeitraum gebucht wurde, das aber noch nicht ausgelastet ist. In diesem Fall wird standardmäßig für den betreffenden Abschnitt der Transportauftrag mit auf dieses Transportmittel gebucht. Das Auffinden jeweils eines passenden Standardtransportes für die einzelnen Abschnitte der zeitlichen Ausprägung einer Route soll durch die Graphdatenbank unterstützt werden.

Finden von Feiertagsanpassungen (optional) Des Weiteren wird in der operativen Planung, sobald die zeitliche Ausprägung der Route für einen Transportauftrag feststeht, in der Regel überprüft, ob sie tatsächlich realisiert werden kann, oder ob sie aufgrund von Feiertagen innerhalb des betreffenden Zeitraumes durch eine alternative zeitliche Ausprägung ersetzt werden muss. In der aktuellen Implementierung muss dazu die berechnete zeitliche Ausprägung mit einer Liste zeitlicher Ausprägungen verglichen werden, die im Vorfeld als von Feiertagen betroffen erkannt wurden. Dieser Vorgang ist aktuell aufgrund der komplexen Datenstruktur einer zeitlichen Ausprägung zeitaufwändig, weshalb er von der Graphdatenbank unterstützt werden soll.
Technisch gesehen können beide Anwendungsfälle aus dem operativen Bereich, im Hinblick auf die bisherige Implementierung der relevanten Datenstruktur in der Logistikanwendung, abstrakt formuliert werden als Problem des effizienten Traversierens durch eine baumartige Datenstruktur. Aufgrund dieser Parallele und des Umstands, dass die betreffende

Datenstruktur in ihrer aktuellen Implementierung für den Anwendungsfall des Auffindens von Standardtransporten um eine Baumebene kleiner ist als für den Anwendungsfall des Auffindens von Feiertagsanpassungen, wird im Zuge der Implementierung der Graphdatenbank zunächst eine Lösung für ersteren Anwendungsfall erarbeitet, um sie später auf den zweiten Anwendungsfall konzeptionell übertragen zu können.

Finden von Routen und Warenrelationen zu einer bestimmten Transportkante In der strategischen Planung von Logistiknetzwerken geht es um längerfristige Entscheidungen und das Simulieren unterschiedlicher Zukunftsszenarien. Zum Beispiel kann es von Interesse sein festzustellen, welche Routen und Lieferbeziehungen betroffen wären, wenn eine bestimmte Transportkante im Netzwerk nicht mehr nutzbar wäre. Eine weitere interessante Information bzgl. einer bestimmten Transportkante ist die Lieferbeziehung, die den größten Anteil an den Gesamtkosten hat, die sich an dieser Kante summieren. Für beide Fragestellungen muss in dem Logistiknetzwerk das Auffinden von Routen und Warenrelationen zu einer bestimmten Transportkante möglich sein. Diese Funktionalität soll von der Graphdatenbank unterstützt werden.

Berechnung des Warenbedarfs einer Transportkante (optional) Die Implementierung des vorangegangenen Anwendungsfalls ist außerdem Voraussetzung für die Umsetzung eines weiteren Anwendungsfalls in der strategischen Logistikplanung. Aktuell werden in der Logistikanwendung zu jeder einzelnen Transportkante Informationen darüber gespeichert, welche Waren in welcher Menge in welchem zeitlichen Rhythmus über diese Kante transportiert werden, der sog. Warenbedarf. Diese Informationen sind fachlich von großem Interesse, allerdings verbrauchen sie in realen Datensätzen von Logistiknetzwerken einen Großteil des Speicherplatzes, verglichen mit der übrigen Datenstruktur in der Logistikanwendung. Für die Graphdatenbank ergibt sich daher der Anwendungsfall diese Daten ad hoc berechnen zu können. Für diese Berechnung ist wiederum die Aggregation der Informationen der zur betrachteten Transportkante gehörenden Warenrelationen erforderlich, wie im vorangegangenen Anwendungsfall beschrieben.

3 Vorgehen

Zur Erreichung der im vorangegangenen Kapitel definierten Zielstellung wurden konkrete Arbeitsschritte definiert. Sie sind als Meilensteine mit relevanten Zwischenergebnissen anzusehen.

3.1 Vorbereitung

Zunächst sollten geeignete Kandidaten für Anwendungsfälle für eine Graphdatenbank gesammelt, priorisiert, ausgewählt und konkretisiert werden, wie bereits erfolgt. Im zweiten Schritt werden die objektorientierten Implementierungen der bestehenden Logistikanwendung und der multiskalen Datenstruktur modelliert. Die Modelle sollen sich auf die für die ausgewählten Anwendungsfälle relevanten Teile der Datenstrukturen beschränken.

Weiterhin werden Parallelen zwischen den einzelnen Modellen identifiziert, um anschließend ein gemeinsames Graphdatenbankmodell zu erstellen. Bei der Überführung sind eventuelle Einschränkungen zu dokumentieren. Im nächsten Schritt werden auf Basis der Anwendungsfälle konkrete Testfälle und Anforderungen an Testdaten definiert. Ebenso sind auf Basis der Anwendungsfälle Anforderungen an das später verwendete Graphdatenbankmanagementsystem festzulegen. Außerdem ist diverse Fachliteratur für diese Aufgabe zu Rate zu ziehen. Weiterführend wird aus in Frage kommenden Kandidaten für das später verwendete Graphdatenbankmanagementsystem eine Vorauswahl von maximal drei Implementierungen getroffen. Anschließend wird jede Software im Hinblick auf die definierten Anforderungen getestet und eine Evaluationsmatrix erstellt, auf deren Basis eine begründete Empfehlung vorgenommen wird.

3.2 Umsetzung

Liegt ein konkretes Graphdatenbankmanagementsystem vor soll unter Verwendung der zugehörigen Abfragesprache eine prototypische Implementierung des entworfenen Graphdatenbankmodells und des Anwendungsfall-unterstützenden API inklusive Funktionalität für den Datenimport entwickelt werden. Dabei wird der Teil der Schnittstelle für die multiskalen Anwendungsfälle in C++ (C++17) und der Teil für die Logistik-Anwendungsfälle in Java (Jav17) implementiert, um Kompatibilität mit der bestehenden Implementierung der multiskalen Datenstruktur bzw. der Logistikanwendung zu gewährleisten. Vor der Implementierung der Schnittstelle werden allerdings die vorbereiteten Testfälle implementiert und ggf. synthetische Testdaten erzeugt, sofern die im Forschungsprojekt und von der bestehenden Logistikanwendung verwendeten Daten nicht allen Anforderungen genügen.

3.3 Evaluation

Zur Bewertung der Güte der entwickelten Implementierung werden die Testfälle auf der Graphdatenbank ausgeführt. Zuvor ist die Datenbank mit den jeweils zugehörigen Testdaten zu füllen. Außer den Testdaten definiert ein Testfall seine Eingabedaten, die durch ihn abgedeckte Fragestellung und die erwartete Ausgabe. Jeder Testfall wird unter verschiedenen Bedingungen durchgeführt. Variiert wird die absolute obere Schranke für den durch den Testfall gleichzeitig belegbaren Arbeitsspeicher. Für jede Variante wird die Laufzeit der Ausführung und der Verlauf der tatsächlichen Auslastung des Arbeitsspeichers gemessen. Ebenso werden die Testfälle auf der bestehenden Implementierung der multiskalen Datenstruktur bzw. in der bestehenden Logistikanwendung ausgeführt und dokumentiert. Um Graphdatenbanken als geeignet und darüber hinaus vorteilhaft für die definierten Anwendungsfälle befinden zu können, sollte ab einer festzulegenden nutzbaren Arbeitsspeichergröße die Laufzeit der Testfälle auf der Graphdatenbank signifikant geringer sein als für die verglichene Datenhaltung. Um Messfehlern vorzubeugen und verlässliche, repräsentative Messwerte zu erhalten, wird jede Variante jedes Testfalls mehrfach durchgeführt.
Außer der Laufzeit wird, unabhängig von der Variante, für jeden Testfall der betroffene Teil

der Datenstruktur identifiziert. Das sind die Knoten und Kanten in der Graphdatenbank, die potenziell betrachtet werden. So soll in Relation zu den Testdaten die für den Testfall benötigte Datenbankgröße, also die Größe des belegten Festplattenspeichers, bestimmt werden. Dieselben Betrachtungen werden für die relationale Datenbank der bestehenden Logistikanwendung durchgeführt. Da in der bisherigen Implementierung der multiskalen Datenstruktur keine persistente Datenhaltung vorliegt, können hier keine Vergleichswerte ermittelt werden. Insgesamt ist jedoch zu berücksichtigen, dass die Graphdatenbank weder die multiskale Datenstruktur noch die domänenspezifische Datenstruktur aus der bestehenden Logistikanwendung in vollem Umfang abbildet. Um auch unter diesem Aspekt Graphdatenbanken als geeignet für die definierten Anwendungsfälle befinden zu können, sollte die Graphdatenbank keine signifikant größere Menge Festplattenspeicher verbrauchen als die relationale Datenbank.

Literatur

[C++17] C++: *C++*. http://www.cplusplus.com/. Version: 2017. – [Online; aufgerufen am 15. Dezember 2017]

[EN09] Elmasri, Ramez A. ; Navathe, Shamkant B.: *Grundlagen von Datenbanksystemen*. 3., aktualisierte Aufl., Bachelorausg. München [u.a.] : Pearson Studium, 2009 (Pearson Studium). – ISBN 978–3–86894–012–1

[HMV17] Hall, Richard J. ; Murray, Christopher W. ; Verdonk, Marcel L.: The Fragment Network: A Chemistry Recommendation Engine Built Using a Graph Database. In: *Journal of Medicinal Chemistry* 60 (2017), July, Nr. 14. – ISSN 1520–4804

[HVK16] Hochgeschwender, Nico ; Voos, Holger ; Kraetzschmar, Gerhard K.: Towards Persistent Storage and Retrieval of Domain Models using Graph Database Technology. (2016), February

[Jav17] Java: *Java*. https://www.java.com/de/. Version: 2017. – [Online; aufgerufen am 15. Dezember 2017]

[Neo17] Neo4j: *Neo4j*. https://www.neo4j.com. Version: 2017. – [Online; aufgerufen am 30. November 2017]

[Ori17] OrientDB: *OrientDB*. https://www.orientdb.com. Version: 2017. – [Online; aufgerufen am 30. November 2017]

[RC95] Rob, Peter ; Coronel, Carlos: *Database systems : design, implementation, and management*. 2. Aufl. Danvers [u.a.] : Boyd & Fraser, 1995. – ISBN 0–7895–0052–3

[RWE13] Robinson, Ian ; Webber, Jim ; Eifrem, Emil: *Graph Databases*. 1. Aufl. Beijing [u.a.] : O'Reilly Media, 2013. – ISBN 978–1–449–35626–2

[SSH13] Saake, Gunter ; Sattler, Kai-Uwe ; Heuer, Andreas: *Datenbanken - Konzepte und Sprachen*. 5. Aufl. Heidelberg, München, Landsberg [u.a.] : mitp, 2013. – ISBN 978–3–8266–9453–0

[VW15] Vukotic, Aleksa ; Watt, Nicki: *Neo4j in action*. Shelter Island : Manning, 2015. – ISBN 978–1–617290–76–3

[YKK17] Yoon, Byoung-Ha ; Kim, Seon-Kyu ; Kim, Seon-Young: Use of Graph Database for the Integration of Heterogeneous Biological Data. In: *Genomics & Informatics* 15 (2017), March, Nr. 1, 19–27. https://doaj.org/article/b1b0d931fe4048cc9bce760b765e3f68. – ISSN 1598–866X