Optimierung der Datenintegration und Analyse hochteurer Medikamente im SwissDRG-System

Bachelor-Thesis

Studiengang:	Medizininformatik
Autor:	Christian Franke
Betreuer:	Murat Sariyar
Auftraggeberin:	SwissDRG AG
Experte:	Thierry Hafner
Datum:	09. Juni 2025

Berner Fachhochschule

Departement Technik und Informatik

Management Summary

Die SwissDRG AG ist eine gemeinsame Institution der Leistungserbringer, Versicherer und Kantone im Schweizer Gesundheitssystem. Sie trägt die Verantwortung für die Weiterentwicklung der stationären Tarifstrukturen. Im Bereich hochteurer Medikamente werden bei Bedarf Zusatzentgelte eingeführt. Trotz steigender Anzahl hochteurer Medikamente müssen deren Zusammenhänge im Blick behalten werden, um Fehlanreize so gut wie möglich auszuschliessen. Dies ist bisher nur mit hohem manuellem Aufwand möglich.

Öffentlich verfügbare Informationen über hochteure Medikamente, deren Präparate und Indikationen sollten in eine Graph-Datenbank vereint werden. Nach der Anforderungserhebung sind die Datenquellen mit R und Python aufbereitet worden. Das Datenmodell wurde zunächst mit arrows.app graphisch skizziert. Mittels Cypher-Code sind im Anschluss die Knoten und Relationen für Neo4j 5.26.1 auf einer virtuellen Maschine der BFH und einer Cloud-Umgebung erstellt worden.

Die Medikamentendaten wurden in 13 Knoten-Typen (Labels) gespeichert, die mit 14 unterschiedlichen Relationen-Typen miteinander verbunden sind. Die Daten stammen aus mehreren öffentlichen Quellen (SwissDRG AG, Europäischen Arzneimittelagentur, Stiftung Refdata, Bundesamt für Gesundheit und WHO Collaborating Centre for Drug Statistics Methodology). Es wurden über 27'000 Knoten und über 41'000 Relationen kreiert. Mit dem Medikamenten-Graphen können alle Substanzen mit «gleicher» Indikation gefunden werden. Zudem lassen sich vielfältige Informationen über ein einzelnes Medikament oder eine Substanz visualisieren. Mit dem Low-Code Tool Neo4j Bloom wurden Abfragen erstellt, um individuelle Analysen mit Drill-Down, Filter und Slicing durchführen zu können.

NoSQL-Datenbanken im Allgemeinen und Graph-Datenbanken im Speziellen ermöglichen flexible Datenmodelle. Insbesondere für die Knoten «Substanz» und «Produkt» hat die Schemalosigkeit Vorteile gegenüber relationalen Systemen, weil Daten aus diversen Quellen und teilweise unterschiedlichen Datenformaten integriert und verbunden werden können. Die Relation «HAT_ATC» wird z.B. auch dann erstellt, wenn «Substanz» und «Produkt» zwar andere ATC-Codes haben, aber der ATC-Code der «Spezialitätenliste» mit dem ATC-Code der «Substanz» identisch ist. Solch indirekte Relationen sind in Graph-Datenbanken (einfach) möglich. Defizitäre Datenqualität der Quelldaten kann damit teilweise ausgeglichen werden.

Das Graph-Modell ermöglicht nun einen Überblick über die Medikamente und deren Zusammenhänge. Es wird zunächst ohne Risiko und Kosten in der Neo4j Cloud-Umgebung (Aura) im Alltag getestet. Danach kann entschieden werden, ob und wenn ja welche Graph-Datenbank bei der SwissDRG AG eingeführt wird, um weitere (interne) Daten in das Modell zu integrieren. Trotz der vielen flexiblen Möglichkeiten werden Graph-Datenbanken die etablierten relationalen Datenbanksysteme zwar nicht ersetzen, aber sie bieten interessante Lösungen für spezifische Use-Cases. Stark vernetzte Medikamentendaten im SwissDRG-System sind ein solcher Use-Case.

Inhaltsverzeichnis

1 Einleitung

Diese Thesis ist von Christian Franke im achten Semester des Bachelorstudiengangs Medizininformatik an der Berner Fachhochschule (BFH) verfasst worden. Das Projekt ist zwischen Februar und Juni 2025 bearbeitet worden. Die Auftraggeberin dieser Arbeit ist die SwissDRG AG und persönlich Frau Isabell Helms als Mitarbeiterin des Geschäftsbereiches Akutsomatik. Frau Helms ist unter anderem für die «Liste der hochteuren Medikamente/Substanzen» zuständig. Der Autor, Christian Franke, ist ebenfalls bei der SwissDRG AG im Geschäftsbereich Akutsomatik angestellt.

Die SwissDRG AG ist eine gemeinsame Institution der Leistungserbringer, Versicherer und Kantone im Schweizer Gesundheitssystem. Sie trägt die Verantwortung für die Einführung, Weiterentwicklung und Pflege der stationären Tarifstrukturen. Im Bereich hochteurer Medikamente und Substanzen werden bei Bedarf Zusatzentgelte eingeführt. Die Anzahl dieser Medikamente und Zusatzentgelte ist in den letzten Jahren kontinuierlich gestiegen. Die Verwaltung der relevanten Daten erfolgt in spezialisierten Web-Applikationen, die jedoch als «Datensilos» betrachtet werden können. Eine Verknüpfung interner Daten sowie deren Anreicherung mit externen Informationen und die Analyse der Zusammenhänge sind bislang nur eingeschränkt oder manuell möglich.

Im Rahmen dieser Bachelorarbeit sollen die Zusammenhänge und Abhängigkeiten der Informationen zu hochteuren Medikamenten mit Hilfe einer Graph-Datenbank dargestellt und analysiert werden. Ziel ist es, die bestehenden «Datensilos» zu verknüpfen, wobei Informationen genutzt werden, die von der SwissDRG AG oder anderen relevanten Institutionen veröffentlicht wurden. Folgende Aufgaben sind zu adressieren:

- Überblick über bestehende Lösungen und vergleichbare Anwendungsszenarien für Graph-Datenbanken.
- Erhebung der Anforderungen für die Speicherung, Integration und Analyse von Daten zu hochpreisigen Medikamenten und entsprechenden Zusatzentgelten.
- Entwicklung eines Datenmodells basierend auf den Anforderungen unter Verwendung von Ontologie-Konzepten.
- Auswahl eines Produkts, das die definierten Anforderungen am besten erfüllt.
- Implementierung des Datenmodells auf einem Graph-Datenbank-Server und Befüllung der Datenbank mit relevanten Daten.
- Evaluation der prototypischen Lösung anhand der definierten Analyse-Anforderungen.
- Definition von weiteren Einsatzmöglichkeiten von Graph-Datenbanken für die SwissDRG AG.

Der Betreuer der Thesis ist Herr Murat Sariyar und der zuständige Experte Herr Thierry Hafner. Dieser arbeitet bei der Metamedic GmbH in Gipf-Oberfrick.

In diesem Bericht wird aus sprachlichen Gründen das generische Maskulinum verwendet, sofern es sich nicht um spezifische natürliche oder juristische Personen handelt.

2 Grundlagen

2.1 SwissDRG und Medikamente

Die SwissDRG AG wurde im Jahr 2008 gegründet und «hat die gesetzliche Aufgabe, die Tarifstrukturen des Fallpauschalensystems zu pflegen und weiterzuentwickeln» [1]. Seit 01. Januar 2012 werden stationäre Spitalbehandlungen als Fallpauschale mit der Tarifstruktur SwissDRG abrechnet [2]. Zudem wurden im Jahr 2018 die Tarifstruktur TARPSY für die stationäre Psychiatrie und im Jahr 2022 die Tarifstruktur ST Reha für die stationäre Rehabilitation eingeführt [1,2].

Mit den Fallpauschalen sowie Tagespauschalen werden Krankheitsbilder, die medizinisch und ökonomisch einen ähnlichen durchschnittlichen Behandlungsaufwand widerspiegeln, schweizweit einheitlich abgerechnet. Dabei werden die berechneten Kostengewichte mit den verhandelten Baserates multipliziert und je nach Fall um spezifische Zusatzentgelte für hochteure Medikamente, Verfahren oder Implantate ergänzt [1,3,4]. Die Anzahl der ATC-Codes in der «Liste der hochteuren Medikamente/Substanzen» hat sich von 116 Substanzen im Jahr 2016 [5] auf 329 ATC-Codes im Jahr 2025 [6] mehr als verdoppelt. Die Zahl der ATC-Codes im Zusatzentgeltkatalog stieg von 58 für die Version 7.0 (gültig im Jahr 2018) [7] auf 143 für die Version 14.0 (gültig im Jahr 2025) [3]. Dies entspricht einem Anstieg von nahezu 150%. Die «Liste der hochteuren Medikamente/Substanzen» hiess bis 2023 «Liste der in der medizinischen Statistik erfassbaren Medikamente/Substanzen» [5].

Die «Liste der hochteuren Medikamente/Substanzen» ist je ATC-Code aggregiert [8]. Sie enthält die Bezeichnungen der ATC-Codes, die relevanten Tarifstrukturen, Zusatzangaben, Einschränkungen, die Verabreichungsarten, die zu kodierende Einheiten und das Jahr, seit dem eine Substanz erfasst werden muss [9]. Die Medikamenten-Zusatzentgelte verwenden die gleichen Informationen und sind um Preise und Dosisklassen angereichert [3]. Zudem sind gewisse Details im jährlich aktualisierten «Technischen Begleitblatt» als PDF- oder Excel-Datei spezifiziert [5,6].

Für die Verwaltung der Medikamente sowie Zusatzentgelte werden von der SwissDRG AG selbstentwickelte Web-Applikationen eingesetzt, die die oben genannten Informationen enthalten. Die Web-Applikationen verfügen über keine direkte gemeinsame Schnittstelle. Es gibt jeweils die Möglichkeit, Daten zu exportieren. Die exportierten Daten entsprechen im Wesentlichen den auf der Website der SwissDRG AG veröffentlichten Dokumenten. Die «Liste der hochteuren Medikamente/Substanzen» stellt die Grundlage für die Etablierung von Zusatzentgelten dar. Für neue Medikamente/Substanzen kann jeweils im Sommer eines Jahres ein Antrag zur Aufnahme auf diese Liste gestellt werden [10].

Die Analyse und Verknüpfung der Antragsdaten, der Daten aus den Web-Applikationen und weiteren Datenquellen ist nur mit manuellem Aufwand möglich. Es ist äusserst herausfordernd bei steigender Zahl an Medikamenten den Überblick über die Zusammenhänge der diversen Informationen zu behalten. Um die Vielzahl an Datenquellen besser zu überblicken und einzelne Medikamente sowie Medikamentengruppen effizient beurteilen zu können, wurde dieses Projekt initiiert. Dabei wird insbesondere geprüft, ob Graph-Datenbanken die zusammenhängenden Informationen besser abbilden können als relationale Systeme.

2.2 Graph-Datenbanken

2.2.1 Definition

Graph-Datenbanken gehören zu den nichtrelationalen Datenbanken, die auch NoSQL Datenbanken genannt werden [11]. Graphen sind besonders dazu geeignet, Zusammenhänge zwischen heterogenen Informationsobjekten unter anderem in den Bereichen Biologie und Medizin abzubilden [12]. Graphen bestehen grundsätzlich aus Knoten (Nodes, Vertices), die über Kanten (Edges, Relations) miteinander verbunden sind. Die Kanten stellen die Relation zwischen den Knoten dar [13,14].

Graph-Datenbanken sollen traditionelle, relationale Datenbanken nicht ablösen, sondern können diese für spezifische Aufgaben oder Fragestellungen ergänzen [15], denn Graphen haben andere Vorteile als relationale Datenbank-Managementsysteme (RDBMS). Bei der Datenmodellierung in RDBMS werden für jede Tabelle eindeutige Primary-Keys erstellt und mit anderen Tabellen über Fremdschlüssel (Foreign Keys) verknüpft. Join-Operationen in SQL verknüpfen die Tabellen und sind in vielen Anwendungsfällen etabliert. Wenn jedoch viele «Many-to-Many-Relations» vorhanden sind, wird die Erstellung der Joins aufwändiger und die Abfragegeschwindigkeit reduziert sich [16]. In solchen Situationen können Graph-Datenbank-Managementsystemen (GDBMS) vorteilhaft sein. Hier können von einem Knoten die verbundenen Knoten direkt angesteuert werden können, ohne nicht relevante Daten abfragen zu müssen [12]. Den Abfragestil in Graphen nennt man Traversal [17]. Ausgehend von z.B. einem einzelnen ATC-Code in der SwissDRG «Liste der hochteuren Medikamente/Substanzen» wären diverse Traversals zu weiteren Informationen möglich.

Neben Traversals haben Graphen den Vorteil, dass die «Many-to-Many-Relations» ohne zusätzliche Zwischentabellen auf natürlichsprachliche Art und Weise modelliert werden können. Die Graph-Abfragesprachen zeichnen sich durch eine intuitive Verknüpfung der Knoten und Kanten aus, sodass Fachexperten ohne spezifische SQL-Kenntnisse relativ schnell Abfragen verstehen und selbst erstellen können [16]. Graphen sind, wie auch andere NoSQL Systeme, grundsätzlich schemalos und bieten sich somit an, vielfältige biologische oder medizinische Zusammenhänge zu modellieren [18]. Graphen unterscheiden man in zwei Typologien: RDF und Property Graph [14,19].

RDF steht für Resource Description Framework und ist ein in den 1990er Jahren eingeführter W3C Standard, der den Datenaustausch und die Wissensrepräsentation im semantischen Web beschreibt und eng mit Ontologien zusammenhängt (siehe Abschnitt 2.2.3). Es handelt sich dabei um ein formales Model, das aus Triples von Subjekt, Prädikat und Objekt besteht. Subjekt und Prädikat werden dabei durch eindeutige URI repräsentiert und Objekte können entweder URI oder konkrete Werte (Literals) annehmen. Subjekt und Objekt sind Knoten und das Prädikat ist die Kante bzw. Relation im RDF-Modell. RDF-Tripels können mit unterschiedlichen Formaten wie z.B. XML, Turtle oder JSON gespeichert werden [20]. Die Daten können mit SPARQL abgefragt und traversiert werden, wobei SPARQL einen ähnlichen Aufbau wie SQL mit SELECT, FROM und WHERE Statements hat [21].

Property Graphs sind ebenfalls aus Knoten und Kanten aufgebaut, die jedoch nicht dem Triple-Schema folgen. Stattdessen besitzen Knoten und Relationen eigene Properties (Eigenschaften) in Form von Key-Value-Paaren. Knoten können ein oder mehrere Labels haben, die sie mit einer «Grundausprägung» kennzeichnen. Deshalb spricht man in diesem Zusammenhang von Labeled Property Graph (LPG). Die Nodes können ein oder mehrere Properties als Key-Value-Paare aufweisen. Relationen im Property Graph sind gerichtet und können ebenfalls über Properties verfügen. Properties können die Knoten und Kanten mit vielfältigen Informationen anreichern, ohne neue Triples wie im RDF-Modell erstellen zu müssen. Knoten können über mehrere, auch gleichartige, Relationen miteinander verbunden sein [14,19,22].

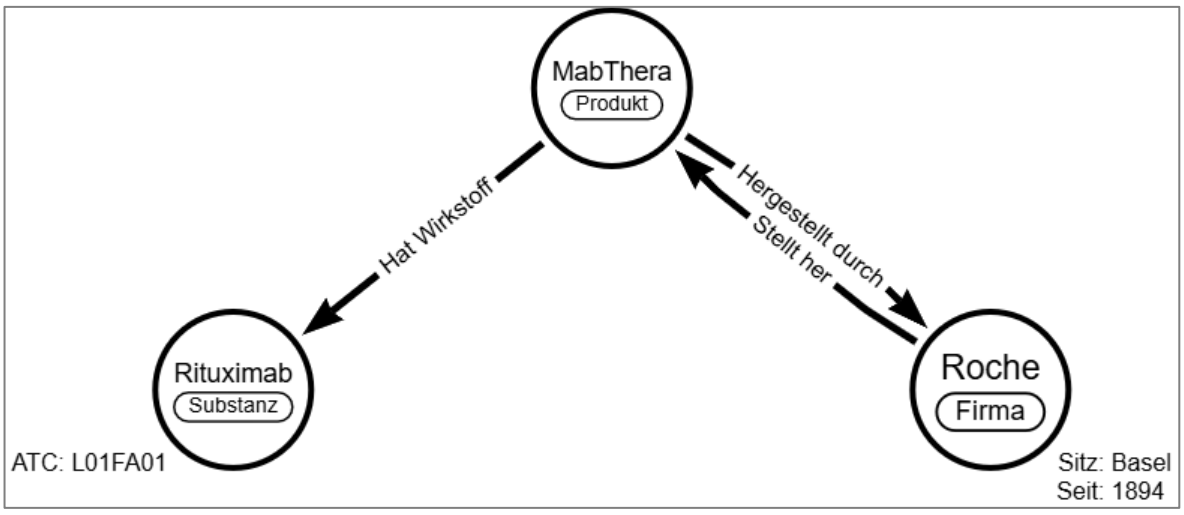


Abbildung 1: Beispiel eines Property Graph

Im beispielhaften Property Graph in Abbildung 1 werden die Labels Substanz, Produkt und Firma modelliert. Die Labels könnte man im übertragenen Sinne als «Tabelle» im relationalen Modell interpretieren. Die drei kreisförmig dargestellten Knoten (Rituximab, MabThea und Roche) entsprechen jeweils «einer Zeile» im RDBMS. Die Properties (ATC L01FA01 für Knoten Rituximab, Sitz Basel und «Seit 1984» für Roche) können als unterschiedliche «Spalten» in RDBMS verstanden werden. Die Relationen werden mit Pfeilen dargestellt und könnten als «Join» angesehen werden. Die Abbildung 2 verdeutlicht dieses Beispiel für das relationale Modell.

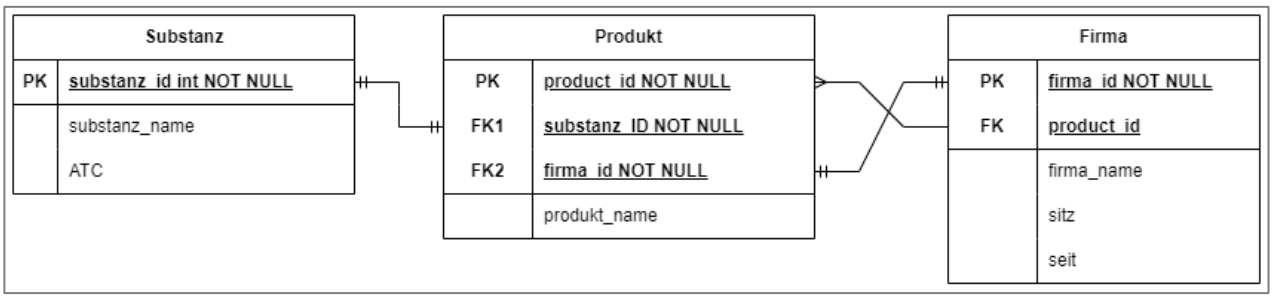


Abbildung 2: Beispiel des Property Graph als ER-Diagramm

Graphen können, je nach GDBMS, mit einer oder mehrere Abfragesprachen erstellt und ausgewertet werden. Bei RDF kommt häufig SPARQL zum Einsatz. Für Property Graphs wird oft Cypher/openCypher sowie GQL oder Gremlin verwendet. Diese Unterteilung ist jedoch nicht disjunkt. Es sind auch Kombinationen sowie weitere Sprachen wie AQL, GraphQL, MQL, nGQL, PGQL oder SQL sowie APIs zur Nutzung von Java, Python etc. im Einsatz (siehe Tabelle 1 in Abschnitt 2.2.2).

2.2.2 Beispiele für Graph-Datenbanken

Die bekannteste und sehr weit verbreitete Graph-Datenbank ist Neo4j [23,24]. Dabei handelt es sich um eine native, Java-basierte Graph-Datenbank, die von Neo4j Inc. entwickelt wurde und noch heute weiterentwickelt wird. Sie ist als Community, Enterprise oder Cloud-Edition verfügbar. Die Community-Edition ist Open-Source und kostenlos. Für die Enterprise-Version wird eine extra Lizenz benötigt [25]. Die Cloud-Lösung namens Aura ist als Free-Edition mit einer einzigen Datenbank-Instanz oder als kostenpflichtige Lösung für den produktiven Einsatz von skalierbaren Applikationen im Unternehmenskontext vorhanden [26]. Neo4j hat die Abfragesprache Cypher entwickelt, die als openCypher Variante in den ISO GQL Standard eingeflossen ist [27,28].

Es existieren weitere Graph-Datenbanken, die unterschiedliche technologische und ökonomische Ansätze verfolgen oder teilweise vereinen (z.B. RDF, openCypher, Apache TinkerPop, Gremlin, native oder Multi-Model Architekturen oder Nutzung anderer DBMS, API und Driver für diverse Programmiersprachen, Apache 2 Lizenz, Cloud (Free/Premium) etc.). Die folgende Zusammenstellung listet aktuelle Graph-Lösungen in alphabetischer Reihenfolge auf [16,23,29–31] und enthält die Ergebnisse eigener Recherchen. Datenbanken mit Sternchen * hinter dem Namen sind auf Docker Hub als «Docker Official Image» oder mit ** als anderes Image verfügbar.

Tabelle 1: Graph-Datenbanken und einige ihrer Besonderheiten (Long-List)

Name	Website	Abfragesprachen	Hinweis
Aerospike*	https://aerospike.com/products/graph-database	Gremlin	Community Edition vorhanden (Free)
Amazon Neptune	https://aws.amazon.com/de/neptune	SPARQL, Gremlin, openCypher	Nur Cloud
AnzoGraph DB**	https://docs.cambridgesemantics.com/anzograph/v2.5/userdoc/features.htm	SPARQL, Cypher	Free Edition vorhanden

ArangoDB*	https://www.arangodb.com	ArangoDB Query Language (AQL)	Free nur für 14 Tage
ArcadeDB**	https://arcadedb.com	GraphQL, Gremlin, openCypher, SQL	Kostenlos
AzureCosmosDB	https://azure.microsoft.com/de-de/products/cosmos-db	Gremlin	Nur Cloud
Blazegraph**	https://blazegraph.com	SPARQL, Gremlin	Kostenlos
Cayley**	https://cayley.io	Gizmo (Gremlin), MQL, GraphQL	Kostenlos
Dgraph	https://dgraph.io	GraphQL	Nur Cloud
DSEGraph	https://www.datastax.com/products/datastax-graph	Gremlin	Nur Cloud
FaunaDB	https://fauna.com	n/a	Nur Cloud, Free Plan mit Limits
GraphBase	https://graphbase.ai	n/a	Nur Cloud
GraphDB**	https://www.ontotext.com/products/graphdb	SPARQL	Free Edition vorhanden
Huawei Cloud Graph Engine	https://www.huaweicloud.com/intl/en-us/product/ges.html	Gremlin, Cypher	Nur Cloud
JanusGraph**	https://janusgraph.org	Gremlin	Kostenlos
Memgraph**	https://memgraph.com	Cypher	Community Edition vorhanden (Free)
NebulaGraph**	https://www.nebula-graph.io	nGQL, openCypher	Open-source Edition vorhanden (Free)
Neo4j*	https://neo4j.com	Cypher	Community Edition vorhanden (Free)
Oracle Integrated Graph Database	https://www.oracle.com/database/integrated-graph-database	SQL, PGQL, SPARQL	Nur bei Nutzung einer Oracle Database 23ai, 21c oder 19c
OrientDB*	https://www.orientdb.dev	Gremlin, adaptiertes SQL	Community Edition vorhanden (Free)
PuppyGraph**	https://www.puppygraph.com	Cypher, Gremlin	Developer (Free)
RedisGraph**	https://redis.io/blog/introducing-redisgraph-2-0/	Cypher	n/a
Spanner Graph	https://cloud.google.com/spanner/docs/graph?hl=de	GQL, SQL	Nur Cloud
Sparksee/DEX	http://www.sparsity-technologies.com	Nur API für Java, C#, C++, Python, Objective-C	Free nur für 1 Monat oder für Hochschulen
TerminusDB**	https://terminusdb.com/products/terminusdb	GraphQL	Kostenlos
TigerGraph**	https://www.tigergraph.com	GSQL, openCypher	Community Edition vorhanden (Free)
TypeDB**	https://typedb.com	TypeQL	Open-source Edition vorhanden (Free)
VelocityDB	https://velocitydb.com	n/a	Free nur für 30 Tage
Virtuoso**	https://virtuoso.openlinksw.com	SPARQL	Free Trail mit unklarer Dauer

Viele Anbieter bewerben ihre Lösung im Zusammenhang mit generativer KI, weil Graphen mit «GraphRAG» zu akkurateren Antworten gelangen können und diese besser nachvollziehbar seien [32]. Zusätzlich gibt es spezifische Datenbanken oder Tools, um Ontologien zu verwalten [33].

2.2.3 Ontologien

Ontologien und Graph-Datenbanken sind thematisch miteinander verwandt, da jeweils Wissen in Form von Netzwerken abgebildet werden kann. In der Informationstechnologie versteht man unter Ontologien «a formal, explicit specification of a shared conceptualization» [34]. Studer et al. erläutern diese Definition wie folgt:

- Formal: Eine Ontologie muss verständlich für Maschinen sein.
- Explicit: Die Konzepte und deren Einschränkungen müssen explizit definiert sein. Es sollte dabei kein implizites Wissen geben, sonst kann das Wissen nicht vollständig durch Maschinen verarbeitet werden (bzw. es fehlen relevante Informationen).
- Shared: Es existiert ein Konsens über eine Ontologie, die öffentlich zur Verfügung steht.
- Conceptualization: Ein abstraktes Modell eines realen Phänomens innerhalb einer Domäne (Fachbereich) wird erstellt und relevante Konzepte (Klassen), Verbindungen und Einschränkungen werden identifiziert [34,35].

Das World Wide Web Consortium (W3C) hat für Ontologien einen eigenen Standard erstellt: Die Web Ontology Language (OWL). OWL ist Bestandteil des «Semantic Web Tech Stack», in dem unter anderem auch RDF und SPARQL definiert sind. Hier schliesst sich der Kreis zu den Graph-Datenbanken. Für Ontologien wird typischerweise das RDF-Modell eingesetzt, sodass die Tripels mit Turtle, RDF/XML etc. persistiert und als Graph visualisiert werden können. Dafür kommen neben spezifischen Modellierungstools eben auch Graph-Datenbanken zum Einsatz, in denen RDF und deren Schemata (RDFS) gespeichert werden und mit SPARQL abgefragt werden können [33]. Der Ontology Lookup Service (OLS) nutzt für die Zusammenstellung von über 270 Ontologien eine Neo4j Datenbank [36].

Ontologien können sehr umfangreich mit einer formalen Logik und Axiomen modelliert werden, um z.B. Ein- oder Ausschlüsse von Instanzen, deren Klassen und Subklassen abzubilden [37]. Snomed CT ist zwar gemäss Eigendarstellung «nur» eine Terminologie [38] und keine Ontologie, aber es enthält Axiome und weitere Formalien. Beispielsweise ist «Intravenous route» eine Subklasse von «Intravascular route», was man an den folgenden beiden Abbildungen (gemäss https://browser.ihtsdotools.org) erkennt.

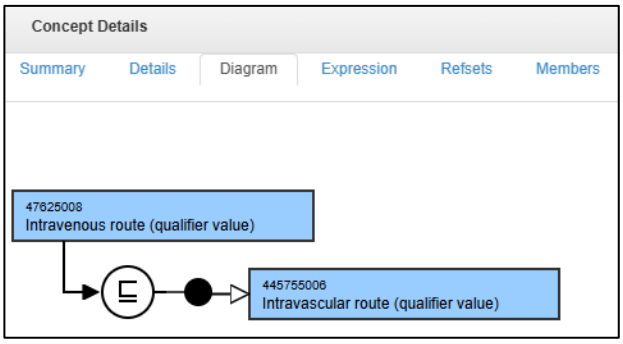


Abbildung 3: Diagrammdarstellung in Snomed CT

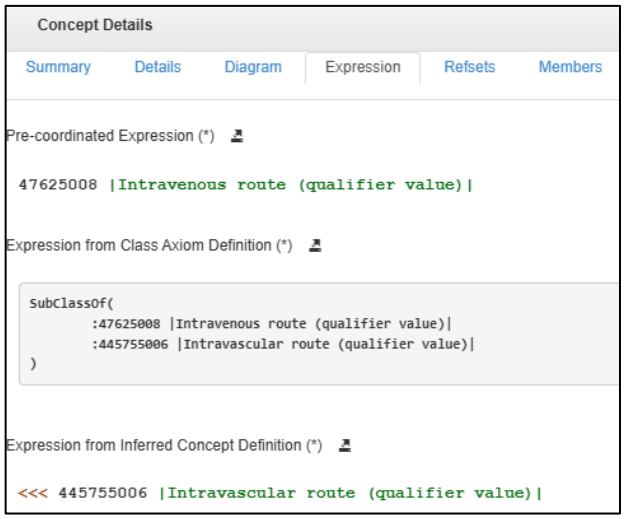


Abbildung 4: Expressions in Snomed CT

Mit Hilfe von Ontologien lassen sich IT-Systeme erstellen, die «die Welt verstehen» und die reale und digitale Welt somit enger miteinander verbinden können. Ontologien sind bereits in den frühen 1990er Jahren mit künstlicher Intelligenz in Verbindung gebracht worden [35]. In der heutigen Zeit haben Ontologien im Zusammenhang mit den Knowledge Graphs und GraphRAG (Graph Retrieval Augmented Generation) eine noch grössere Bedeutung für das «Trendthema AI» erhalten [32]. Daneben können Ontologien eingesetzt werden, um die semantische Interoperabilität zu verbessern [39].

2.2.4 Anwendungen, die auf Graphen und Ontologien aufbauen

Timón-Reina et al., Yoon et al. und Walke et al. geben in ihren Veröffentlichungen aus den Jahren 2017 bis 2024 einen umfangreichen Überblick über vielfältige Anwendungen von Graph-Datenbanken (inkl. Neo4j [36]) im medizinischen Umfeld. Die Anwendungen betreffen insbesondere Medikamente/Interaktionen, Proteine, Metabolismus, Genetik, Diagnostik von Erkrankungen insb. im Bereich der Onkologie, Covid-19, Mikrobiologie und Ontologien. Die Anwendungen haben oft eine Gemeinsamkeit: Graphen werden eingesetzt, um stark vernetzte und miteinander verbundene Daten darzustellen und zu analysieren [14,16,18].

Ein einzelnes Beispiel einer Graphen-Anwendung wird im Folgenden vorgestellt. Bei Orphanet (www.orpha.net) handelt es sich um ein Portal über seltene Erkrankungen sowie der Diagnose, Pflege und Behandlung von Patienten mit seltenen Erkrankungen. Die nicht-kommerzielle Initiative wurde 1997 vom «Institut national de la santé et de la recherche médicale» (INSERM) gegründet und entwickelte sich zu einem Netzwerk in 40 Ländern [40]. Nebst einer API wurde ein Graph-Visualisierungstool entwickelt, das unter

```
dataviz.orphacode.org
```

geöffnet werden kann. Dort kann mit ORPHAcode, ICD-10, ICD-11, OMIM (genetischen Phenotypen) oder Freitext nach Erkrankungen gesucht werden. Resultate werden mit äusserst kurzer Latenz in einer Tabelle angezeigt. Die Auswahl einer Erkrankung präsentiert weitere Informationen unter anderem als Graph. Mit Rechtsklick kann der Graph traversiert werden und mit der Maus lässt sich die Anordnung verschieben [41].

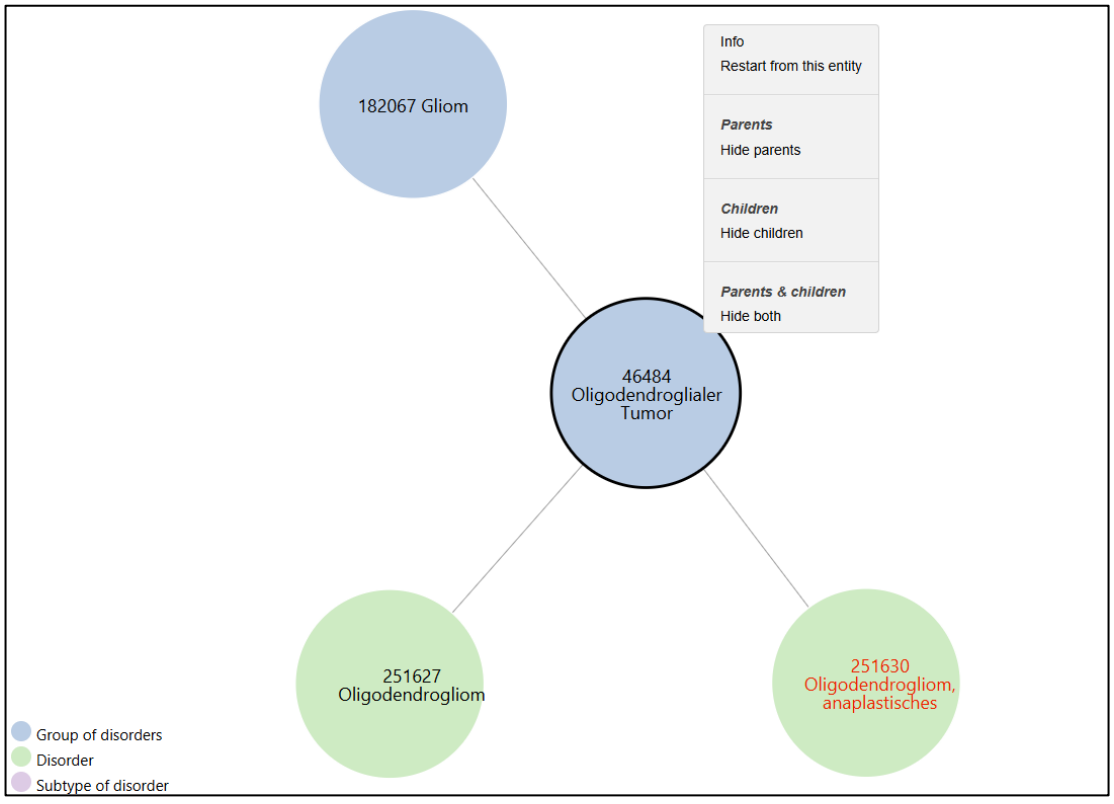


Abbildung 5: Datenvisualisierung in Orphanet

Der Java-Script-Code des Visualisierungstools wurde in GitHub veröffentlicht. Die verwendeten Variablen heissen beispielsweise «graphData» oder «queryNode» [42]. Zudem kann die Orphanet Rare Disease Ontology (ORDO) über einen SPARQL Endpoint abgefragt (siehe Abbildung 6) oder als owl-Datei (siehe Abbildung 7) bei GitHub heruntergeladen [43] werden. Dort stehen die Daten auch als Blazegraph triplestore zum Download bereit [44]. An diesem Beispiel erkennt man die enge Verzahnung von Ontologien und Graph-Datenbanken.

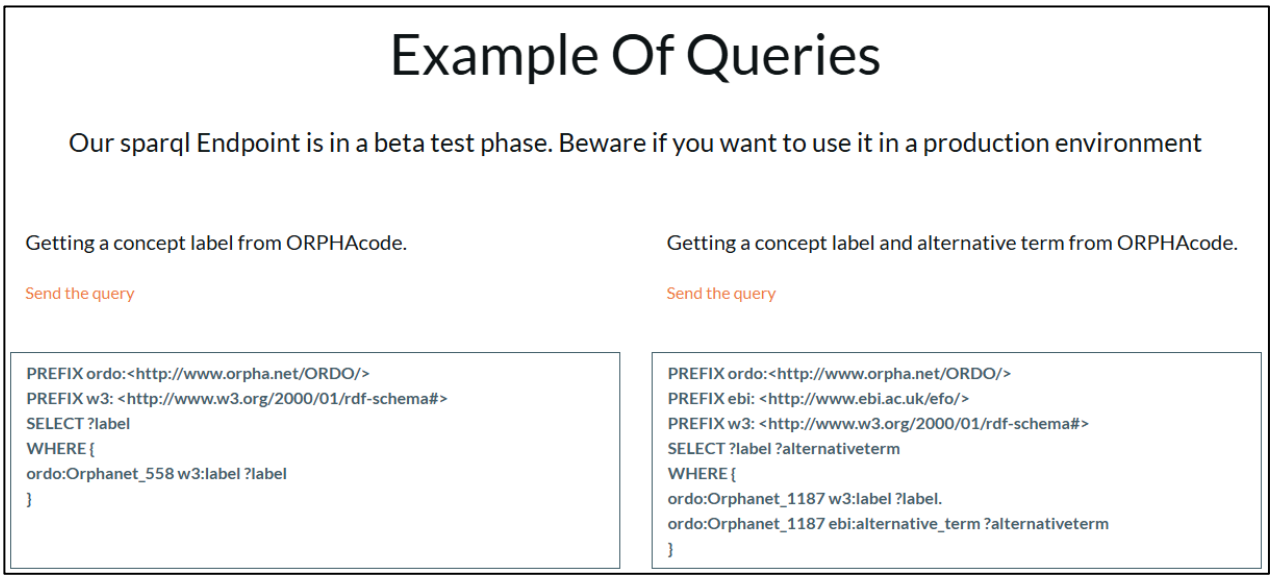


Abbildung 6: ORDO SPARQL Endpunkt

Abbildung 7: Ausschnitt aus der Orphanet Rare Disease Ontology owl Datei

3 Methoden

3.1 Literaturrecherche

Ausgangspunkt dieser Bachelorarbeit waren zwei Artikel als Teil der Aufgabenstellung [12,45]. Darüber hinaus wurde eine Literaturrecherche in der Datenbank «Pubmed» durchgeführt, um das Thema weiter zu analysieren und zusätzliches Wissen zu erlangen.

Tabelle 2: Suchbegriffe und Anzahl Treffer in der Pubmed Datenbank (am 06.03.2025)

Suchbegriff (Search-String)	Anzahl Treffer
"graph database"[Title/Abstract]	195
"graph database"[Title/Abstract] OR "neo4j"[All Fields]	240
"graph database"[Title/Abstract] AND "neo4j"[All Fields]	68
"graph database"[Title/Abstract] NOT "neo4j"[All Fields]	127
"neo4j"[All Fields]	113
"graph database"[Title/Abstract] AND "ontology"[All Fields]	36

3.2 Projektmanagement

Das Projektmanagement war integraler Bestand der Bachelorarbeit. Die ausführlichen Dokumente über die Projektplanung sind im GitLab-Repository verfügbar. In diesem Abschnitt folgt lediglich eine kurze Zusammenfassung des Projektmanagements.

Tabelle 3: Projektphasen

Phase	Zeitraum	Beschreibung
Initiierung	Januar 2025	Der Auftrag wurde fixiert.
Planung	Ende Februar 2025	Das Projekt wurde geplant.
Konzept	März 2025	Recherchen wurden durchgeführt und die Anforderungen erhoben.
Umsetzung	April und Mai 2025	Datenquellen wurden aufgearbeitet, das Datenmodell entwickelt und getestet.
Abschluss	Juni 2025	Präsentation und Abgabe der Ergebnisse.

Das Projektergebnis besteht aus drei zentralen Komponenten:
1. Graph-Datenmodell: Die Datenquellen werden in Neo4j integriert, sodass die fachlichen Fragestellungen beantwortet werden können.
2. Analysen für die fachlichen Anforderungen: Die Auftraggeberin kann mit vorhandenen Analysetools selbstständige Analysen durchführen und die Fragen beantworten.
3. Dokumentation: Die technischen Dokumente, Anwenderdokumente und weitere Deliverables werden der SwissDRG AG und der BFH (Betreuer und Experte) zur Verfügung gestellt.

Im Rahmen des Risikomanagements wurden Projektrisiken identifiziert, bewertet und Massnahmen definiert. Die erarbeiteten Gegenmassnahmen können der folgenden Tabelle entnommen werden.

Tabelle 4: Projektrisiken und Gegenmassnahmen

Risiken	Massnahmen
Ausfall der Graph-Datenbank	Mehrere Betriebsumgebungen, konsequente Nutzung von GitLab zur Speicherung der Zwischenresultate.
Verzögerung bei Aufbereitung der Datenquellen	Problematische Datenquellen erst später einbinden und/oder Experten konsultieren.
Falsche Use-Cases	Fachlichen Anforderungen mit der Auftraggeberin validieren und einen User-Test durchführen.
Technisch fehlerhaftes Datenmodell	Testkonzept erarbeiten und konsequent anwenden.
Einflussnahme aus SwissDRG internem Parallelprojekt auf Bachelor-Thesis	An eigenen Projektplan halten und dies gegenüber den Stakeholdern des Parallelprojektes so kommunizieren.

Als Arbeitsinstrumente wurde fünf Haupttools genutzt.
- GitLab: Tracking und Versionierung der Arbeitspakete, Zeiterfassung, Code, Protokolle und Projektdokumente.
- Neo4j: Graph-Datenbank mit Version 5.26.1 in mehreren Umgebungen als Community-Edition sowie als Desktop-Entwicklerapplikation (Enterprise-Edition).
- Code-Editors: VS-Code 1.97 mit Extensions u.a. für Neo4j, Git und R 4.3.1, R-Studio 2024.12.0
- MS Office 365: Office Paket der BFH inkl. OneDrive für Bearbeitung diverser Projektdokumente.
- Literatur: Zotero 6.0.36 wurde als Literaturverwaltungsprogramm genutzt.

Mit Hilfe eines R-Skriptes sind die erfassten Zeiten aus der GitLab GraphQL API abgefragt und graphisch dargestellt worden. Zudem sind Besprechungen zwischen Betreuer und/oder Experte mit dem Studenten durchgeführt worden, um den Projektfortschritt zu besprechen und Rückmeldung einzuholen. Die Besprechungen haben alle drei Wochen per MS Teams stattgefunden und wurden protokolliert.

3.3 Anforderungserhebung

Im Rahmen der Bachelor-Thesis sollen insbesondere die fachlichen Anforderungen umgesetzt werden. Die technischen Anforderungen dienen der Beurteilung diverser Graph-Datenbanken und stellen die Grundlage für eine spätere Entscheidung über den Einsatz einer konkreten Graph-DB in der SwissDRG AG dar.

3.3.1 Erhebung der fachlichen Anforderungen

Am 04.03.2025 sowie am 12.03.2025 sind die fachlichen Anforderungen mit der Auftraggeberin erarbeitet worden. Hierbei handelt es sich um fachliche Fragen, die mit Hilfe des Datenmodells beantwortet werden sollen. Grundsätzlich sollen relevante Informationen und deren Zusammenhänge mit diversen Grunddaten ausgegeben werden. Es sind drei unterschiedliche Ansätze («User-Stories») besprochen, die die grundlegenden funktionalen Anforderungen darstellen. Zudem sind die benötigten Abfragen formuliert und priorisiert worden. Anforderungen an die Anzeige oder den Umgang mit den Daten wurden nicht erhoben, da für die Datenpräsentation bestehende Anwendungen (z.B. von Neo4j der Browser oder Bloom/Explore) genutzt werden sollen.

3.3.2 Erhebung der technischen Anforderungen

Falls das Graph-Datenmodell in der SwissDRG AG eingesetzt und eine Graph-Datenbank installiert werden sollte, müssen die Rahmenbedingungen der technischen Infrastruktur der SwissDRG AG berücksichtigt werden. In einer Besprechung am 11.03.2025 wurden die technischen Systemanforderungen mit dem Systemadministrator erhoben.

3.4 Vorgehen für die Entwicklung des Datenmodells

Im Zentrum des Projektes steht die «Liste der hochteuren Medikamente/Substanzen» (Medi-Liste) der SwissDRG AG. Aus diesem Grund hat die Entwicklung des Datenmodells mit der Medi-Liste begonnen und es sind nach und nach weitere Daten zum Datenmodell hinzugefügt worden. Nach der Anforderungserhebung wurden zunächst die benötigten Datenquellen identifiziert. Danach wurden viele Datenquellen mit Hilfe von R oder Python und den folgenden Preprocessing-Skripten aufbereitet:

- 02_technisches_begleitblatt.R
- 03_zusatzentgelte.R
- 04_ema_produkte.R
- 05_refdata.R
- 06_parse_atc_ddd.py
- 07_bag_sl.R
- 09_antraege.R

Die ausführliche Beschreibung der Preprocessing-Schritte kann dem Anhang entnommen werden (Kapitel 11.2). Die csv-Input-Dateien können lokal im Import-Ordner der Neo4j Datenbank gespeichert werden, wenn man mit 'file:/// …' darauf zugreifen möchte. Alternativ können die csv-Dateien über eine https-Verbindung in Neo4j hineingeladen werden.

Schliesslich wurde das Datenmodell pro Themengebiet sukzessive skizziert und mit Cypher-Code in Neo4j implementiert. Das Kapitel 4.2 beschreibt ausführlich die Entwicklung des Datenmodells.

3.5 Test-Konzept

Testen und die Erstellung eines Test-Konzeptes sind ein wesentlicher Bestandteil jedes Software-Entwicklungsprojektes. Dadurch kann die Qualität einer Software erhöht werden [46]. Im Rahmen der Bachelor-Thesis steht die Entwicklung eines Datenmodells im Zentrum. Gewisse Tests wie Integrations-Komponententests mit klassischen «assertions» spielen somit keine Rolle, weil keine eigenen Methoden für die Graph-Datenbank entwickelt werden.

Vielmehr werden bereits implementierte Methoden der Graph-Datenbank verwenden und man kann davon ausgehen, dass das Datenmodell (Knoten und Kanten) ordnungsgemäss erstellt wird. Allerdings muss getestet werden, ob die Anweisungen zur Erstellung des Datenmodells fehlerfrei sind. Zudem sollten mit Hilfe von User-Tests geprüft werden, ob die erarbeitete Lösung der Auftraggeberin einen Mehrwert bietet und die fachlichen Anforderungen erfüllt werden.

3.5.1 Tests des Datenmodells

Die Entwicklung des Datenmodells wurde systematisch getestet.
- Die Entwicklungsumgebung enthält zwei Datenbanken: Eine Test-Datenbank und eine normale Datenbank mit den Ergebnissen.
- Das Datenmodell wird zunächst mit https://arrows.app skizziert
- Vorgängig wird ein kleiner Beispieldatensatz aus den Datenquellen extrahiert.
- Der Cypher-Code wird mit dem kleinen Auszug auf der Test-Datenbank entwickelt.
- Für die Validierung des Modells wird
  - die Anzahl an Knoten und Kanten gezählt
  - jeder Datensatz des Testdatensatzes in der Tabellenansicht und/oder Graph-Ansicht validiert und
  - die Validierung dokumentiert.
- Bei erfolgreicher Validierung wird das Skript mit dem vollen Datensatz auf der normalen Datenbank ausgeführt.
- Das skizzierte Datenmodell wird mit dem finalen Datenmodell verglichen.
- Anzahl an Knoten und Kanten auf der normalen Umgebung werden gezählt, drei Stichproben geprüft und dokumentiert.

3.5.2 User-Test

Für die zweite Hälfte der Entwicklung des Datenmodells wurde ein User-Test geplant. Hierbei sollte überprüft werden, ob die Anwenderin die fachlichen Fragestellungen mit Hilfe des Datenmodells beantworten kann. Der User-Test ist am 07.05.2025 durchgeführt worden. Um die Anforderung und die gewünschten Abfragen zu prüfen, wurden fünf Szenarien erstellt. Die Anwenderin hat folgende Szenarien in einem praktischen Test auf Ihrem eigenen Arbeitslaptop mit der Neo4j Aura Cloud-Umgebung und einer vorbereiteten User-Test-Perspektive bearbeitet.

Szenario 1:
- Ich möchte wissen, ob sich die Substanz mit dem ATC-Code B01AX01 auf der Medikamentenliste befindet.
- Ich möchte wissen, wie die Substanz heisst,
- welche Verabreichungsart vorgesehen ist und
- ob ein Zusatzentgelt (wenn ja, welche Nummer) etabliert wurde.

Szenario 2:
- Ich suche Informationen zu dem Medikament mit dem Namen «phesgo».
- Zu diesem Produkt möchte ich den Namen der Zulassungsinhaberin wissen und
- ob sich die entsprechende Substanz auf der Medikamentenliste befindet.
- Ich möchte dabei prüfen, ob sich der ATC-Code der Substanz früher einmal geändert hat.
- Zudem möchte ich den aktuellen ExFactory-Preis der Spezialitätenliste für die 1200mg Packung wissen.

Szenario 3:
Ein neues Produkt wird von der SwissMedic zugelassen mit der Indikation «Ovarialkarzinom, Eileiterkarzinom und primäres Peritonealkarzinom».
- Nun möchte ich wissen, ob es bereits Substanzen auf der Medikamentenliste mit der gleichen Indikation gibt und
- ob dafür Zusatzentgelte etabliert wurden.
- Zudem sollen weitere Informationen im Zusammenhang mit dieser Frage erkundet werden.

Szenario 4:
- Ich möchte mir alle Substanzen und die ATC-Hierarchie für die ATC-Gruppe J02A anzeigen lassen.
- Diese soll im hierarchischen Layout angezeigt werden.
- Zudem möchte ich mir die Tagesdosis (DDD) und die «Route of Administration» (WHO-Daten) für den ATC-Code J02AB02 anzeigen lassen.

Szenario 5:
Durch einem Anruf eines SwissDRG Partners werde ich auf eine Substanz aufmerksam gemacht, die im Jahr 2023 gelöscht wurde.
- Ich möchte alle gelöschten Substanzen aus 2023 finden.
- Zudem möchte ich mir die Indikationen der entsprechenden Produkte anzeigen lassen.

Der User-Test bildete die Grundlage für eine Weiterentwicklung des Datenmodells gegen Ende der Umsetzungsphase. Vor der Durchführung des User-Tests erfolgte eine Schulung der Anwenderin, sodass sie die Szenarien selbstständig mit Neo4j Bloom sowie vorgefertigten Queries bearbeitet werden konnten.

3.6 Schulungs-Konzept

Der User-Test soll einerseits das Datenmodell testen und Verbesserungsoptionen aufzeigen und andererseits soll damit die Akzeptanz zur Analyse der Daten in Form eines Graphen geprüft werden. Die Auftraggeberin hat bisher wenig Erfahrung mit dynamisch erstellten Graphen und dem Toolset von Neo4j.

Vor dem User-Test musste eine Schulung durchgeführt werden, sodass die Anwenderin den grundsätzlichen Umgang mit Neo4j Bloom lernen konnte. Aufgrund des grossen Bekanntheitsgrades

von Neo4j sind viele Online-Ressourcen verfügbar. Die Schulung sollte auf die frei verfügbaren Quellen (z.B. Youtube-Tutorials etc.) zurückgreifen.

Im Rahmen des Schulungskonzeptes sollte das Online-Material gesichtet und Text- oder Video-Tutorials ausgewählt werden, mit denen die Anwenderin innerhalb von 10-20 Minuten die zentralen Funktionen zur Durchführung der User-Testszenarien lernen kann. Zudem wurde bei der Recherche bereits Text- oder Videomaterial gespeichert, das einen Lernaufwand von mehr als einer Stunde verursacht und im Falle des Einsatzes von Neo4j bei der SwissDRG AG präsentiert werden könnte. Ausserdem wurde eine kompakte Nutzerdokumentation mit Screenshots und Erläuterung der Schaltflächen mit dem Datenmodell des User-Tests erstellt. Der User-Test und die Schulung wurden mit der Neo4j Aura-Umgebung auf dem Laptop der Auftraggeberin durchgeführt.

Während der Vorbereitung des Schulungsmaterials fiel auf, dass die Visualisierungen in Neo4j Bloom situativ GPU-Ressourcen benötigen. Die folgende Abbildung 8 illustriert die Leistung der lokalen Windows-Umgebung, während eine Abfrage in Neo4j Explore mit einem Limit von 1'000 Knoten erstellt wurde. Die beiden Leistungsspitzen der GPU (ca. 40% bis 50%) und des WLAN sind auf zwei Abfragen in der Cloud-Umgebung zurückzuführen. Die benötigten GPU-Ressourcen sollten bei der Schulung, dem User-Test und der Planung über den Einsatz von Neo4j in der SwissDRG AG berücksichtigt werden.

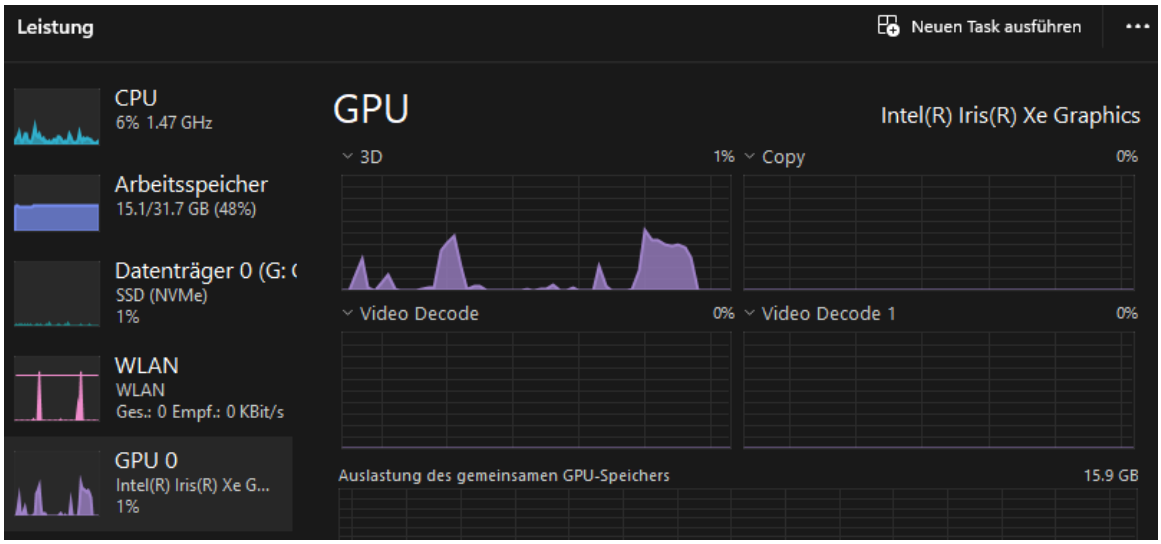


Abbildung 8: GPU-Ressourcen bei Graph-Darstellungen in Neo4j Bloom

3.7 Kriterien zur Auswahl möglicher Graph-Datenbanken für SwissDRG AG

Möglicherweise wird nach Abschluss der Bachelorarbeit eine Graph-Datenbank bei der SwissDRG AG installiert. In Kapitel 2.2.2 wurden viele aktuell einsetzbare Graph-Datenbanken aufgelistet. Die Tabelle 1 kann somit als «long list» verstanden werden. Mit Hilfe der technischen Anforderungen (siehe Abschnitt 4.1.2) sowie weiterer Kriterien kann eine «short list» erstellt werden.

Kriterium 1 (Schulungsprogramm)
Der Hersteller der Graph-Datenbank bietet ein kostenloses Schulungsprogramm/Tutorials an, um Besonderheiten der Datenbank und der Abfragesprache zu erlernen.

Kriterium 2 (Strukturierte Dokumentation)
Es gibt eine strukturierte Dokumentation mit Erläuterungen über den grundlegenden Umgang mit der Datenbank und insbesondere Administrationsthemen.

Kriterium 3 (Abfragesprache)
Cypher oder openCypher sollten unterstützt werden, da die Bachelor-Thesis mit Cypher umgesetzt wird. Zudem könnte die Codebasis und die Dokumentation weiterverwendet werden, wenn man die Abfragesprache nicht ändern muss, aber man dennoch die Datenbank wechseln möchte. Ein Backend-Wechsel wäre somit ohne allzu grosse Umstellung möglich.

Kriterium 4 (Visuelle Analysetools)
Die Datenbank sollte über ein graphisches Analysetool verfügen, sodass die Anwenderin die Daten ohne (viel) Code abfragen kann (no/low code).

Kriterium 5 (Community-Edition)
Die Datenbank sollte eine kostenlose Edition anbieten. Dadurch kann die SwissDRG AG das Datenmodell als Pilot einführen und sich danach bei Erfolg für ein Enterprise-Lizenz entscheiden.

Die technischen Anforderungen 1 und 2 (lokaler Betrieb in einem Docker-Container, siehe Abschnitt 4.1.2) sind die beiden einzigen absoluten Einschlusskriterien. Die dritte technische Anforderung wurde als Kriterium bewertet, in dem die komprimierte Image-Grösse gemäss Docker Hub unter ein Gigabyte liegen muss. Ist ein Kriterium erfüllt, wird 1 Punkt vergeben, sonst 0. Im Anschluss wurde die Summe gebildet und absteigend sortiert. Bei Punktgleichheit wurde alphabetisch sortiert.

3.8 Infrastruktur

3.8.1 Virtuelle Maschine

Für die Ergebnisse des Projektes kann die virtuelle Maschine (VM) der BFH vom LC2-Projekt aus dem Herbstsemester 24/25 weiterverwendet werden. Dabei handelt es sich um ein Debian System ohne GUI, mit 4 GB RAM und 200 GB Speicher. Die VM ist bis 31.07.2025 gültig und wird danach terminiert. Die VM wurde mit einem allgemeinen BFH-User, SSH und aktiver Firewall (ufw) ausgeliefert. Die IP-Adresse lautet: 10.248.11.58. Für ein effizientes Arbeiten mit der VM wurde der SSH-Zugang mit einem RSA-Schlüsselpaar und IP-Alias «graph» eingerichtet (siehe Abbildung 9).

3.8.2 Docker

Fast alle Applikationen der SwissDRG AG laufen in Docker-Containern (siehe dazu auch Kapitel 4.1.2, Anforderung 3). Um dieses Projekt möglichst realitätsnah zu gestalten, wurde Docker in der VM installiert wie es in der offiziellen Docker-Dokumentation beschrieben ist [47]. Docker wurde mit der Community-Version 27.5.0 auf der VM installiert.

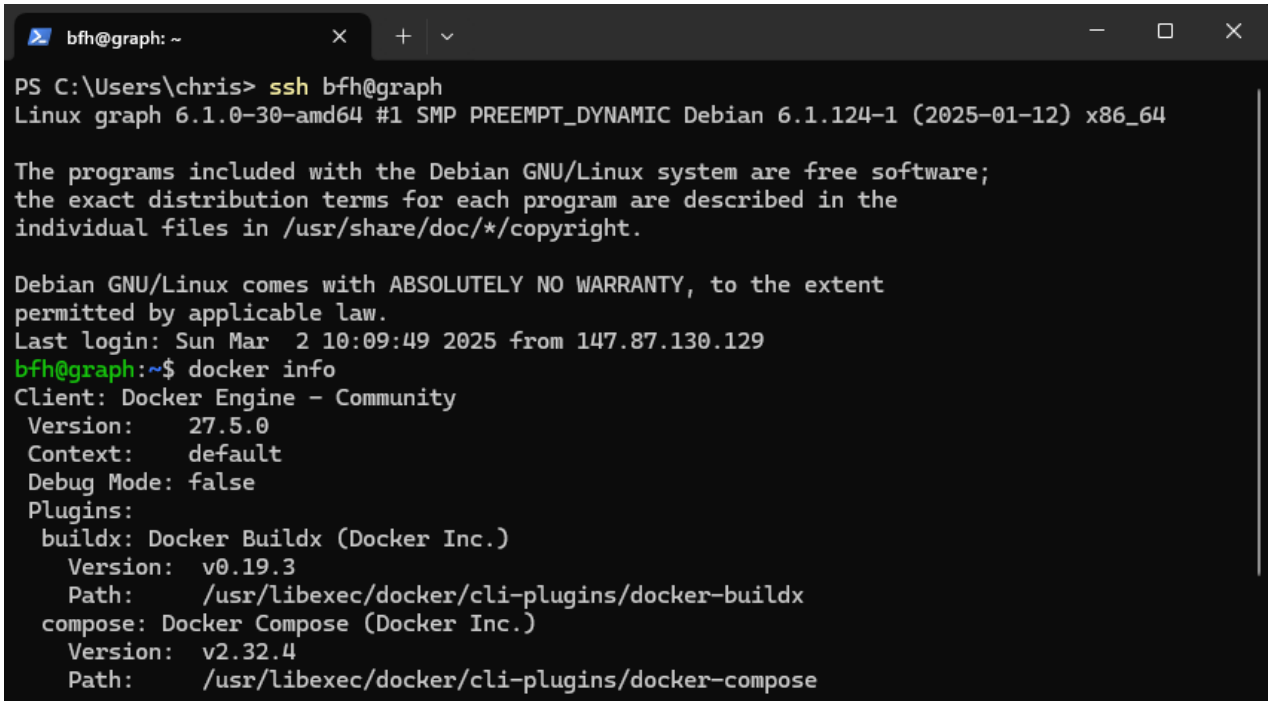


Abbildung 9: Docker auf der VM der BFH

3.8.3 Neo4j

Im Rahmen der Bachelor-Thesis wird zwar evaluiert, welche Graph-Datenbanken gemäss den Anforderungen für einen Einsatz innerhalb der SwissDRG AG in Frage kämen, allerdings wird das Datenmodell mit Neo4j entwickelt. Der Grund ist, dass Neo4j ein weit verbreitetes System und dies im Modul Advanced Databases im siebten Semester gelehrt worden ist.

3.8.3.1 Umgebungen

Es wurden vier Umgebungen von Neo4j betrieben. Die Zuordnung zu den Bereitstellungsumgebungen kann man sich im weitesten Sinne wie folgt vorstellen:

- Entwicklung: Lokale Windows-Umgebung mit Neo4j Desktop inkl. Enterprise-Edition mit zwei Datenbanken für normale Entwicklung und Testing des Datenmodells.
- Test: Lokale Debian VM mit Neo4j Community-Edition zum Test der Installation.
- Staging: Remote auf der VM der BFH (siehe Abschnitt 3.8.1) mit Neo4j Community-Edition.
- Produktion: Cloud-Lösung mit Neo4j Aura (free) [48].

Die Entwicklung des Datenmodells sowie der Analyseansichten und das Testing wurde in Neo4j Desktop unter Windows 11 realisiert. Diese Entwicklungsumgebung hätte grundsätzlich für die Durchführung des Projektes genügt. Die weiteren Umgebungen wurden allerdings für spezifische Aufgaben erstellt. Die lokale Debian VM Instanz diente dabei lediglich als Testumgebung, um die Infrastruktur auf der VM der BFH aufzusetzen.

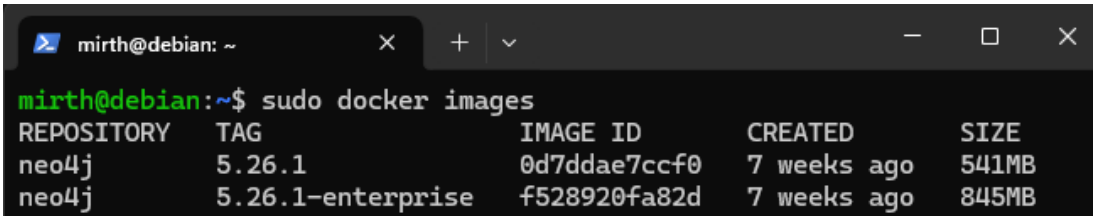


Abbildung 10: Informationen über Docker Images auf der lokalen Test-Umgebung

Das Datenmodell als ein zentrales Ergebnis dieser Bachelorarbeit wird auf der VM der BFH zur Verfügung gestellt. Somit kann die VM der BFH als «Staging» angesehen werden. Allerdings wurde auf Staging nur die Community-Edition installiert, weil die Enterprise-Version hätte lizensiert werden müssen.

Die Cloudlösung Neo4j Aura wurde deshalb verwendet, weil dort die Enterprise-Version vorhanden und «Explore» integriert ist. Das Modul «Explore» ist die Cloud-Variante von Neo4j Bloom und es fallen keine Kosten für die Nutzung an [49]. User-Tests und die Analysemöglichkeiten wurden somit auf der Cloud-Plattform durchgeführt. Die Auftraggeberin kann die Fragen der Use-Cases bzw. fachlichen Anforderungen in Neo4j Aura selbstständig durchführen, nachdem sie sich mit Username und Passwort eingeloggt hat. Nach Abschluss des Projektes wird entschieden, ob und welche technische Lösung weiter betrieben wird und ob eine Lizensierung der Neo4j Enterprise-Edition oder eines anderen Herstellers durchgeführt wird.

3.8.3.2 Installation von Neo4j auf der VM der BFH (Remote)

Für die «Staging Umgebung» auf der VM der BFH (siehe Kapitel 3.8.1) wurde das Docker-Image Neo4j 5.26.1 (Community-Edition) heruntergeladen. Folgender Befehl wurde im Terminal ausgeführt:

```
docker pull neo4j:5.26.1
```

Die Neo4j-Version 5.26.1 wurde verwendet, weil diese Version zum Beginn des Projektes für alle Betriebsumgebungen verfügbar war. Der Docker-Container wurde mit üblichen Konfigurationen gestartet [50]. Allerdings wurde der Name (swissdrg) und das Passwort (medikamente) geändert. Zudem werden die Daten in einer Volume (/home/bfh/neo4j) gespeichert. Wenn die Datenbank resp. der Container abstürzt und/oder der Server neu gestartet werden muss, so startet der Container resp. die Neo4j-Datenbank automatisch neu.

```
docker run -d \
    --name=swissdrg \
    --restart always \
    --publish=7474:7474 --publish=7687:7687 \
    --env NEO4J_AUTH=neo4j/medikamente \
    --volume=/home/bfh/neo4j:/data \
    neo4j:5.26.1
```

Mit diesem Setup kann man sich beispielweise über den Web Bowser in den Neo4j Bowser einloggen (siehe Abbildung 11). Dafür muss man im Web Browser die IP-Adresse und den Port 7474 (entsprechend der durchgeführten Docker-Container Konfiguration) eingeben. Danach können der Username und das Passwort eingegeben werden. Alternativ ist eine Verbindung zur Datenbank mit Hilfe von Neo4j Desktop (siehe Abbildung 12) oder mit anderen Tools wie z.B. VS-Code mit der Neo4j Extension möglich (siehe Abbildung 13).

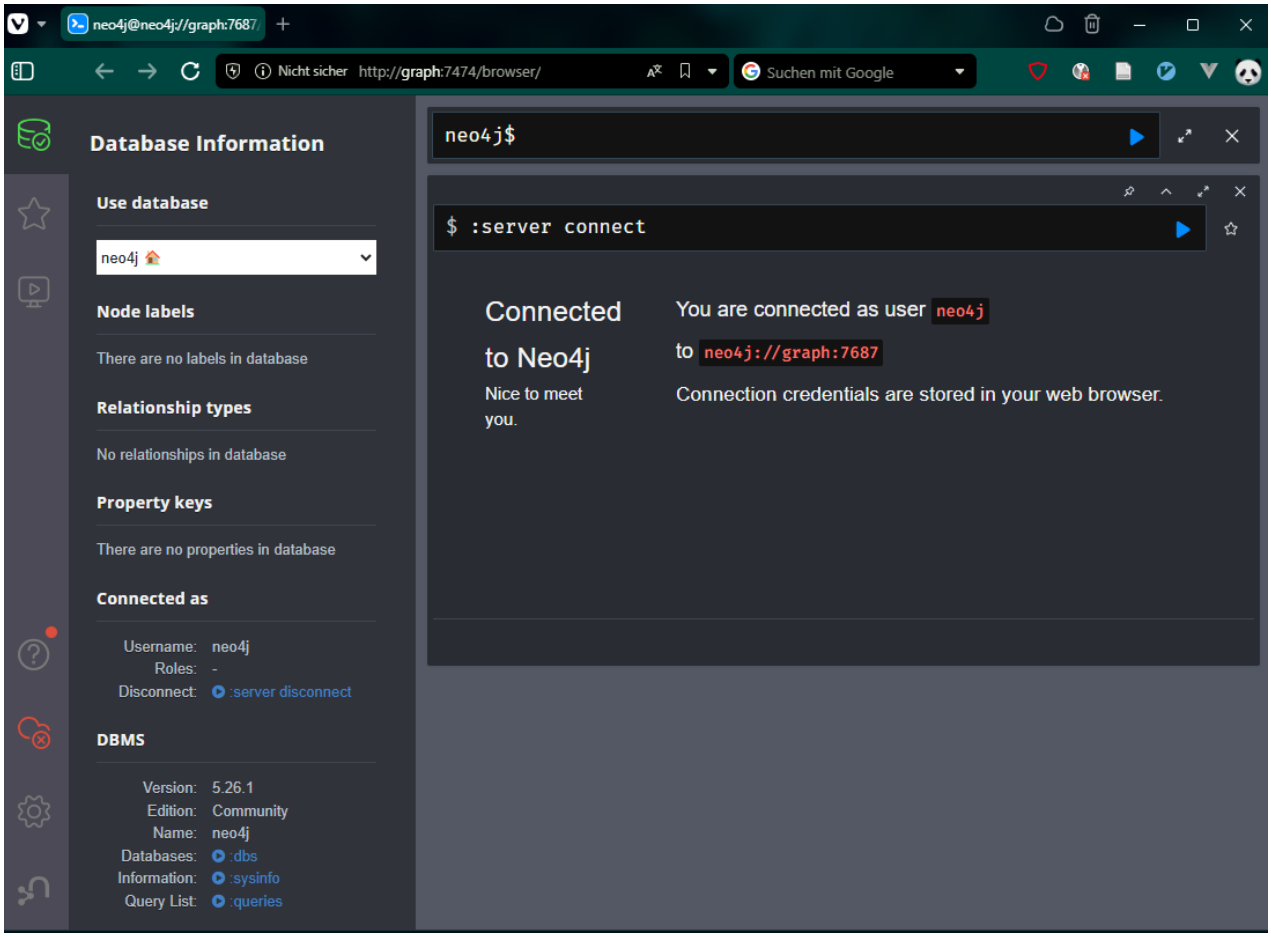


Abbildung 11: Neo4j Bowser auf der VM der BFH

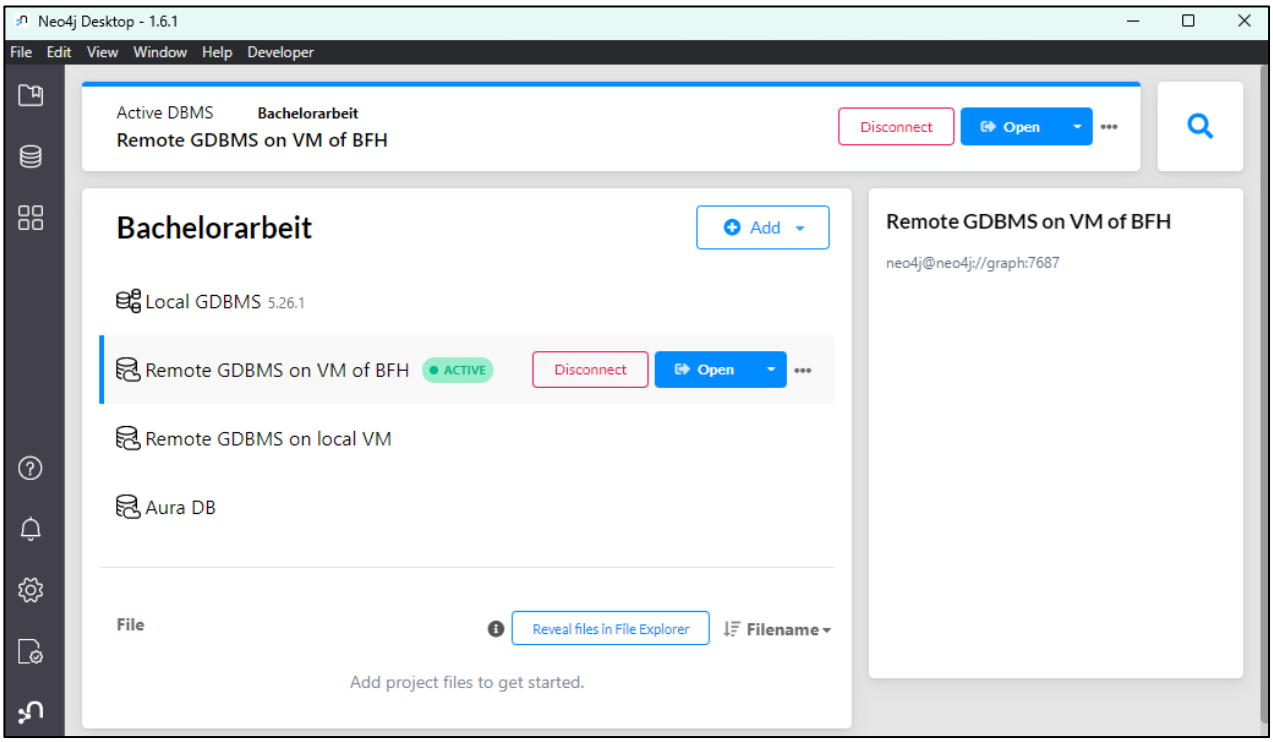


Abbildung 12: Neo4j Desktop mit Verbindung zu DB auf VM der BFH

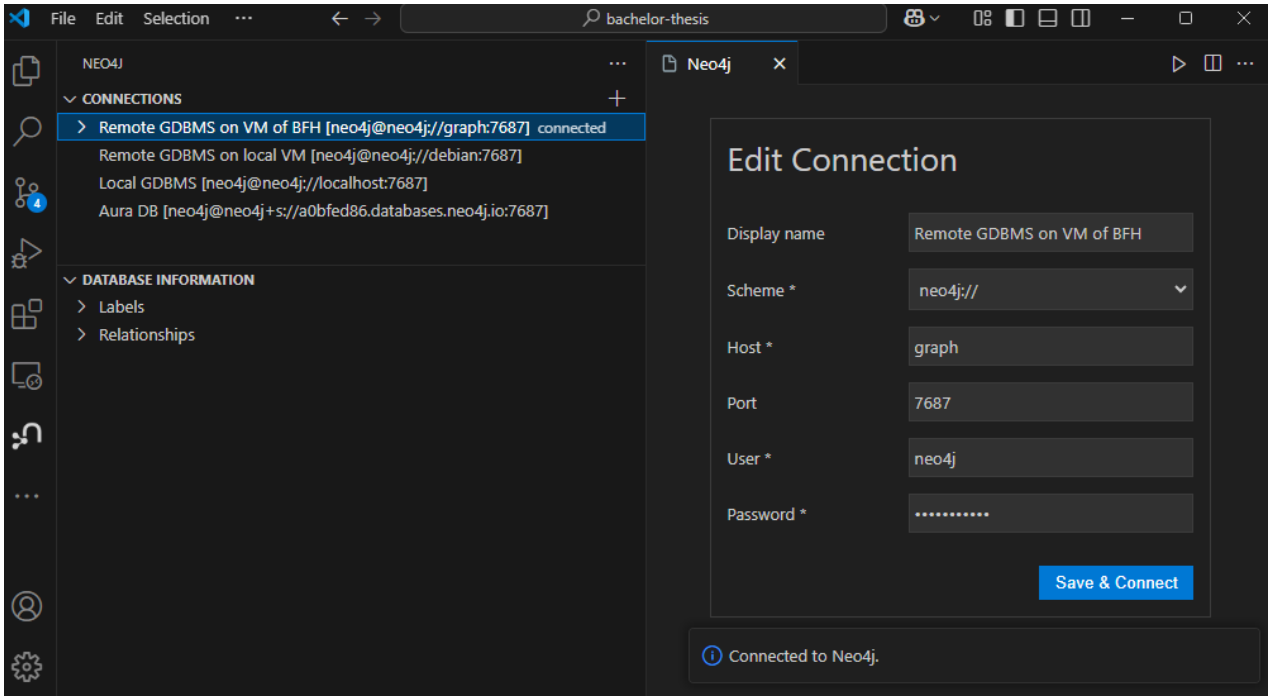


Abbildung 13: Verbindung zu Neo4j auf VM der BFH mit VS-Code

4 Ergebnisse

4.1 Anforderungen

In mehreren Besprechungen mit der Auftraggeberin und dem IT-Systemadministrator wurden die fachlichen und technischen Anforderungen erhoben.

4.1.1 Fachliche Anforderungen

Fachliche Anforderung 1 (Medikamente für eine Indikation anzeigen, hohe Priorität)
Als Fachexperte nehme ich Kenntnis von einem neuen (potenziell hochteuren) Medikament/Produkt, dass in den USA, der EU oder der Schweiz zugelassen werden soll bzw. worden ist. Dieses wird üblicherweise im stationären Umfeld eingesetzt. Als Anwender möchte ich eine Suchmaske öffnen und die medizinische Indikation (therapeutischer Bereich/MeSH [51]) eingeben. Das System soll mir die Daten aus der «Liste der hochteuren Medikamente/Substanzen» und den Zusatzentgelten anzeigen, die die gleiche Indikation haben wie das neue Produkt. Es sollen die Stammdaten der bisherigen Substanzen, Produktbezeichnungen, Zulassungsinhaberinnen und Informationen der Spezialitätenliste sowie deren Verbindungen untereinander angezeigt werden. Dadurch können die Medikamentenliste oder die Zusatzentgelte zielgerichtet angepasst werden.

Fachliche Anforderung 2
Im Arbeitsalltag tauchen Fragen über ein Medikament oder einen ATC-Code auf. Als Anwender möchte ich eine Suchmaske öffnen und die Produktbezeichnung, den ATC-Code oder die ATC-Bezeichnung eingeben können. Das System soll mir diverse Informationen dafür zusammenstellen.

- **Abfrage 1 (Medikamente der ATC-Hierarchie, hohe Priorität)**
  Das System soll mir alle Medikamente der «Liste der hochteuren Medikamente» und der Zusatzentgelte anzeigen, die in eine pharmakologische Wirkstoffgruppe fallen. Als Wirkstoffgruppe möchte ich die drei- oder vierstelligen ATC-Codes auswählen können. Stammdaten der siebenstelligen ATC-Codes sowie deren Produkte sollen angezeigt werden. Dadurch erhalte ich einen guten Überblick über eine Gruppe von Medikamenten.

- **Abfrage 2 (Jahresübergreifende Daten, mittlere Priorität)**
  Das System soll mir nicht nur die Einträge der aktuellen «Liste der hochteuren Medikamente/Substanzen» und der Zusatzentgelte anzeigen, sondern auch die Einträge der vergangenen Jahre bzw. gelöschte/nicht mehr vorhandene Einträge. Dabei sollen die aktuellen ATC-Codes und frühere ATC-Codes (vor ATC-Code-Änderungen) auswählbar sein und angezeigt werden.

- **Abfrage 3 (Medikamenten-Kombinationen, mittlere Priorität)**
  Das System zeigt mir Medikamente an, die üblicherweise in Kombination mit einem bestimmten Produkt verabreicht werden. Dadurch kann ich besser einschätzen, welche weiteren Produkte auf die «Liste der hochteuren Medikamente/Substanzen» aufgenommen oder Zusatzentgelte etabliert werden sollten. Zudem könnte ich dadurch eine realistischere Bewertung der Zusatzentgelte vornehmen (z.B. höhere CHF-Beträge, falls ein Produkt für einen Behandlungskomplex kalkuliert werden sollte).

- **Abfrage 4 (Preisinformationen der Medikamente, mittlere Priorität)**
  Das System soll mir optimalerweise die Tageskosten einer Substanz oder eines Produktes in CHF anzeigen. Die üblichen Tageskosten sind jedoch nur in wenigen Fällen bekannt bzw. auf unterschiedliche Art und Weise zu berechnen. Deshalb soll mir das System möglichst umfangreiche Informationen anzeigen, mit denen die Tageskosten nachträglich manuell (und näherungsweise) berechnet oder abgeschätzt werden können. Dafür benötige ich insbesondere Informationen über die Tagesdosis (Defined Daily Dose, DDD), aktuelle Preise der Spezialitätenliste inkl. Hinweisen über Generika, Preise je Einheit (gemäss Detailerhebung) oder sonstige Preisinformationen. Der Status als Orphan-Drug ist ebenfalls hilfreich. Dadurch kann ich schneller beurteilen, ob ein

Medikament «hochteuer» ist und zeitnah auf die «Liste der hochteuren Medikamente» aufgenommen werden sollte.

- **Abfrage 5 (Abgelehnte Anträge, mittlere Priorität)**
  Das System soll mir anzeigen, ob ein bestimmter ATC-Code bereits in der Vergangenheit beantragt und abgelehnt wurde, weshalb sich eine Substanz nicht auf der «Liste der hochteuren Medikamente» oder den Zusatzentgelten befindet. Dadurch kann ich mir einen besseren jahresübergreifenden Überblick verschaffen. Die öffentlich verfügbaren Antragsinformationen sollten zu einem späteren Zeitpunkt durch die internen Antragsinformationen ersetzt werden. Der Grund hierfür ist, dass die internen Daten mehr Details über einen Antrag anzeigen und auch das letzte Antragsjahr beinhalten.

- **Abfrage 6 (Nutzenbewertung, geringe Priorität)**
  Das System soll mir für ein Produkt die Nutzenbewertung gemäss des gemeinsamen Bundesausschusses (aus Deutschland) anzeigen. Neben der Auswahl eines einzelnen Produktes soll mir das System die Nutzenbewertung für alle Produkte eines ATC-Codes ausgeben. Dadurch kann ich besser beurteilen, ob eine Aufnahme auf die «Liste der hochteuren Medikamente/Substanzen» oder die Etablierung eines Zusatzentgeltes sinnvoll erscheint oder ob allenfalls eine Streichung von den Listen angezeigt ist.

Anforderung 3 (Interne Fallzahlen, geringe Priorität)
Das System soll mir die Anzahl der stationären Fälle und simulierter Zusatzentgelte je ATC-Code für die letzten drei Jahre anzeigen. Es sollen die Gesamtzahl sowie die einzelnen Werte je Tarifsystem (SwissDRG, TARPSY und ST Reha) ausgewiesen werden. Für jeden ATC-Code benötige ich zudem eine Angabe, ob und welche Zusatzentgelte vorhanden sind. Gleichzeitig möchte ich nach der Anzahl filtern können. Dadurch kann ich besser beurteilen, welche Medikamente von der «Liste der hochteuren Medikamente/Substanzen» entfernt werden können. Diese Anforderung kann erst zu einem späteren Zeitpunkt umgesetzt werden, da auf interne Daten zugegriffen werden muss. Der Zugriff auf interne Daten ist im Rahmen der Bachelor-Thesis aus datenschutzrechtlichen Gründen nicht vorgesehen.

4.1.2 Technische Anforderungen

Technische Anforderung 1 (Lokale Installation mit LDAP)
Als IT-Systemadministrator bevorzuge ich eine lokale Installation (onside) gegenüber einer gehosteten Lösung (Cloud), weil Sicherheitsthemen, Updates und Benutzerverwaltung intern flexibel verwaltet werden könnten. Dabei favorisiere ich LDAP für Windows-Authentifizierung, um auf das bestehende Berechtigungskonzept aufzusetzen (alternativ wäre Google Authentifizierung möglich).

Technische Anforderung 2 (Docker mit Compose File)
Das System (Graph-Datenbank) soll mit aktuellen Standard-Basesimages (z.B. Debian oder Ubuntu) dockerisiert sein. Ein Docker Compose File für das Deployment kann ich als IT-Systemadministrator selbst erstellen.

Technische Anforderung 3 (Benötigte Ressourcen)
Die Grösse des Docker-Images sollte mehrere GB nicht übersteigen (zum Vergleich: Neo4j 5.26.1 hat in der Community-Edition eine Grösse von 541 MB und in der Enterprise-Edition von 858 MB, siehe Abbildung 10). Die Nutzung des CPU sollte 4 Prozessoren nicht übersteigen. Falls mehr benötigt werden, müsste dies separat geplant werden.

Neben diesen Anforderungen müsste bei Bedarf noch geklärt werden, ob ein separater Neo4j Bloom Container erstellt werden muss. Backups könnten mit etablierten Prozessen der SwissDRG AG erfolgen. Die Graph-Datenbank würde über einen Nginx Server als Reverse Proxy aufgerufen werden. So werden alle internen Applikationen verwaltet. Das Deployment eines Graph-DB Containers würde vermutlich weniger als eine Stunde Aufwand machen. Lizenzkosten bis ca. 1'000 CHF könnten über das Budget der IT-Abteilung finanziert werden. Für höhere Beträge müsste die Geschäftsleitung konsultiert werden. Eine Entscheidung der Geschäftsleitung ist in der Regel innert weniger Tage möglich.

4.2 Entwicklung des Datenmodells

Im Zentrum des Datenmodells steht die «Liste der hochteuren Medikamente/Substanzen». Deshalb hat die Entwicklung dort begonnen. Danach wurden die Zusatzentgelte von SwissDRG, TARPSY und ST Reha dem Modell hinzugefügt. Weiter wurden Medikamente (Produkte/Präparate) gemäss der Europäischen Arzneimittelagentur und der Stiftung Refdata inklusive deren Indikationen ergänzt. Danach wurde der ATC/DDD-Index der WHO hinzugefügt, um alle ATC-Codes inkl. deren Änderungen über die Jahre in das Modell zu integrieren. Es folgte die Spezialitätenliste des Bundesamts für Gesundheit, die eine von mehrere Preisquellen darstellt. Zu guter Letzt wurden die Anträge für Medikamente und Zusatzentgelte aus dem SwissDRG Antragverfahren ergänzt. Die folgenden Unterkapitel erklären ausführlich die Zwischenergebnisse je Themengebiet.

4.2.1 Datenquellen als Grundlage

Basierend auf den fachlichen Anforderungen sind zunächst die Datenquellen der benötigten Informationen ermittelt worden. Diese sind oft für mehrere Abfragen relevant und eine Anforderung kann mehrere Datenquellen beinhalten. Eine Zuordnung ist somit nicht eindeutig. Die folgenden Links beinhalten weitere Informationen sowie die konkret benötigten Daten. Teilweise können die Daten direkt heruntergeladen werden. Viele Datenquellen werden in den Preprocessing-Schritten leicht modifiziert, sodass sie später in die Neo4j Datenbank importiert werden können.

- SwissDRG AG, Liste der hochteuren Medikamente
  https://www.swissdrg.org/de/akutsomatik/datenerhebung/erhebung-2026-daten-2025
- SwissDRG AG, Liste der hochteuren Medikamente, Vorjahre
  https://www.swissdrg.org/de/akutsomatik/datenerhebung/archiv
- SwissDRG AG, Zusatzentgeltkatalog (aktuelle Version 14.0)
  https://www.swissdrg.org/de/akutsomatik/swissdrg-system-1402025/fallpauschalenkatalog
- SwissDRG AG, Antragsverfahren
  https://antragsverfahren.swissdrg.org/de/swissdrg/proposals/public_index
- European Medicines Agency EMA, Download medicine data
  https://www.ema.europa.eu/en/medicines/download-medicine-data
- Stiftung Refdata, Strukturierte Arzneimittelinformationen
  https://sai.refdata.ch
- WHO Collaborating Centre for Drug Statistics Methodology, ATC/DDD Index
  https://atcddd.fhi.no/atc_ddd_index
- Bundesamt für Gesundheit, Spezialitätenliste
  https://www.spezialitätenliste.ch/Default.aspx

4.2.2 Nutzungsrechte

Für das Graph-Datenmodell werden die oben genannten, öffentlich verfügbaren Informationen verwendet. Dennoch sollten Nutzungsrechte abgeklärt und vorhanden sein.

Für die Daten der SwissDRG gilt grundsätzlich «Es ist dem Benutzer untersagt, irgendwelche Informationen und Inhalte (einschliesslich Texte, Daten, Grafiken, Logos, Dokumente, Bilder, Formulare, Verzeichnisse, Listen oder Software), die von dieser Website stammen, im Ganzen oder teilweise ohne die vorherige schriftliche Genehmigung der SwissDRG AG […] zu nutzen» [52]. Da die SwissDRG AG als Auftraggeber fungiert und eine schriftliche Vereinbarung geschlossen wurde, liegt die Genehmigung somit vor.

Die EMA stellt auf der Website «Download medicine data» diverse Informationen zur Verfügung. Dort heisst es «You can download data related to medicines published on EMA's website in table format» [53]. Da die Daten explizit als Excel-Datei angeboten werden, kann man davon ausgehen, dass diese für weitere Zwecke genutzt werden dürfen.

Für die Daten der Stiftung RefData gilt folgendes: «Zulässig sind der Export der INFORMATIONEN (XML-Download) zur Nutzung der Daten ausserhalb dieser Applikation, die Weiterverwendung und das

Bearbeiten der INFORMATIONEN sowie die Verwendung für öffentliche oder kommerzielle Zwecke» [54]. Dadurch können auch diese strukturierten Arzneimittelinformationen verwendet werden.

Bezüglich der Nutzungsrechte des ATC/DDD-Index schreibt das Norwegische «Institute of Public Health WHO Collaborating Centre for Drug Statistics Methodology» folgendes: «Access to a searchable version of the ATC Index with DDDs (including text from the Guidelines) is available free of charge on this website» [55]. Zusätzlich findet man auf der Website https://atcddd.fhi.no/robots.txt den Eintrag «Crawl-Delay: 10». Im Rahmen des Preprocessings wird die Website des ATC/DDD-Index mit einem Python-Skript rekursiv abgerufen. Der Websiteaufruf wird um 10 Sekunden verzögert, in dem der Befehl `time.sleep(10)` zwischen den einzelnen Requests hinzugefügt wurde.

Das Bundesamt für Gesundheit «erstellt verschiedene Listen, welche die Preise und Tarife für Arzneimittel festlegen, die von der obligatorischen Krankenpflegeversicherung (OKP) oder der Invalidenversicherung (IV) höchstens vergütet werden. Dazu gehören die Liste der pharmazeutischen Spezialitäten und konfektionierten Arzneimittel mit Preisen (Spezialitätenliste, SL)» [56]. Da die Daten explizit auf der Website spezialitätenliste.ch als Excel-Datei, XML-Dateien und XSD-Schema-Dateien zum Download zur Verfügung gestellt werden, ist davon auszugehen, dass die Daten der Spezialitätenliste für weitere Zwecke genutzt werden dürfen.

4.2.3 Integration der «Liste der hochteuren Medikamente/Substanzen»

Für die Medi-Liste und alle anderen Datenquellen wurde jeweils ein Datenmodell skizziert. Im Label «Substanz» wurden die ATC-Codes und Bezeichnungen sowie weitere Stammdaten als Properties gespeichert. Vier weitere Labels repräsentieren die Verabreichungsarten, Zusatzangaben und Einheiten, die jeweils mit «HAT_» Relationen verbunden sind. Das Label «Tarif» soll mit der «Substanz» über die Relation «GILT_FUER_TARIF» verbunden sein. Substanzen können keine, eine oder viele Relationen zu keinen, einem oder mehreren Knoten «Tarif», «Verabreichungsart» und «Zusatzangaben» besitzen. Die Relation der «Substanz» zur «Einheit» ist jedoch eindeutig (eins zu eins Relation).

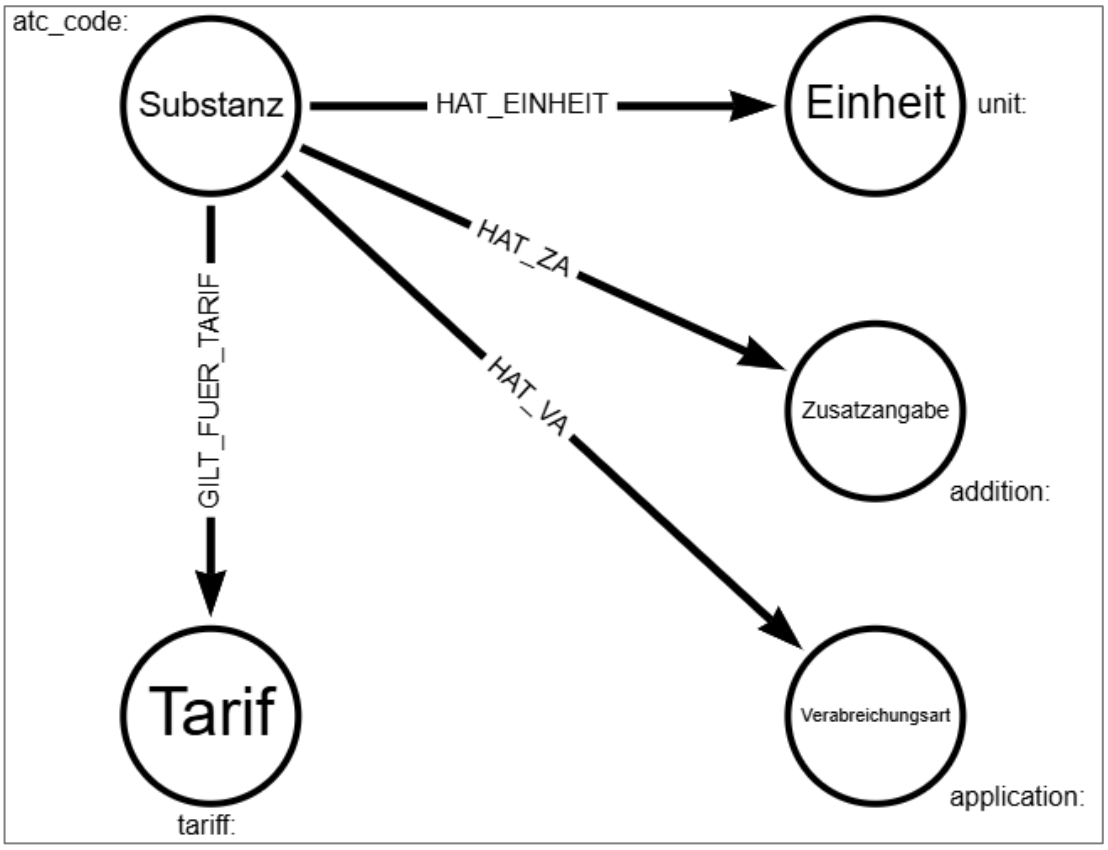


Abbildung 14: Skizze des Datenmodells der SwissDRG «Liste der hochteuren Medikamente/Substanzen»

Die Medikamenten-Liste steht als Pipe-getrennte csv-Datei auf der Website der SwissDRG AG zum Download zur Verfügung [6,9]. Dadurch konnten die Daten des Jahres 2025 mit folgendem Cypher-Code direkt in Neo4j hineingeladen werden:

```
WITH 'https://www.swissdrg.org/download_file/view/4992/2253' AS Import
LOAD CSV WITH HEADERS FROM Import AS row FIELDTERMINATOR '|'
```

Mehrfachinformationen wurden zunächst in einem WITH-Statement und CASE-WHEN sowie der Split-Methode aufgeteilt. Die Substanz-Knoten wurden danach mit einem Merge-Statement erstellt:

```
MERGE (s:Substanz {Name: row.atc_code})
    SET
    s.Bezeichnung = row.compound_de,
    s.`Bezeichnung FR`=row.compound_fr,
    s.`Einschränkung`=row.constraints_de,
    s.`Einschränkung FR`= row.constraints_fr,
    s.`Einschränkung IT`= row.constraints_it,
    s.`Zu erfassen seit`= toInteger(row.since)
```

Im Anschluss konnten die weiteren Knoten und deren Relationen mit MERGE-Statements erstellt werden, indem die Mehrfachinformationen innerhalb eines FOREACH ausgeführt wurden. Zudem sind Constraints für die Substanz-Namen und den ATC-Code erstellt worden. Mit diesem Modellierungsschritt im Skript «01_liste_hochteure_medikamente.cypher» konnten fünf Labels und vier Relationen erstellt werden. Es wurden Tests durchgeführt und dokumentiert.

4.2.4 Integration des «Technischen Begleitblattes»

Das technische Begleitblatt enthält die Bezeichnungen zu den Abkürzungen der «Liste der hochteuren Medikamente/Substanzen». Einheiten, Verabreichungsarten und Zusatzangaben sind in drei Landessprachen vorhanden. Das Begleitblatt besteht seit 2025 als Excel-Datei. Zuvor waren jeweils drei PDF-Dateien für drei Landessprachen vorhanden.

Die im Preprocessing aufbereiteten Daten wurden mit dem Cypher-Skript «02_technisches_begleitblatt.cypher» in Neo4j hineingeladen. Die Bezeichnungen wurden je Sprache mit Hilfe einer UNWIND Anweisung separiert weiterverarbeitet und dreisprachige Properties bei den Knoten der «Verabreichungsart» (siehe Code unten) und «Einheit» hinzugefügt.

```
UNWIND [
  {lang: 'de', suffix: 'Bezeichnung'},
  {lang: 'fr', suffix: 'Bezeichnung FR'},
  {lang: 'it', suffix: 'Bezeichnung IT'}
] AS langInfo
LOAD CSV WITH HEADERS FROM 'file:///technisches_begleitblatt.csv' AS row
MATCH (n:Verabreichungsart {Name: row.kuerzel})
WHERE langInfo.lang = row.sprache
SET n[langInfo.suffix] = row.kuerzel_bezeichnung
```

Bei den Knoten der «Zusatzangabe» wurden die deutsche Bezeichnung sowie «Hinweise» als Properties ergänzt. Die Knoten der «Einheit» erhielten das Property «Hinweis», falls es vorhanden ist. Dokumentierte Tests schlossen dieses kleine Arbeitspaket ab. Es wurden keine neuen Knoten oder Relationen, sondern nur Properties erstellt.

4.2.5 Integration der Zusatzentgelte

Da die Zusatzentgelte grundsätzlich sehr ähnliche Informationen besitzen wie die hochteuren Medikamente, konnte das Datenmodell auf ähnliche Art und Weise erstellt werden. Die Relationen der Zusatzentgelte zu den Knoten «Einheit, Zusatzangabe, Verabreichungsart und Tarif» heissen genauso, wie die Relationen der «Substanz» zu genau diesen Knoten. Eine Verbindung zwischen den Knoten «Substanz» und «Zusatzentgelt» ist mit Hilfe des ATC-Codes erstellt worden.

In den Skizzen des Datenmodells auf den folgenden Seiten werden neue Knoten oder Relationen in schwarz dargestellt. Knoten und Relationen, die schon zuvor kreiert wurden, sind grau eingefärbt.

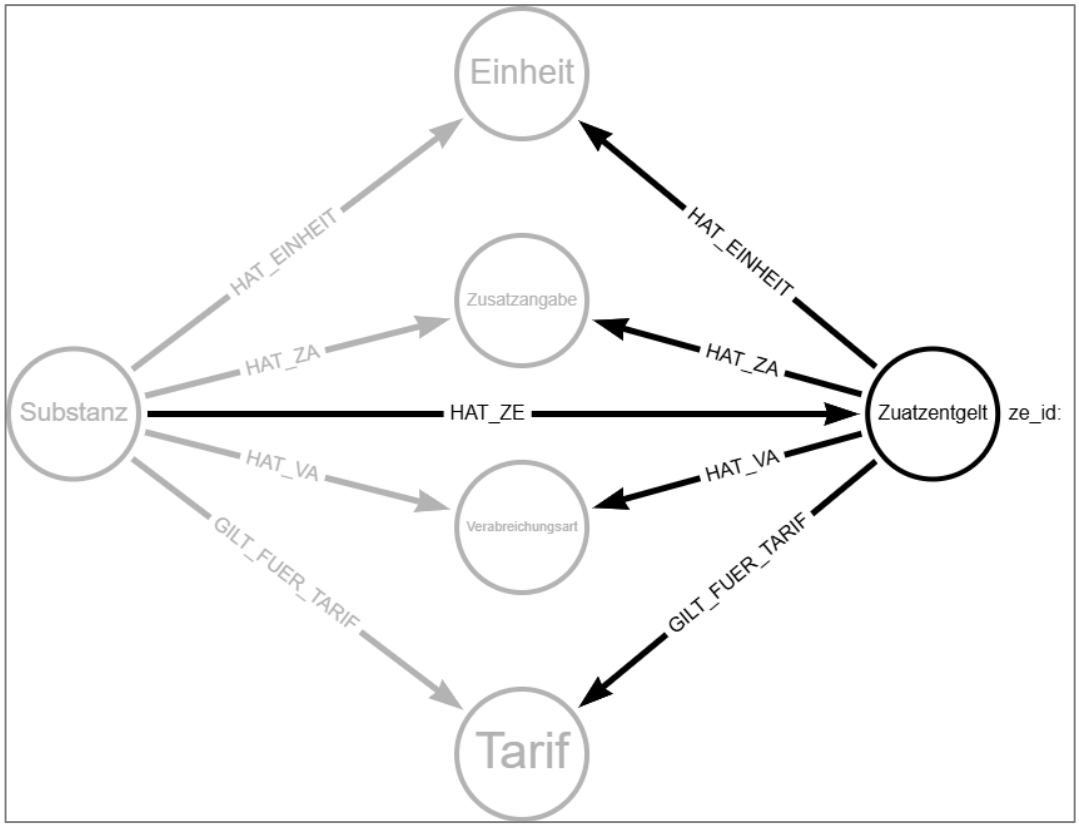


Abbildung 15: Skizze des Datenmodells der SwissDRG Zusatzentgelte

Mit dem Skript «03_zusatzentgelte.cypher» wurden die Zusatzentgelte in Neo4j importiert und die Knoten «Zusatzentgelt» erstellt. Neben einigen Stammdaten wie der ZE-Nummer, Version etc. wurden die ATC-Codes, Einheiten, Verabreichungsarten und Zusatzangaben ebenfalls als Property hinzugefügt. Dies ermöglicht es, die Relationen zu den Knoten «Substanz, Einheit, Verabreichungsart und Zusatzangabe» zu generieren, ohne die Datenquelle nochmals importieren zu müssen. Folgender Cypher-Code zeigt die Erstellung der Relation «HAT_ZE»:

```
MATCH (z:Zusatzentgelt)
MATCH (s:Substanz)
WHERE z.ATC = s.ATC
MERGE (s)-[r:HAT_ZE]->(z)
```

Durch diesen Modellierungsschritt wurde dem Datenmodell ein neues Label und eine neue Relation hinzugefügt. Vier bestehende Relationen sind nun mehrfach vorhanden.

4.2.6 Integration der Daten der European Medicines Agency (EMA)

Die Europäische Arzneimittelagentur EMA stellt eine täglich aktualisierte Excel-Liste mit Medikamenten-Produkten, deren MeSH-Terms sowie weiteren Stammdaten zum Download zur Verfügung. MeSH steht für «Medical Subject Headings» und wird von der «National Library of Medicine» der Vereinigten Staaten von Amerika gepflegt. Dabei handelt es sich um standardisierte medizinische Bezeichnungen, die zudem als Ontologie verfügbar sind [51,57]. Die MeSH-Terms in englischer Sprache sind im Graph-Datenmodell eine der beiden Quellen für die Indikation eines Medikaments. Die Daten der EMA werden im Preprocessing als csv-Datei aufbereitet.

Das Skript «04_ema_products.cypher» erstellt die Knoten mit dem Label «Produkt» in Neo4j mit den Properties MeSH-Terms, Indikation, Zulassungsinhaberin, ATC-Code und weiteren Informationen.

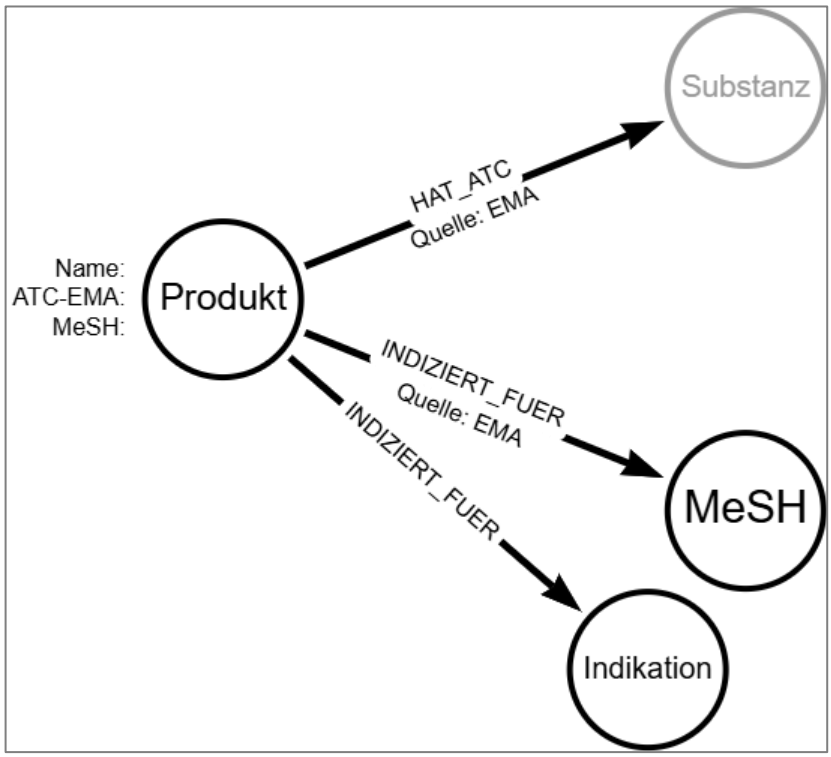


Abbildung 16: Skizze des Datenmodells der Medikamente gemäss der Europäischen Arzneimittelagentur (EMA)

Für die Relation zu «Substanz» ist der ATC-Code der EMA genutzt worden. Dabei wurde die Quellenangabe («EMA») als Relationenproperty ebenfalls kreiert. Je Produkt können mehrere MeSH-Terms vorhanden sein. Die MeSH-Terms wurden mit einem UNWIND-Statement aufgeteilt (mit Semikolon als Trennzeichen). Die Relation zwischen «Produkt» und «MeSH» heisst «INDIZIERT_FUER», wie der folgende Code verdeutlicht.

```
MATCH (n:Produkt)
UNWIND (split(lower(n.Mesh), ';')) as mesh_terms
MERGE (m:MeSH {Name: mesh_terms})
MERGE (n)-[:INDIZIERT_FUER {Quelle: 'EMA'}]->(m)
```

Auch hier findet man die «Quelle» als Relationenproperty. Die Knoten mit dem Label «MeSH» erhielten zusätzlich noch das Label «Indikation», sodass die MeSH-Terms der EMA und die Anwendungsgebiete der Stiftung Refdata im folgenden Schritt vereint werden konnten.

4.2.7 Integration der Daten von Refdata (SwissMedic)

Die Stiftung Refdata hat gemäss Heilmittelgesetz den Auftrag, Arzneimittelinformationen zur Verfügung zu stellen und die Fachinformationen der SwissMedic zu publizieren [58]. Neben der Webapplikation stehen Download-Dateien der Fachinformation sowie seit 2021 strukturierte Arzneimittelinformationen (SAI) als XML-Dateien zur Verfügung [59]. Die SAI-Daten werden monatlich aktualisiert und enthalten nicht nur die XML-Dateien, sondern auch die XDS-Schema-Dateien (aktuell in der Version 3.2) [60].

Die «SAI_PRAEPARATE» sind dabei vergleichbar mit den Produkten der EMA. Deshalb wurden für Präparate ebenfalls neue Produkt-Knoten erstellt oder diese in bestehende Knoten eingefügt. Als Identifikator dient grundsätzlich ein kurzer Produktname. Diese Kurznamen sind nicht eindeutig und können unterschiedliche Indikationen haben, sodass ein Knoten eines Kurznamens mit anderen Produkten mit «langen» Namen über die Relation «IST_AUCH_PRODUKT» verbunden sein muss. Die Relation zur Substanz («HAT_ATC») wurden danach mit dem ATC-Code erstellt, der sich teilweise vom ATC-Code der EMA oder der «Liste der hochteuren Medikamente/Substanzen» unterscheidet.

Die Knoten «Anwendungsgebiet» erhielten ebenfalls das Label «Indikation» und bestehen aus deutschen oder französischen Freitexten (Strings), die mit Semikolon voneinander getrennt sind. Die Anwendungsgebiete wurden (ähnlich wie die MeSH-Terms) mit einer UNWIND-Anweisung aufgesplittet. Die Relation «INDIZIERT_FUER» mit dem Property «Quelle: SAI» stellt sie Verbindung zwischen «Produkt» und «Indikation» her. Bei den Anwendungsgebieten handelt es sich um nicht-standardisierte Begriffe, sodass die Vielfalt/Anzahl der Indikationsknoten erheblich stieg.

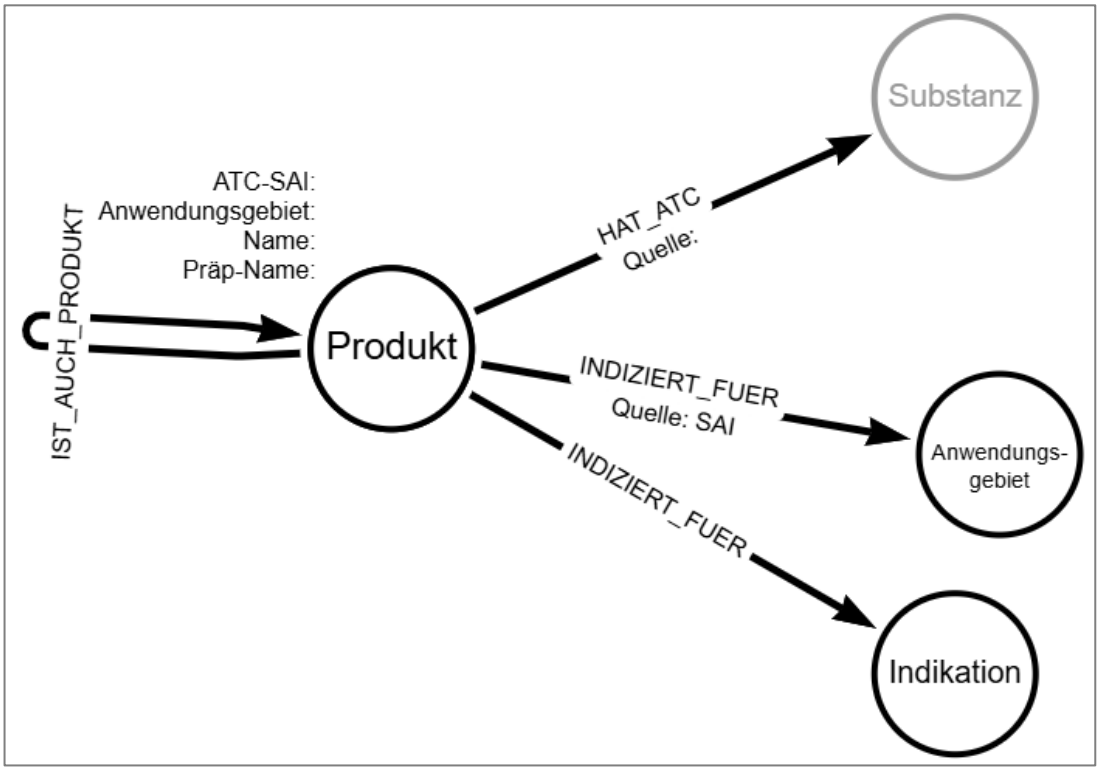


Abbildung 17: Skizze des Datenmodells der Präparate (Produkte) gemäss Stiftung Refdata

Zudem wurden im Skript «05_sai_praeparate.cypher» nicht nur Relationen zwischen den Produkt- und Indikationsknoten, sondern bei Mehrfach-Produkten auch «indirekte» Relationen erstellt und doppelte Relationen entfernt. Das folgende Cypher-Statement verdeutlicht die Erstellung der «INDIZIERT_FUER» Relation mit dem Property «Quelle: Ist auch Produkt» für Produkte, deren Kurzname nicht eindeutig ist.

```
MATCH (i:Indikation)<-[:INDIZIERT_FUER]-(p:Produkt)-[:IST_AUCH]->(d:Produkt)
MERGE (d)-[:INDIZIERT_FUER {Quelle: 'Ist auch Produkt'}]->(i)
```

Mit diesen Mechanismus soll sichergestellt werden, dass alle Produkte mit den Indikationen verbunden werden können. Die Begriffe Produkt und Präparat werden hier synonym verwendet.

4.2.8 Integration des ATC/DDD-Index der WHO

Die «Liste der hochteuren Medikamente/Substanzen» enthält nur den Teil der ATC-Codes, der für die stationären Tarife in der Schweiz relevant ist. Weitere ATC-Codes und die Angaben über die «defined daily dose» (DDD) sind im ATC/DDD-Index des WHO Collaborating Centre for Drug Statistics Methodology vorhanden [8].

Der ATC/DDD-Index kann auf einer Website abgerufen werden. Mit Hilfe des Python-Skriptes «06_parse_atc_ddd.py» wurden die ATC-Hierarchie und die DDD-Angaben im Rahmen des Preprocessings als csv-Dateien zusammengestellt.

Die ATC-Hierarchie von ein- bis fünfstelligen ATC-Codes wurde in Neo4j mit dem Label «Hierarchie» abgebildet. Die Kinderelemente und deren Elternknoten sind mit der Relation «IST_IN_ATC_GRUPPE»

verbunden worden. Die ATC-Codes mit sieben Ziffern sollten den Substanzen der Medikamentenliste gleichgestellt werden, sodass weitere Knoten des Typs «Substanz» entstanden. Die Substanzen des ATC/DDD-Index enthalten nur die englischen Substanzbezeichnungen und Angaben über die «defined daily dose» (DDD). Die englischen Bezeichnungen und DDD-Angaben wurden den bereits vorhandenen Substanzknoten der «Liste der hochteuren Medikamente/Substanzen» hinzugefügt.

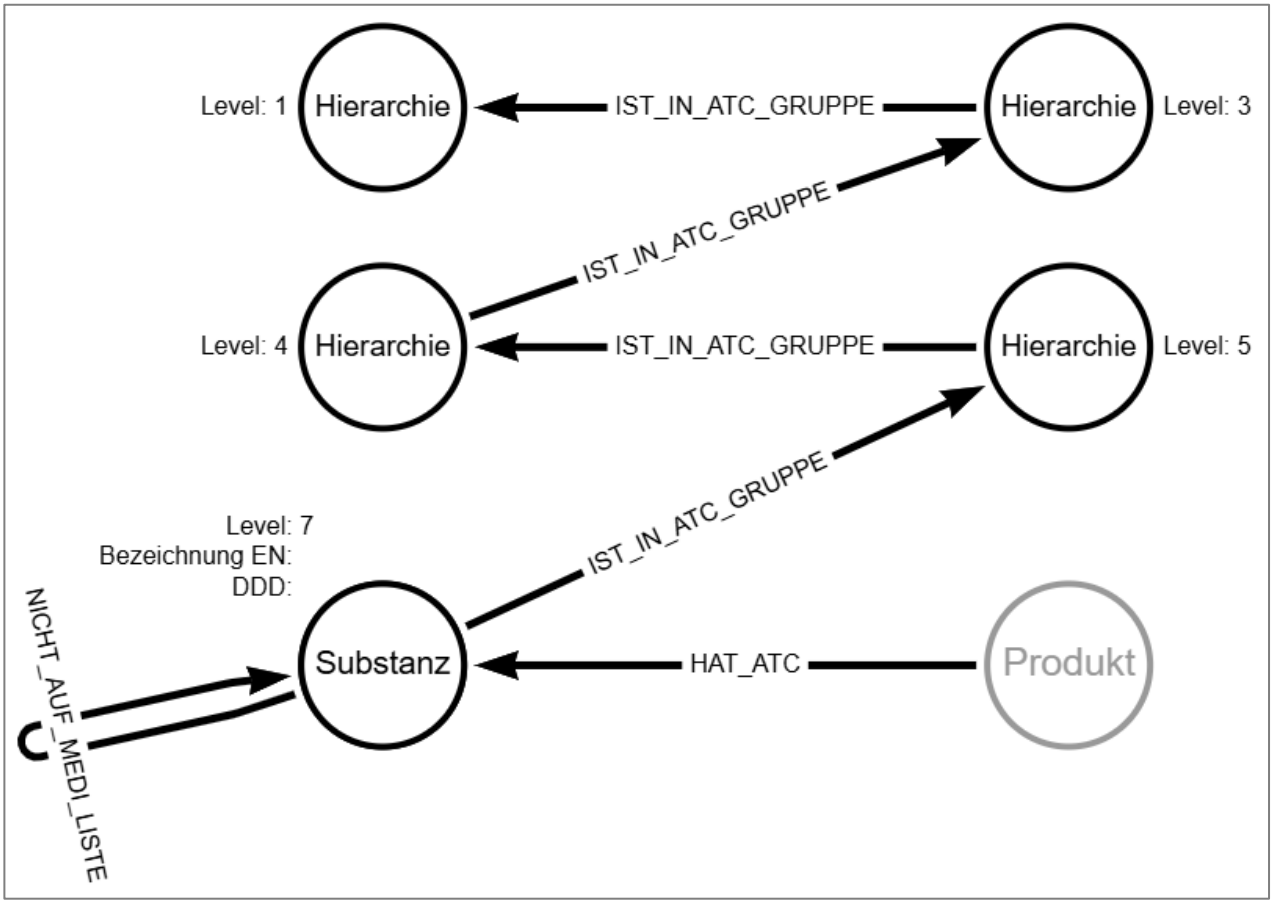


Abbildung 18: Skizze des Datenmodell für den ATC/DDD-Index der WHO

Zudem mussten die Knoten «Produkt» und «Substanz» verbunden werden. Ähnlich wie zuvor konnte die Relation «HAT_ATC» über die ATC-Codes hergestellt werden. Dabei wurde «Produkt» entweder mit Hilfe des ATC-Codes der EMA oder der Stiftung Refdata mit «Substanz» verbunden. Falls sich eine Substanz nicht auf der «Liste der hochteuren Medikamente/Substanzen» befindet, ist die Relation «NICHT_AUF_MEDI_LISTE» zu sich selbst erstellt worden.

Das Cypher-Skript «06_who_atc.cypher» erschafft das oben skizzierte Datenmodell. Die «IST_IN_ATC_GRUPPE» Relationen sind über den String-Abgleich der Knoten-Namen generiert worden, wie das folgende Beispiel von Level 4 zu Level 3 verdeutlicht:

```
MATCH (l:Hierarchie {Level: 4})
MATCH (h:Hierarchie {Name: substring(l.Name,0,3)})
MERGE (l)-[:IST_IN_ATC_GRUPPE]->(h)
```

Zusätzlich wurden in diesem Modellierungsschritt die Substanzknoten gekennzeichnet, die sich in früheren Jahren auf der «Liste der hochteuren Medikamente/Substanzen» befunden haben. Dafür wurde das Property «War auf Medi-Liste bis und mit Jahr» gemäss der manuell erstellten csv-Datei «geloeschte_substanzen.csv» eingefügt.

4.2.9 Integration der Spezialitätenliste des Bundesamts für Gesundheit

Das Bundesamt für Gesundheit (BAG) stellt monatliche Updates der Spezialitätenliste (SL) auf der Website spezialitaetenliste.ch zur Verfügung. Neben der Websuche ist ein Download der Daten möglich. Die Daten befinden sich auf Ebene der Packung und somit auf einem genaueren Detaillierungsgrad im Vergleich zu den Präparaten der Stiftung Refdata.

Im Cypher-Skript «07_bag_sl.cypher» wurden die Zeilen der csv-Datei in die Graph-Datenbank hineingeladen und pro Packung ein Knoten mit dem Label «Spezialitätenliste» erstellt. Als Properties erhielten die Knoten neben der Bezeichnung zusätzlich den ATC-Code, die Zulassungsnummer, die GTIN, die letzte Preisänderung als Datum sowie den ExFactory- und Publikumspreis. Oft wurde für ein Produkt mehrere Knoten «Spezialitätenliste» erstellt, weil die Packungsgrössen der SL in die Graph-Datenbank mit übernommen wurden.

Abbildung 19: Skizze des Datenmodells für die Spezialitätenliste des BAG

Die Produkte (Präparate) von Refdata enthalten die fünfstellige Zulassungsnummer von SwissMedic. Diese Zulassungsnummer konnte verwendet werden, um die Relation der Knoten «Produkt» zu den Packungen der SL herzustellen («IST_AUF_SL»).

```
MATCH (n:Produkt)
MATCH (s:Spezialitätenliste)
WHERE n.Zulassungsnummer = s.Zulassungsnummer
MERGE (n)-[:IST_AUF_SL]->(s)
```

Ausserdem enthält die Spezialitätenliste den aktuellen ATC-Code. Diese zusätzliche Information wurde genutzt, um die Knoten «Produkte» und «Substanzen» zu verbinden, sofern dies mit den ATC-Codes der EMA oder Refdata bisher nicht möglich war. Falls der ATC-Code der SL mit ATC-Codes des Labels «Substanz» übereistimmte, so ist die Relation «HAT_ATC» mit dem Property «Quelle = SL» erstellt worden. Der folgende Cypher-Code verdeutlicht diese indirekt hergestellte Relation:

```
MATCH (s:Substanz)
MATCH (p:Produkt)-[:IST_AUF_SL]->(l:`Spezialitätenliste`)
WHERE l.`ATC-SL` = s.Name
AND NOT EXISTS ((p)-[]->(s))
MERGE (p)-[:HAT_ATC {Quelle: 'SL'}]->(s)
```

4.2.10 Integration der Änderungen von ATC-Codes

ATC-Codes können sich über die Jahre verändern. Auf der Website des ATC/DDD-Index werden diese «alterations» dokumentiert und veröffentlicht [61].

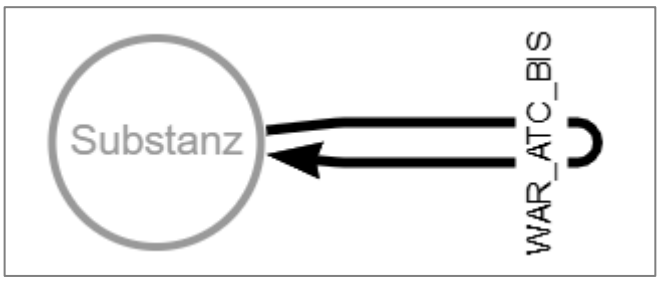


Abbildung 20: Skizze des Datenmodells für die ATC-Codes, die sich über die Jahre hinweg geändert haben

Geänderte Substanzen-Codes wurden als Relation zu sich selbst modelliert. Die Relation heisst «WAR_ATC_BIS» und enthält die beiden Properties «Gültig bis und mit Jahr» als Zahl und «Früherer ATC». Zusätzlich wurde die Relation «HATTE_ATC» zwischen «Produkt» und «Substanz» erstellt, wenn die ATC-Codes der EMA oder Refdata mit dem früheren ATC-Code übereinstimmen und bisher keine Verbindung zwischen Produkt und Substanz existierte. Bei der Relation «HATTE_ATC» handelt es sich somit wieder um eine «indirekte» Relation.

4.2.11 Integration der Anträge aus dem SwissDRG Antragsverfahren

Die öffentlichen Antragsdaten stehen im Antragsportal der SwissDRG AG als Download zur Verfügung.

Das Cypher-Skript «09_antraege.cypher» erstellte je Antrag einen Knoten «Antrag» mit den Properties Antragsnummer, Antragsjahr und dem Partner, der den Antrag eingereicht hatte. Die Relation «HATTE_ANTRAG» zwischen den Anträgen und dem Label «Substanz» wurde mit Hilfe des aktuellen ATC-Codes sichergestellt.

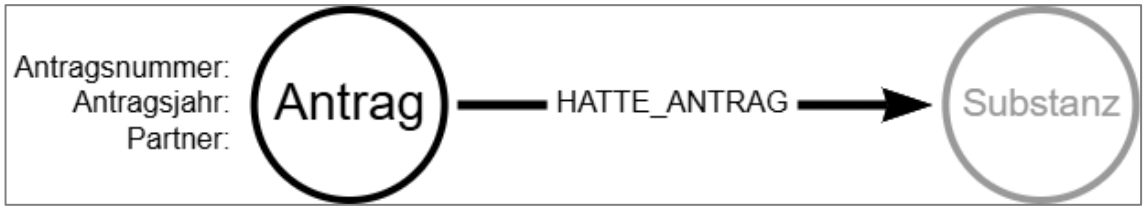


Abbildung 21: Skizze des Datenmodells für die Anträge des SwissDRG Antragsverfahrens

4.3 Datenmodell

Die Einzelschritte zum Erstellen des Datenmodells wurden im Gesamtskript «99_load_all_into_neo4j.cypher» zusammengefasst. Das finale Datenmodell kann folgendermassen dargestellt werden.

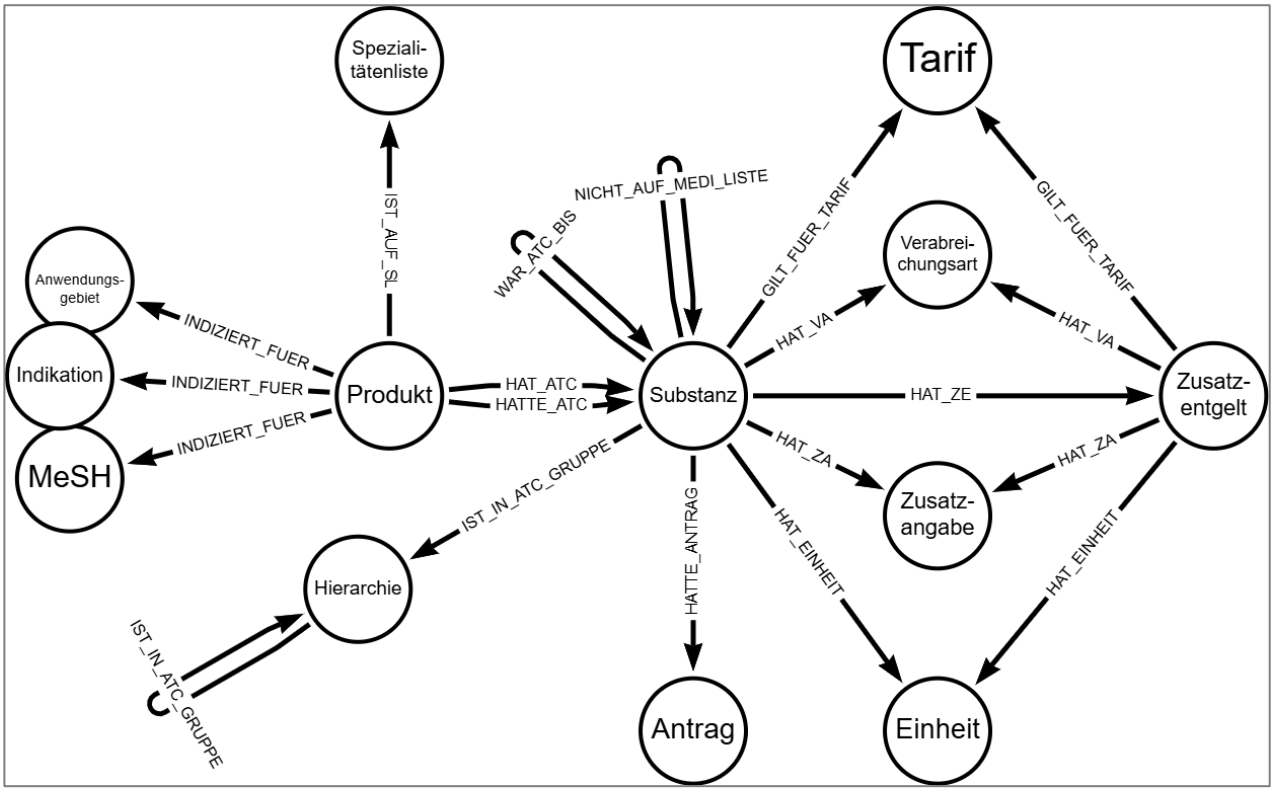


Abbildung 22: Datenmodell

4.3.1 Knoten

Das Datenmodell besteht aus 13 Labels (Kreise in der Abbildung 22), die zusammen über 27'000 einzelne Knoten ausmachen. Bei den Knoten handelt es sich um:

- Substanz: basierend auf den siebenstelligen ATC-Codes der SwissDRG Liste der hochteuren Medikamente/Substanzen sowie dem ATC-Index der WHO (inklusive Angabe der Definded Daily Dose (DDD))
- Tarif, Einheit, Verarbeitungsart und Zusatzangabe: gemäss der SwissDRG «Liste der hochteuren Medikamente/Substanzen» und dem dazugehörigen technischen Begleitblatt
- Zusatzentgelte: stationäre Zusatzentgelte für die Tarife SwissDRG, TARPSY und ST Reha
- Anträge: abgelehnte oder nicht rechenbare Anträge gemäss SwissDRG Antragsverfahren für Medikamente oder Zusatzentgelte
- Hierarchie: basierend auf dem ATC-Index mit den Detailierungslevels 1, 3, 4 und 5
- Produkt: basiert auf den Produktnamen der EMA oder der Stiftung Refdata ohne Unterteilung nach Packungen
- Spezialitätenliste: alle Packungen inkl. Preisangaben der Spezialitätenliste (SL) des Bundesamts für Gesundheit (BAG)
- Indikation, MeSH, Anwendungsgebiet: Diese drei Labels geben an, für welche Erkrankungen die Produkte gemäss EMA (gespeichert im Label «MeSH») oder Refdata (gespeichert im Label «Anwendungsgebiet») zugelassen sind. Die Knoten basieren auf den Indikationstexten. Mehrere semikolongetrennte Indikationen wurden in mehrere Knoten aufgeteilt. Kommagetrennte Indikationen werden jedoch nicht separiert, sondern bleiben als ein gemeinsamer Knoten bestehen. Das Label «Indikation» ist die Vereinigungsmenge von «MeSH» und «Anwendungsgebiet».

Alle Labels haben Properties, in denen Detailinformationen gespeichert sind.

Tabelle 5: Labels und deren Properties

Node Labels	Property Name	Property Typen
Substanz	Name	String
Substanz	Zu erfassen seit	Long
Substanz	Einschränkung	String
Substanz	Einschränkung FR	String
Substanz	Einschränkung IT	String
Substanz	url	String
Substanz	DDD	StringArray, Double
Substanz	Unit	StringArray, String
Substanz	Level	Long
Substanz	Route	StringArray, String
Substanz	Note	StringArray, String
Substanz	Bezeichnung	String
Substanz	Bezeichnung EN	String
Substanz	Bezeichnung FR	String
Substanz	War auf Medi-Liste bis und mit Jahr	Long
Tarif	Name	String
Zusatzangabe	Name	String
Zusatzangabe	Bezeichnung	String
Zusatzangabe	Hinweis	String
Verabreichungsart	Name	String
Verabreichungsart	Bezeichnung	String
Verabreichungsart	Bezeichnung FR	String
Verabreichungsart	Bezeichnung IT	String
Einheit	Name	String
Einheit	Bezeichnung	String
Einheit	Bezeichnung FR	String
Einheit	Bezeichnung IT	String
Einheit	Hinweis	String
Zusatzentgelt	Name	String
Zusatzentgelt	ATC	String
Zusatzentgelt	Version	String
Zusatzentgelt	ZE-Code	String
Zusatzentgelt	Zusatzangabe	String
Zusatzentgelt	Einheit	String
Zusatzentgelt	Verabreichungsart	String

Zusatzentgelt	ZE-Nr	String
Zusatzentgelt	url	String
Produkt	Name	String
Produkt	Mesh	String
Produkt	Indikation	String
Produkt	Zulassungsinhaberin	String
Produkt	Orphan Status	String
Produkt	Generikum	String
Produkt	Biosimilar	String
Produkt	url	String
Produkt	IT	String
Produkt	ATC-EMA	String
Produkt	Anwendungsgebiet	String
Produkt	Erstzulassung	String
Produkt	Zulassungsnummer	String
Produkt	ATC-SAI	String
Produkt	Aktive Substanz EMA	String
Produkt	Praeparatename	String
Indikation, MeSH	Name	String
Indikation, Anwendungsgebiet	Name	String
Hierarchie	Name	String
Hierarchie	url	String
Hierarchie	Bezeichnung EN	String
Hierarchie	Level	Long
Spezialitätenliste	Name	String
Spezialitätenliste	GTIN	String
Spezialitätenliste	Zulassungsnummer	String
Spezialitätenliste	ExFactoryPreis	Double
Spezialitätenliste	PublikumsPreis	Double
Spezialitätenliste	Letzte Preis Änderung	Date
Spezialitätenliste	ATC-SL	String
Antrag	Tarifsystem	String
Antrag	Partner	String
Antrag	Antragsjahr	Long
Antrag	Status	String
Antrag	Antragsnummer	String

4.3.2 Relationen

Die 13 Labels sind mit 14 unterschiedlichen Relationen untereinander verbunden. Einige Relationen verbinden unterschiedliche Labels, sodass teilweise Mehrfachrelationen entstehen.

- GILT_FUER_TARIF: Relation einer Substanz der Medi-Liste oder eines Zusatzentgeltes zum Tarif (zweifache Relation)
- HAT_EINHEIT: Relation einer Substanz der Medi-Liste oder eines Zusatzentgeltes zur Einheit (zweifache Relation)
- HAT_VA: zweifache Relation einer Substanz der Medi-Liste oder eines Zusatzentgeltes zur Verabreichungsart (VA)
- HAT_ZA: zweifache Relation einer Substanz der Medi-Liste oder eines Zusatzentgeltes zur Zusatzangabe (ZA)
- HAT_ZE: Relation einer Substanz der Medi-Liste zu einem Zusatzentgelt (ZE)
- HATTE_ANTRAG: Relation einer Substanz der Liste der hochteuren Medikamente zu Anträgen des Antragsverfahrens
- WAR_ATC_BIS: Relation einer Substanz zu sich selbst, falls sich der ATC-Code in der Vergangenheit geändert hat (inkl. Relationenproperties)
- NICHT_AUF_MEDI_LISTE: Relation einer Substanz zu sich selbst, falls es sich um einen ATC-Code handelt, der sich aktuell nicht auf der Liste der hochteuren Medikamente befindet
- IST_IN_ATC_GRUPPE: Relation einer Substanz zum ATC-Hierarchie-Level 5 oder vom ATC-Hierarchie-Level jeweils ein Hierarchie-Level höher (vom Kind- zum Eltern-Knoten)
- HAT_ATC: Relation eines Produktes (Präparates) zu einer Substanz, wobei ATC-Codes der EMA und/oder von Refdata/SAI und/oder von der SL zu dieser Relation beitragen. Die Quelle der Relation ist als Relationenproperty vorhanden.
- HATTE_ATC: Relation eines Produktes zu einer Substanz, wobei der ATC-Code zu einem früheren ATC-Code gehört, der in der Relation WAR_ATC_BIS hinterlegt ist
- INDIZIERT_FUER: Relation zwischen den Produkten und den Indikationsknoten (dreifache Relation)
- IST_AUF_SL: Relation eines Produktes zu einer konkreten Packung, falls sich diese auf der Spezialitätenliste befindet
- IST_AUCH_PRODUKT: Relation eines Produktes zu sich selbst, um Produkte mit gleichem Namen, aber unterschiedlichen Indikationen zu berücksichtigen. Diese Relation stellt sicher, dass keine Daten verloren gehen und alle Verknüpfungen aufgebaut werden.

Bei fünf Relationen gibt es Properties, aber die Mehrheit der Relationen hat keine Properties.

Tabelle 6: Relationen und deren Properties

Relation	Property Name	Property Typ
HAT_ZA	null	null
HAT_VA	null	null
HAT_EINHEIT	null	null
HAT_ZE	null	null
INDIZIERT_FUER	Quelle	String
IST_AUF_SL	null	null
GILT_FUER_TARIF	null	null
IST_AUCH_PRODUKT	null	null
IST_IN_ATC_GRUPPE	null	null
WAR_ATC_BIS	Gültig bis und mit Jahr	Long
WAR_ATC_BIS	Früherer ATC	String
HAT_ATC	Quelle	String
NICHT_AUF_MEDI_LISTE	null	null
HATTE_ANTRAG	Art	String
HATTE_ATC	Quelle	String

4.3.3 Perspektive für Neo4j Explore/Bloom

Die Auftraggeberin und weitere Anwender können das Datenmodell auf zwei Arten nutzen bzw. analysieren. Erstens steht Neo4j Query zur Verfügung, was die Bezeichnung des Neo4j Browsers für die Cloud ist. Zweitens kann Neo4j Explore genutzt werden, was die Cloud-Variante von Neo4j Bloom darstellt.

Für das Tool Explore ist eine Perspektive erstellt worden, die in die Cloud-Konten der Anwender importiert wurde. Die Perspektive findet man im GitLab-Repository unter

```
src/main/perspectives
```

mit der Bezeichnung `Medi-Graph V1.0.json`.

Die Perspektive enthält elf Labels und alle Relationen. Die beiden Labels «MeSH» und «Anwendungsgebiete» sind ausgeblendet worden, weil es sich dabei nur um Teilgebiete des Labels «Indikation» handelt.

Ausserdem sind regelbasierte Anpassungen im Design der Knoten vorgenommen worden, um die User-Experience zu verbessern.

- Das Label «Hierarchie» hat einen Farbübergang von dunkelblau (für Level 1) bis hellblau (für Level 5) erhalten.
- Das Label «Substanz» wird grau eingefärbt, wenn das Property «War auf Medi-Liste bis und mit Jahr» vorhanden ist. Damit werden Substanzen, von der Medi-Liste gelöscht worden sind, gekennzeichnet. Nicht gelöschte Substanzen behalten die Farbe Gelb.
- Das Label «Spezialitätenliste» hat einen Farbübergang von hellorange (für geringe ExFactory-Preise), über orange (für ExFactory-Preise von rund 500 CHF) bis dunkelorange (für sehr hohe ExFactory-Preise) erhalten (siehe folgende Abbildung).

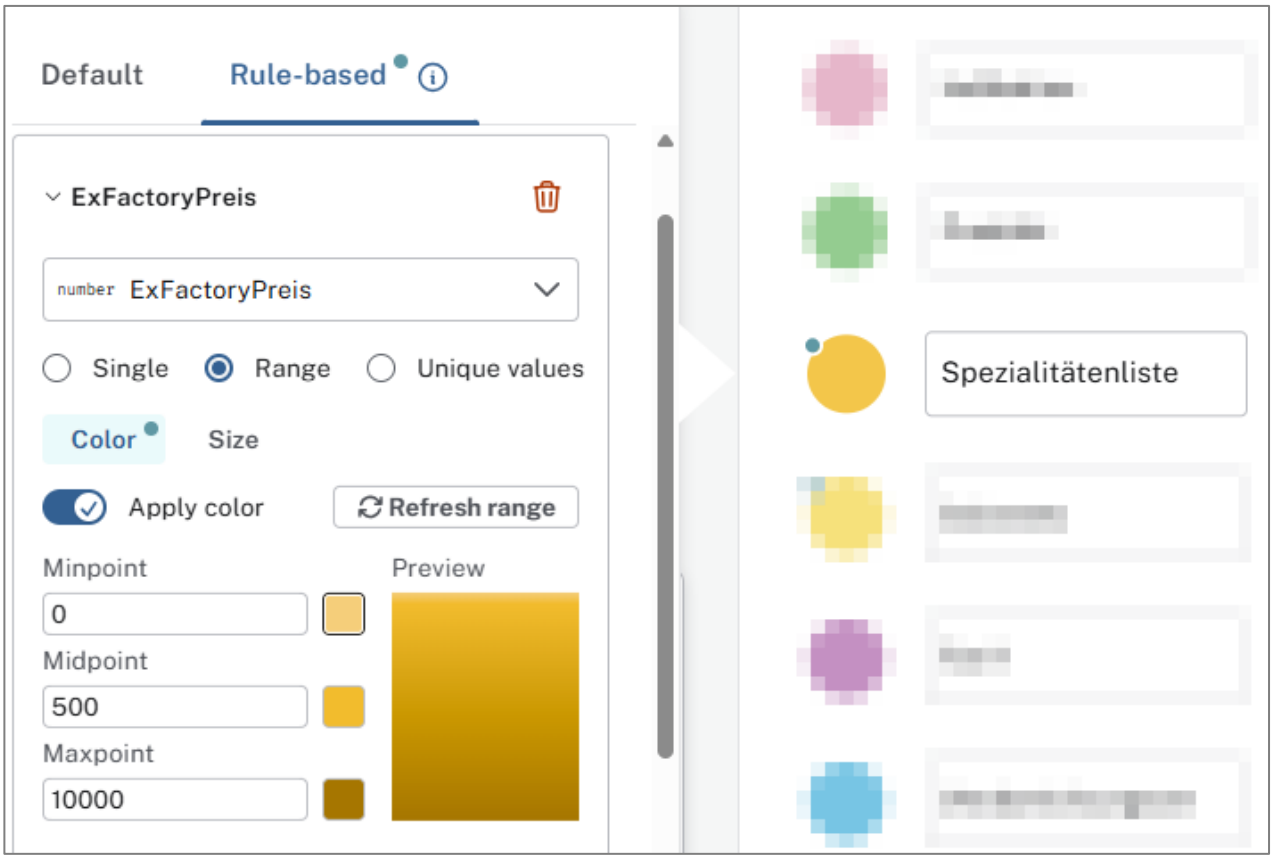


Abbildung 23: Bedingte Formatierung des Labels "Spezialitätenliste" (Farbübergang)

Um einige weitergehende Analysefunktionen von Explore nutzen zu können, sind sog. «Saved Cypher» in die Perspektive eingebaut worden. Dabei handelt es sich um fünf «Search phrases» und zwei «Scene actions». Die «Saved Cypher» ermöglichen es der Auftraggeberin, standardisierte aber komplexe Cypher-Queries entweder mit Parametern (Variablen) zu starten oder von einzelnen Knoten aus die Abfragen zu beginnen. Dadurch sind weniger Klicks in Explore nötig, weil beispielweise mehrere Relationen auf einmal eingeblendet werden können.

Es wurden «Search phrases» implementiert für folgende Abfragen, wobei das Wort hinter dem Dollarzeichen den Parameter repräsentiert.
- Diverse Informationen für Substanz mit $ATC
- Diverse Informationen für Produkte mit $Indikation
- Substanzen, ZE und Produkte für $Indikation (siehe Abbildung 27 in Abschnitt 5.1)
- WHO ATC-Hierarchie mit Substanz und ZE für $ATC
- Zeige Substanzen der aktuellen Medi-Liste

Basierend auf den «Search phrases» ist ein erfolgreicher Test mit Deep-Links durchgeführt worden. Mit Hilfe von Deep-Links können Parameter an «Search phrases» übergeben und von anderen Applikationen zur Perspektive nach Neo4j Explore abgesprungen werden.

```
https://console-
preview.neo4j.io/tools/explore?search=Diverse%20Informationen%20für%20Substanz%20mit%20L01FY01&perspective=Medi-Graph%20V1.0&run=true
```

Der Parameter hinter «search» verweist auf die «Search phrase», «perspective» definiert die zu öffnende Perspektive und «run» steuert, ob die Suche tatsächlich ausgeführt oder nur vorbereitet wird.

Bei den «Scene actions» handelt es sich um die folgenden beiden Abfragen:
- WHO ATC-Hierarchie einer Substanz anzeigen
- Diverse Relationen eines Produktes anzeigen (siehe folgende Abbildung)

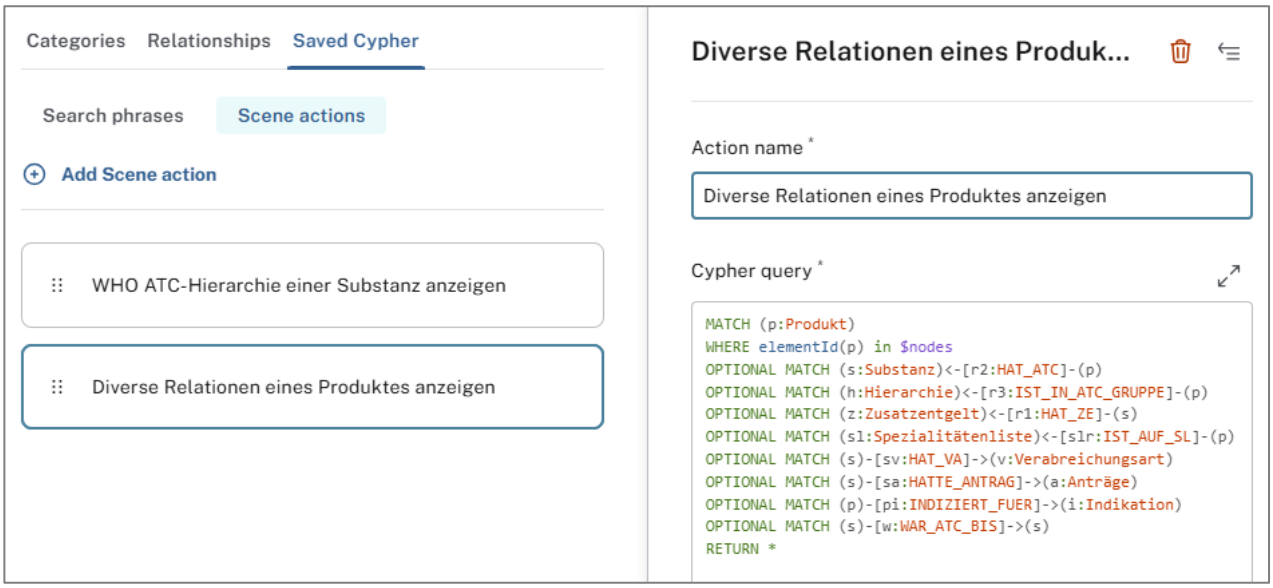


Abbildung 24: Beispiel einer "Saved Cypher" Abfrage

Die Anwender können beispielweise mit «Diverse Relationen eines Produktes anzeigen» einen Rechtsklick auf einen Produktknoten durchführen und über ein Kontextmenü die «Scene actions» ausführen. Dann würde zu diesem Produkt ein Netzwerk mit den Knoten «Substanz, Hierarchie, Zusatzentgelt, Spezialitätenliste, Verabreichungsart, Anträge und Indikation» sowie deren Relationen angezeigt werden.

4.4 Testing

4.4.1 Testing des Codes

Gemäss Test-Konzept gab es zwei zentrale Elemente des Testings. Zuerst sind nur wenige Daten in die Testdatenbank geschrieben worden. Das erwartete Ergebnis ist analysiert und bei Bedarf angepasst worden. Die Zwischenergebnisse sind stichpunktartig und mit Screenshot dokumentiert worden. Die testbezogenen Dokumente befinden sich im GitLab-Repository im Ordner

```
src/test
```

Hiernach sind die Knoten und Kanten erst dann auf der Produktionsdatenbank erstellt worden, wenn die vorherigen Tests erfolgreich gewesen sind. Die Knoten und Kanten auf der Produktionsdatenbank sind mengenmässig plausibilisiert und drei Einzelfälle genau untersucht worden. Die Ergebnisse sind stichpunktartig und mit Screenshots unter

```
src/main/documentation
```

dokumentiert worden. Das Test-Protokoll ist als Markdown erstellt worden und im Repository vorhanden.

4.4.2 User-Test

Für den User-Test ist eine vorbereitete Perspektive in den Neo4j Aura Account der Auftraggeberin importiert worden. Die Szenarien (siehe Abschnitt 3.5.2) sind durchgeführt und dokumentiert worden.

Als Ergebnis des User-Tests sind folgende konkrete Verbesserungsideen identifiziert worden:
- Die Knoten «Hierarchie» auf Level 7, «Substanz» und «Produkt» waren in der ersten Version des Modells so verbunden, dass es zu Verwirrung kommen konnte. «Hierarchie» mit Level 7 sollte entweder farblich hervorgehoben oder als eigenes Label kreiert werden.
- Die Saved Phrase Abfrage der WHO-Hierarchie sollte das Level 7 einschliessen.
- Index auf Produkt-Namen wäre hilfreich für die Freitext-Suche.
- In der Relation WAR_ATC_BIS sollte das Property «Jahr» in «Gültig bis und mit Jahr» umbenannt werden.
- Zunächst sollte nur mit «kleinen» und wenigen Saved Cypher Phrases gearbeitet werden, um die Übersichtlichkeit zu erhöhen.

Als allgemeines Zwischenfazit ist Folgendes festgehalten worden:
- Arbeiten mit Bloom/Explore ist nicht komplett selbsterklärend, d.h. man muss sich damit auseinandersetzen und eine «direkte» einfache Übergabe ist nicht möglich.
- Zunächst sollte man weniger mit Saved Phrases, sondern eher manuell mit den Knoten und Relationen arbeiten. Dadurch kann das Gesamtverständnis für das Datenmodells und deren Zusammenhänge erlernt werden. Pfade von Knoten zu Knoten sollte man auch «selbst finden», um Vertrauen im Umgang mit den Tools zu erhalten. Im weiteren Verlauf können die Safed Queries ausgebaut werden.
- Ein Graph-Modell kann durchaus im Alltag unterstützen, insbesondere wenn man mit vielen Informationspunkten/Medikamenten arbeitet.
- Es gibt gute, kleine Features (z.B. URL zur EMA/Compendium), die den Alltag vereinfachen.

Als Konsequenz dieser Ergebnisse ist die initiale Version des Datenmodells angepasst worden. Ursprünglich hatten die hochteuren Medikamente/Substanzen mit siebenstelligem ATC-Code das Label «Substanz», aber die siebenstelligen ATC-Codes, die sich nicht auf der «Liste der hochteuren Medikamente/Substanzen» befinden, besassen das Label «Hierarchie» mit dem Property «Level = 7». Dieses Nebeneinander von identischen Substanzen ist im Rahmen eines Refactorings behoben worden. In der verbesserten Version ist die «Hierarchie» mit Level = 7 direkt als Knoten «Substanz» erstellt worden. Bei ATC-Codes aus der «Liste der hochteuren Medikamente/Substanzen» sind die Properties in ein und denselben Knoten gemerged worden.

4.5 Schulung

4.5.1 Verfügbare Video-Tutorials

Neo4j ist ein System mit vielen Online-Tutorials. Die Anwenderin konnte mit Hilfe von zwei knapp siebenminütigen Videos, die auf YouTube verfügbar sind, die Grundlagen erlernen.

- Near Natural Language Search in Bloom (Video #2 in Bloom Series), Dauer: 6:42 Minuten
  https://youtu.be/9rL8O0lsuDc?si=nBTUzCShKzVu3qZY
- Exploration from Bloom Canvas (Video #3 in Bloom Series), Dauer: 6:54 Minuten
  https://youtu.be/7LHXYKAu9Nk?si=F--zt1UG8sa0_bbP

Darüber hinaus demonstriert ein weiteres Video die Informationen mit leicht anderen Schwerpunkten, aber auf sehr praktische Art und Weise. Dieses dritte Video konnte optional angeschaut werden.

- Exploring the WorldCup Graph with Neo4j Bloom, Dauer: 10:59 Minuten
  https://youtu.be/6RWK3e_-hbk?si=da9JshEHqaXSpZHW

Falls das erarbeitete Datenmodell mit Neo4j in der SwissDRG AG eingesetzt werden würde, sollten die Anwender ein ausführliches Video-Tutorial durcharbeiten. Dafür muss vorgängig eine Neo4j-Datenbank mit einem Beispieldatensatz (z.B. olympische Winterspiele) erstellt werden. Dieses Video enthält eine ausführliche Erläuterung über eine frühere Neo4j Bloom Version sowie praktische Übungsaufgaben.

- Training Series - Getting started with Neo4j Bloom, ab Zeitpunkt 32:45 bis Zeitpunkt 1:45:50
  https://www.youtube.com/live/7yS2e4p0_H4?si=AZQItNAICAe_-BPH

Zum Setup der Trainingsdatenbank mit Daten der olympischen Winterspiele muss ein Cypher-Script ausgeführt werden, das man unter folgendem Link finden kann:

- https://dev.neo4j.com/bloom-training-1

Da das Training mit einer frühere Neo4j Version durchgeführt worden ist, muss die Definition der Contraints angepasst werden. Der Befehl «ON» muss mit «FOR» und der Befehl «ASSERT» mit «REQUIRE» ersetzt werden.

```
Alt: CREATE CONSTRAINT IF NOT EXISTS ON (a:Athlete) ASSERT a.id IS UNIQUE;
Neu: CREATE CONSTRAINT IF NOT EXISTS FOR (a:Athlete) REQUIRE a.id IS UNIQUE;
```

Zudem hat sich die Syntax bei der Index-Erstellung leicht geändert.

```
Alt: CREATE INDEX /*IF NOT EXISTS*/ ON :Athlete(name);
Neu: CREATE INDEX /*IF NOT EXISTS*/ FOR (a:Athlete) ON a.name;
```

4.5.2 Benutzerhandbuch

Für die Auftraggeberin ist ein kompaktes Benutzerhandbuch mit Screenshots und kurzen Erläuterungen erstellt worden. Dieses wurde als Markdown-Datei kreiert, weil es damit problemlos in das markdownbasierte, interne Wiki bei der SwissDRG AG eingebaut werden konnte. Die Markdown-Datei findet man unter dem öffentlichen Link

- https://github.com/teletrabbie/medi_graph/blob/main/src/main/documentation/Benutzerhandbuch.MD

und im GitLab-Repository im Ordner `/src/main/documentation`. Das Benutzerhandbuch fasst einerseits das Datenmodell und das Schulungskonzept zusammen, andererseits werden konkrete Analysemöglichkeiten dokumentiert.

4.6 Auswahl möglicher Graph-Datenbanken für die SwissDRG AG

Die Bewertung der Graph-Datenbanken wurde mit folgenden Kriterien durchgeführt, wobei die technischen Anforderungen 1 und 2 als Einschlusskriterium galten.

- Anforderung 3: Image-Grösse gemäss Docker Hub unter 1 Gigabyte
- Kriterium 1: Schulungsprogramm vorhanden
- Kriterium 2: Strukturierte Dokumentation vorhanden
- Kriterium 3: Abfragesprache ist Cypher oder openCypher
- Kriterium 4: Visuelle Analysetools vorhanden
- Kriterium 5: Community-Edition existiert

Memgraph und Neo4j erfüllen alle Kriterien. Bei ArcadeDB, NebulaGraph, OrientDB und TypeDB ist jeweils nur ein Kriterium nicht erfüllt. Diese sechs Datenbanken stellen somit die «Short-List» dar. Bei anderer Kriteriengewichtung würde die Short-Liste möglicherweise anders ausfallen.

Tabelle 7: Vergleich der Graph-Datenbanken gemäss definierter Kriterien (Short-List)

Name der DB	Anf. 3	Krit. 1	Krit. 2	Krit. 3	Krit. 4	Krit. 5	Summe
Memgraph[1]	1	1	1	1	1	1	6
Neo4j	1	1	1	1	1	1	6
ArcadeDB[2]	1	0	1	1	1	1	5
NebulaGraph	0	1	1	1	1	1	5
OrientDB	1	1	1	0	1	1	5
TypeDB	1	1	1	0	1	1	5
Aerospike	1	1	1	0	0	1	4
ArangoDB	1	1	1	0	1	0	4
GraphDB	1	1	1	0	0	1	4
PuppyGraph	0	0	1	1	1	1	4
TigerGraph	0	0	1	1	1	1	4
AnzoGraph DB	0	0	1	1	0	1	3
Cayley	1	0	1	0	0	1	3
JanusGraph	1	0	1	0	0	1	3
RedisGraph	1	0	1	1	0	0	3
TerminusDB	1	0	1	0	0	1	3
Blazegraph[3]	1	0	0	0	0	1	2
Virtuoso	1	0	1	0	0	0	2

[1] Memgraph LDAP Integration nicht bei Community Edition.
[2] ArcadeDB kann andere Graph-DB importieren.
[3] Visualisierung mit AWS Graph-Notebook (github.com/aws/graph-notebook) in Jupyter Notebooks möglich.

5 Diskussion

5.1 Übersichtliche Informationen für die SwissDRG AG

Basierend auf den Anforderungen der Auftraggeberin konnten eine Reihe von Datenquellen in die Neo4j Datenbank integriert werden. So wurden Informationen über die hochteuren Medikamente/Substanzen, Zusatzentgelte, Produkte/Präparate, Indikationen, ATC/DDD-Index und Anträge zusammengeführt. Mehrere Relationen zwischen den Knoten verknüpfen die Informationen miteinander. Dadurch werden die Zusammenhänge graphisch dargestellt und sind einfach verständlich.

Zwei Beispiele können dies verdeutlichen. Erstens zeigt die Abbildung 25 sechs Produkte (grün) und deren Substanzen (gelb), die mit den Zusatzangaben (orange) verknüpft sind. Jedes Produkt ist eindeutig einem ATC-Code zugeordnet. Die ATC-Codes und die Zusatzangaben sind unterschiedlich oft verbunden.

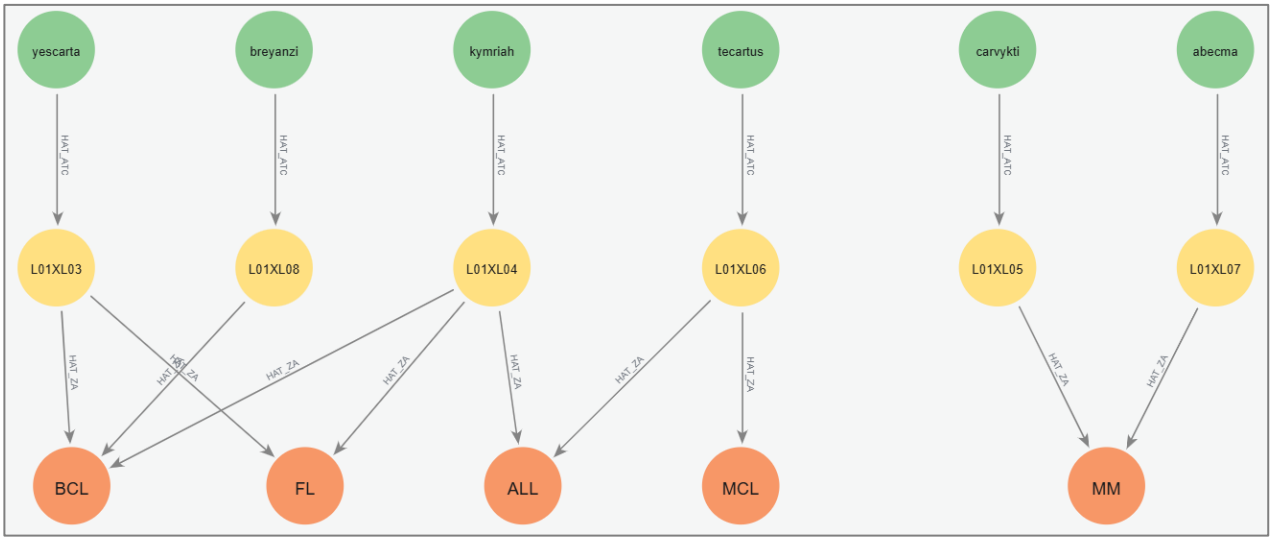


Abbildung 25: Produkte, Substanzen und Zusatzangaben im hierarchischen Layout

Die tabellarische Excel-Darstellung in der «Liste der hochteuren Medikamente/Substanzen 2025» sieht wie folgt aus:

Liste der hochteuren Medikamente / Substanzen
gültig ab dem 1. Januar 2025

ATC Code 2025	ATC Code 2024	Medikament / Substanz [1)]	zu codierende Zusatzangaben [2)]
L01XL03	L01XL03	Axicabtagen Ciloleucel	BCL, FL
L01XL04	L01XL04	Tisagenlecleucel	BCL, ALL, FL
L01XL05	L01XL05	Ciltacabtagene autoleucel	MM
L01XL06	L01XL06	Brexucabtagene autoleucel	ALL, MCL
L01XL07	L01XL07	Idecabtagene vicleucel	MM
L01XL08	L01XL08	Lisocabtagene Maraleucel	BCL

Abbildung 26: Substanzen und Zusatzangaben gemäss der Medi-Liste 2025

In der Graph-Darstellung ist leicht zu erkennen, dass die gelben Substanz-Knoten «L01XL05» und «L01XL07» (rechts) mit einem gemeinsamen Knoten «MM» verbunden sind. Diese Information ist in der Excel-Datei zwar auch enthalten, aber man muss etwas genauer hinschauen, um die Zusatzangabe «MM» in der dritten und fünften Zeile zu finden. Ein weiterer Zusatznutzen des Graph-Modells liegt in

den Produktbezeichnungen, die dort verknüpft sind. Um die Produkte zu finden, müsste die Auftraggeberin anderenfalls sechs einzelne Suchen im Internet bzw. im Compendium durchführen.

Zweitens können «Substanzen» (gelb) und deren «Zusatzentgelte» (grau) für «Produkte» (grün) mit bestimmten «Indikationen» (rosa) mit einer einfachen Abfrage gesucht werden. In der folgenden Darstellung wurde in der Graph-Datenbank nach Indikationen mit der Buchstabenfolge «ovar» gesucht. «Ovar» ist hier ein Teilwort von «Ovarialkarzinom». Dabei wurde nicht nur das Zusatzentgelt Nr. 11 für die Substanz «L01FG01», sondern auch das «ZE-95» für die Substanz «L01CX01» gefunden. Mit Hilfe von Tool-Tipps beim Mouse-Over oder der Property-Anzeige beim Doppelklick auf Knoten oder Relationen können schnell weitere Informationen angezeigt werden.

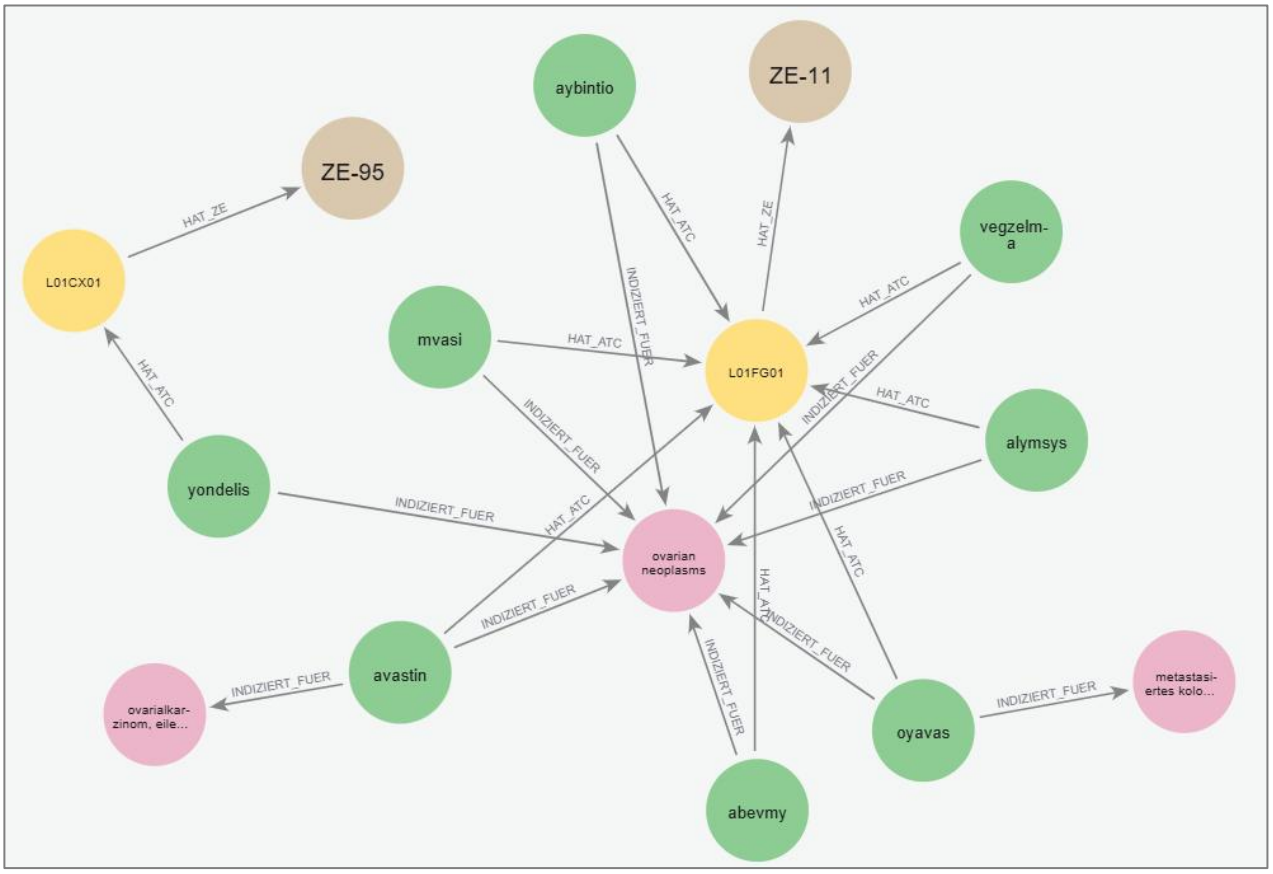


Abbildung 27: Produkte, Substanzen und Zusatzentgelte mit der Indikation "ovar"

Ohne die Graph-Datenbank müsste die Auftraggeberin nach Produkten zu dieser Indikation im Internet oder im Compendium suchen und zu jedem Treffer sowohl in der Medikamenten- als auch der Zusatzentgeltliste nachschlagen, ob die Substanzen dort vorhanden sind. Dies wäre zwar eine machbare Recherche, die aber Zeit in Anspruch nehmen würden. Die Graph-Datenbank kann nun dabei helfen, diese Recherchetätigkeit auf wenige Klicks zu reduzieren. Somit konnte die Hauptaufgabe dieses Projektes, die «Datensilos» zusammenzuführen, erfolgreich umgesetzt werden.

Der User-Test hatte zu einem Refactoring der initialen Version des Datenmodells geführt. Das aktuelle Model wurde für weitere Tests auf die Aura Cloud-Umgebung geladen. Für regelmässige Recherchetätigkeiten wurde das Modell bereits im Alltag der SwissDRG AG verwendet, die Daten plausibilisiert und der Aufbau des Modells kritisch hinterfragt. In diesem Testbetrieb zeigte sich, dass die Graph-Datenbank eine sinnvolle und praktische Ergänzung sein kann, da diverse benötigte Informationen an zentraler Stelle gespeichert sind und nicht an unterschiedlichen Orten gesucht werden müssen. Es hat jedoch keine Messung, Befragung oder Analyse gegeben, wie viel Recherchezeit effektiv durch die Nutzung der Graph-Datenbank eingespart werden kann. Subjektive Einzelaussagen der Auftraggeberin und eines IT-Mitarbeiters sind allerding positiv.

5.2 Besonderheiten des Datenmodells und Herausforderungen

Einer der Vorteile von Graph-Datenbanken im Vergleich zu relationalen Systemen kann die Abfragegeschwindigkeit sein. Eine Untersuchung der Performance hat jedoch nicht stattgefunden. Zudem muss man festhalten, dass die Zahl der Knoten und Relationen im erstellen Graph-Modell eher gering ist und die Vorteile der Abfrageperformance erst bei grösseren Datenmengen eine Rolle spielen. Die Abfragezeit ist somit kein Pro-Argument bei der Bewertung der Graph-Datenbank.

Die Modellierung von Relationen in diesem Projekt zeigt interessante Aspekte im Vergleich zu klassischen Joins im relationalen Modell. So wurden mehrere «Selbstrelationen» implementiert. Beim Label «Produkt» gibt es die Relation «IST_AUCH_PRODUKT» oder bei der «Substanz» die Relationen «WAR_ATC_BIS» oder «NICHT_AUF_MEDI_LISTE». Die Relation «IST_AUCH_PRODUKT» würde man in RDBMS als «Foreign-Key» zum «Primary-Key» in einer anderen Zeile der gleichen Tabelle modellieren. Im Graphen ist diese Beziehung praktisch genauso vorhanden und nicht besonders innovativ. Dennoch wird durch diese Konstruktion sichergestellt, dass alle Produkte auffindbar sind und zu einem Knoten zusammengefasst werden. Der Hintergrund ist, dass in Neo4j Knoten mit dem gleichen Namen bei MERGE-Statements aktualisiert resp. aggregiert werden. Dieser Eigenschaft muss man sich als Entwickler bewusst sein und das Datenmodell entsprechend gestalten.

Die Relationen der «Substanz» zu sich selbst sind Verbindungen von einer «Zeile» (im Sinnen des RDBMS) zur «gleichen Zeile». Typischerweise würde man in solchen Fällen im RDBMS einfach weitere Spalten zu einer Tabelle hinzufügen. Die Relation «NICHT_AUF_MEDI_LISTE» würde einem «Flag» mit einer Ausprägung von «TRUE» oder «FALSE» entsprechen. Dies ist ein gängiges Mittel relationaler Tabellen. Allerdings könnte dies zu unnötig vielen Daten führen, wenn man eigentlich nur wenige Datenpunkte mit «TRUE» kennzeichnen möchte und stattdessen für alle Zeilen einen Eintrag in der Datenbank anlegen muss. Das Argument von unnötig vielen «FALSE» Einträgen in einer Spalte gilt beim erarbeiteten Graph-Modell jedoch gerade nicht, weil die überwiegende Zahl der Substanzen die Relation «NICHT_AUF_MEDI_LISTE» aufweist. Das folgende Cypher-Query ergibt einen Anteil der Substanz-Knoten mit der Relation «NICHT_AUF_MEDI_LISTE» an allen Substanzen von 94.1%.

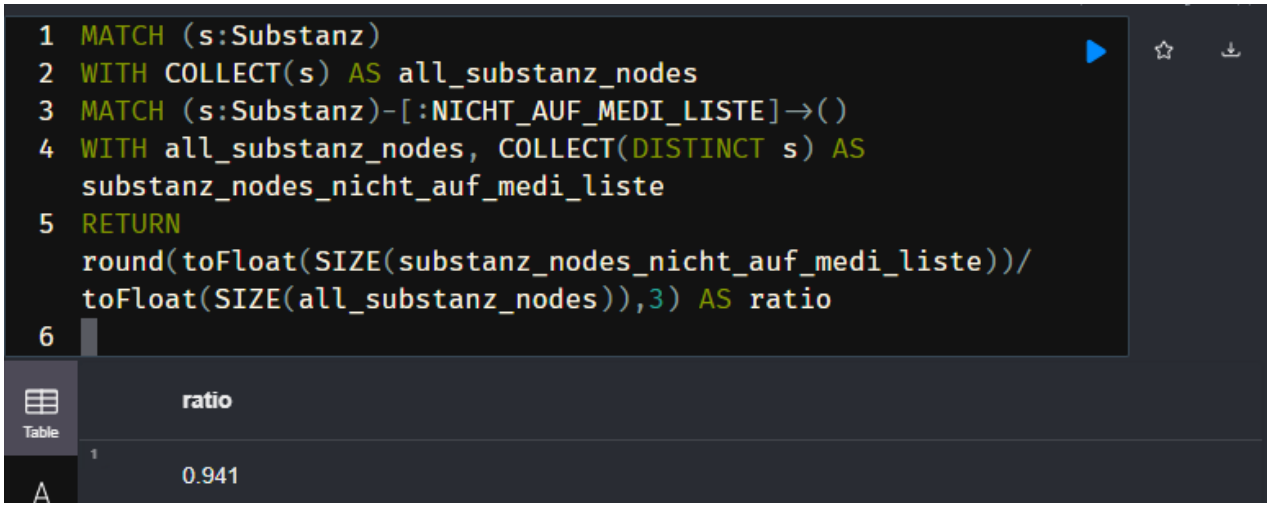


Abbildung 28: Code und Anteil der Substanzen mit und ohne Relation "NICHT_AUF_MEDI_LISTE"

Der Cypher-Code ist im Skript «20_analytics.cypher» vorhanden.

Hier hätte man die Logik der Relation umdrehen und als Relation «IST_AUF_MEDI_LISTE» modellieren können. Allerdings wurde diese Relation einfach als «pragmatische» Kennzeichnung im Zusammenhang mit der Usability von Neo4j Bloom modelliert. Damit werden Substanzen auf der Medi-Liste von Substanzen ohne Eintrag auf der Medi-Liste unterschieden. Man hätte auch ein zweites Label verwenden können. Labels können sich jedoch «überlagern», was in der Visualisierung von Netzen zu mehr Verwirrung führen kann als zur Aufklärung beiträgt. Für einen weiteren Weiterentwicklungsschritt des Datenmodells sollte im Rahmen eines weiteren User-Tests geprüft werden, ob das «Flag» als Relation oder als eigenes Label zielführender ist.

Es gab weitere Details bei der Implementierung des Property-Graph-Modells, die sich vom relationalen Modell klar unterscheiden und durchaus als «besser» oder zumindest flexibler angesehen werden können. So können Knoten des gleichen Labels grundsätzlich unterschiedliche Properties enthalten. Bei dem «Produkt» gibt es in einigen Knoten beispielweise das Property «Erstzulassung» und bei anderen nicht. Dafür gibt es bei anderen Knoten gewisse Properties wie z.B. «Biosimilar» und bei den erstgenannten wiederum nicht. In RDBMS hätten somit viele Spalten einen hohen Anteil an fehlenden Werten («NULL»). Zudem müssen die Datentypen in RDBMS pro Spalte einheitlich sein. Auch dies ist bei Property-Graphen nicht nötig, wo unterschiedliche Datentypen für das gleiche Property gemischt werden können. Beispielsweise kann das Property «DDD» im Label «Substanz» entweder vom Typ «StringArray» oder vom Typ «Double» sein.

Auch bei der Modellierung der Relationen sind Unterschiede zwischen GDBMS und RDBMS sichtbar. Im entwickelten Medikamenten-Graphen gibt es beispielsweise die Relation «HAT_ATC», die die Knoten «Produkt» und «Substanz» verbinden. Die Relation «HAT_ATC» wird gleich auf drei Arten erstellt. Entweder stimmen die ATC-Codes der EMA und der «Substanz» überein oder die ATC-Codes von Refdata und der «Substanz» sind identisch. Falls bis dahin keine Relation erstellt werden konnte, aber der ATC-Code der «Spezialitätenliste» gleich dem ATC-Code der «Substanz» ist, wird die Relation von «Produkt» und «Substanz» als dritte Option (indirekt über die Zulassungsnummer) hergestellt. Ausserdem wird die Relation «HATTE_ATC» mit dem Property «Früherer ATC» der Relation «WAR_ATC_BIS» erstellt und ergänzt die Verknüpfungen. Dadurch kann beispielweise einer mangelnden Datenqualität einer Datenquelle entgegengewirkt werden, weil Informationen aus diversen Datenquellen in die Relation einfliessen. In Graph-Modellen «ergänzen» sich die Informationen somit im weitesten Sinn. In relationalen Systemen müsste man die Joins zwischen den Tabellen hingegen mit komplizierten COALESCE-Statements oder ähnlichen Konstruktionen «zusammenbasteln», was zu Abfrageproblemen führen kann und die Lesbarkeit der Queries deutlich reduziert.

Die Relationen «HAT_ATC» hat eine weitere Besonderheit, die sich aus den Datenquellen ergibt. So kann ein und dasselbe «Produkt» in wenigen Fällen direkt mit unterschiedlichen «Substanz» Knoten verbunden sein.

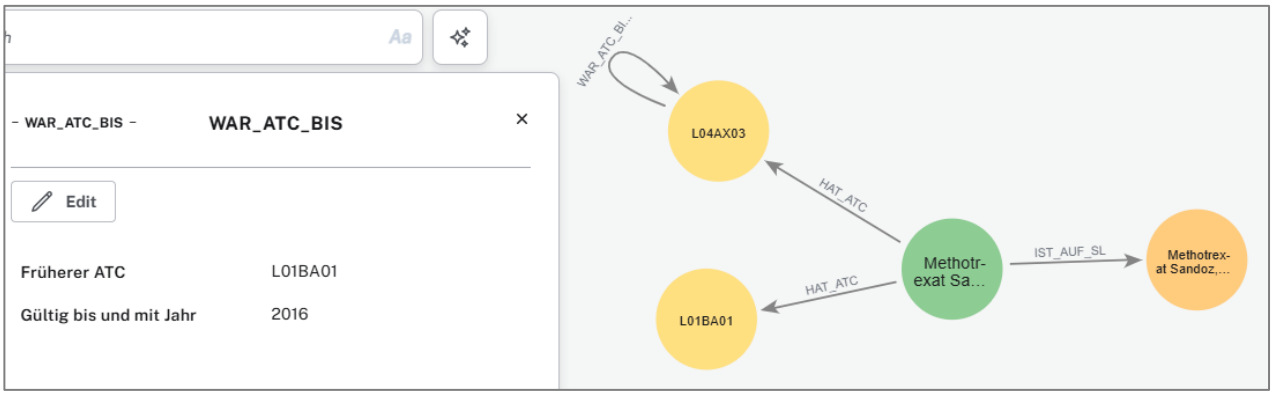


Abbildung 29: Produkt mit zwei "Substanz" Knoten

Beispielsweise hat das Produkt mit dem GTIN «7680583700066» (METHOTREXAT Sandoz Tabl 10 mg, 10 Stk) zwei Relationen zu den Substanz-Knoten «L01BA01» und «L04AX03». Die Relation zum erstgenannten Knoten stammt von veralteten Daten aus den strukturierten Arzneimittelinformationen der Stiftung Refdata (siehe Screenshot im Anhang, Kapitel 11.1). Die zweite Relation wird mit dem aktuellen ATC-Code der Spezialitätenliste hergestellt (siehe Screenshot im Anhang, Kapitel 11.1). In einem RDBMS müsste man für solche Konstellationen eine Zwischentabelle mit 1:m Beziehungen anlegen, wobei diese Zwischentabelle für die absolute Mehrheit der Zeilen eine Eins-zu-Eins Relation wäre und nur wenige 1:m Einträge hätte. Alternative Lösungen (z.B. Datenbereinigung) wären denkbar, aber an diesem Beispiel zeigt sich wieder die Flexibilität einer Graph-Datenbank. Im Medikamenten-Graph lassen sich somit Verknüpfungen zwischen den Informationspunkten herstellen, auch wenn veraltete Daten enthalten sind. In einem anderen Kontext, z.B. in klinischen Informationssystemen oder Apothekeninformationssystemen, wäre es dagegen äusserst wichtig, dass allein nur der korrekte ATC-

Code ausgewiesen wird. Für diese anderen Anwendungen müsste eher ein strenges relationales Modell genutzt und/oder die Datenbereinigung systematischer durchgeführt werden.

Dies bedeutet aber nicht, dass «viele falsche» Daten im Medikamenten-Graph vorhanden sind oder dass kein Wert auf Datenqualität in Graph-Modellen gelegt werden sollte. Im Rahmen des Preprocessing der SAI-Daten wurden veraltete ATC-Codes für 22 Zulassungsnummern aktualisiert. Die Aktualisierung erfolgte zwar im R-Skript «05_refdata.R», aber das Update hätte alternativ auch im Cypher-Skript «05_sai_praeparate.cypher» erfolgen können (nach Erzeugung der Produkt-Knoten, aber vor Erstellung der Relation «HAT_ATC» zur Substanz). Beispielweise ist der R-Code

```
df_praeparate[df_praeparate$ZULASSUNGSNUMMER=='67575',"ATC_CODE"]<-'L01XL07'
```

gleichbedeutend mit dem Cypher-Code

```
MATCH (p:Produkt {Zulassungsnummer: '67575'}) SET p.`ATC-SAI`='L01XL07'
```

Die Code-Fragmente wären austauschbar und man kann sich durchaus die Frage stellen, in welchen Skript die Datenkorrektur am sinnvollsten aufgehoben ist. Teilweise kommt es auf die Datenquelle an. Falls Daten direkt von der Quelle (als csv-Datei oder relationale DB-Tabelle) in Neo4j eingelesen werden, so kann die Datenkorrektur durchaus direkt im Cypher-Skript erfolgen. Wenn die Datenquelle aber sowieso zuvor aufbereitet werden muss (im Rahmen eines Preprocessings), dann sollte die Korrektur besser im Preprocessing-Skript durchgeführt werden. Grund ist vor allem, dass es das «Ziel» des Preprocessing ist, die Datenquellen möglichst gut für den Import in die Graph-Datenbank vorzubereiten. Hierzu gehören möglichst fehlerfreie Daten. Daten mit geringer Qualität sollten, wenn immer möglich, gar nicht erst in die Graph-Datenbank einfliessen.

Eine weitere grundsätzliche Frage stellt sich beim Datenimport: wieso ist überhaupt ein Preprocessing nötig? Beispielsweise werden die SAI-Daten mit Hilfe von R aufwändig von einer XML-Struktur in ein Data-Frame umgewandelt. Das Data-Frame wird als csv-Datei gespeichert und in Neo4j eingelesen. Die Einzelschritte könnte man alternativ in einem Gesamtskript zusammenfassen. Dafür könnte man die Java- oder Python-Konnektoren von Neo4j nutzen und beim Auslesen der XML-Daten direkt einen einzelnen Knoten in der Graph-Datenbank erstellen. Es gibt sicherlich Situationen, in denen eine solche serielle Datenverarbeitung Sinn machen würde. Aber für den Medikamenten-Graph sind bisher keine besonderen Gründe aufgetaucht, wieso man von der Batch-Verarbeitung abweichen müsste. Zudem kennt man das Design-Pattern «Loose coupling» aus dem Software-Engineering. Danach sollten einzelne Komponenten möglichst unabhängig voneinander sein, um sie bei Bedarf flexibel austauschen zu können. Das R-Skript repräsentiert eine und das Cypher-Skript eine andere Komponente. Änderungen der Datenquelle (z.B. der XML-Struktur) würden somit nur im R-Skript zu Anpassungen führen, aber das Cypher-Skript bliebe unverändert. Oder das Preprocessing bleibt unverändert mit qualitativ hochwertigen Daten, wenn man einen andere Graph-DB nutzen möchte. Dann muss man sich beim Aufbau eines neuen Graph-Modells nicht mehr um die Datenvorbereitung kümmern.

Bei der Modellierung von relationalen Datenbanken führt die Erzeugung der Normalformen, Schemata, Constraints und Zwischentabellen für n:m Beziehungen zu umfangreichem Entwicklungsaufwand und Besprechungen zwischen IT und Fachabteilungen. Im Gegensatz dazu können die Graph-Modelle flexibel und intuitiv skizziert werden, im dem man mit Stift und Papier oder mit Tools wie arrows.app die Knoten und Relationen zeichnet. Fachspezialisten, die keine Datenbankentwickler sind, können somit einen Grossteil des Datenmodells skizzieren und deren Expertise für unternehmensrelevante Informationsbedürfnisse direkt einfliessen lassen. Der Entwicklungsprozess für ein Datenmodell könnte somit grundsätzlich verkürzt werden. Dennoch führt auch ein Graph-Modell zu Entwicklungsaufwand, denn Datenquellen müssen analysiert, angebunden, teilweise aufbereitet und getestet werden. Die Implementierung ist somit nicht «ETL frei», sondern gewisse Schritte von «extract, transform und load» müssen auch bei Graph-Datenbanken erledigt werden. Zudem sind es viele Anwender gewohnt, dass sie Listen oder Tabellen analysieren, statt Graphen. Bei der Planung eines Graph-Datenbank-Projektes sollten somit umfangreiche Bedürfnisse aller Beteiligten berücksichtigt werden, um eine realistische Aufwandsschätzung für die Entwicklung, den Betrieb und Change-Prozesse zu erstellen.

In dieser Bachelorarbeit wurden die Cypher-Skripte mehrheitlich manuell oder punktuell mit Unterstützung von generativer KI entwickelt. Die technische Umsetzung eines Graph-Modells könnte mit Hilfe spezieller Tools noch beschleunigt werden. So bietet Neo4j in der Aura Umgebung ein Import-Tool, mit dem csv-Dateien oder relationale Datenbanken direkt in ein Graph-Schema überführt werden können. Wie bei arrows.app können Knoten und Kanten graphisch skizziert und die Importdaten zugeordnet werden. Für ein skizziertes Datenmodell wird dort automatisch Cypher-Code generiert, den man nutzen oder weiterentwickeln kann. Somit besteht Vereinfachungspotential für zukünftige Projekte oder für die Weiterentwicklung des bestehenden Datenmodells.

5.3 Ausblick

Die Auftraggeberin und zwei andere Mitarbeiter der SwissDRG AG nutzen den Medikamenten-Graph zunächst in der Neo4j Aura-Cloud-Umgebung. Im Rahmen des Projektabschlusses wurde zudem das Benutzerhandbuch an die Auftraggeberin übergeben, indem es in das interne Wiki integriert wurde. Alle relevanten Dateien wurden vom GitLab der BFH in das eigene GitLab der SwissDRG AG übertragen. Somit kann das Datenmodell innerhalb der SwissDRG AG weiterentwickelt werden.

Neben Neo4j mit der Abfragesprache Cypher existieren weitere Graph-Datenbanken und Sprachen, mit denen man ein Graph-Modell aufbauen kann. Eine abschliessende Bewertung von Neo4j gegenüber Konkurrenzprodukten oder von Cypher gegenüber anderen Sprachen konnte im Rahmen der Bachelorarbeit nicht geleistet werden. Dennoch wurden z.B. mit Memgraph und ArcadeDB Datenbanken mit sehr ähnlichem Leistungsumfang gefunden, mit denen man das Datenmodell nutzen könnte. Beide Datenbanken verwenden ebenfalls Cypher bzw. einen Cypher-Dialekt. Für die weitere Nutzung des Datenmodells ist die SwissDRG AG jedoch nicht an Cypher gebunden. In den heutigen Zeiten mit generativer KI könnte man sich den erstellten Cypher-Code in eine andere Abfragesprache z.B. Gremlin übersetzen lassen. Dennoch ist der Fokus auf Cypher durchaus sinnvoll, weil die Abfragesprache GQL als neuer ISO-Standard auf openCypher aufbaut und der zukünftige Standard in GDBMS werden könnte (ähnlich wie SQL nicht nur ISO- sondern faktischer Standard bei RDBMS ist).

Falls die SwissDRG AG nicht Neo4j, sondern eine andere Datenbank, z.B. Memgraph, für den weiteren Betrieb nutzen würde, müsste ein geringer Entwicklungsaufwand für die Umstellung eingeplant werden. Ein Praxistest in der Memgraph-Cloud-Umgebung hat gezeigt, dass der für Neo4j entwickelte Cypher-Code nicht sofort durchläuft. Erstens gibt es im Cypher-Editor von Memgraph eine Beschränkung auf 5'000 Zeichen, sodass die Einzelskripte statt des Gesamtskripts ausgeführt werden müssten. Zweitens gibt es geringe Unterschiede bei Schlüsselbegriffen z.B. für den Datenimport aus einer csv-Datei. So wird die Anweisung «FIELDTERMINATOR» in Memgraph nicht akzeptiert, sondern muss mit «DELIMITER» ausgetauscht und vor «AS row» geschrieben werden. Ausserdem lautet die Anweisung «HEADERS» in Memgraph «HEADER» ohne «S». Der Beispielcode für Neo4j

```
LOAD CSV WITH HEADERS FROM '/file.csv' AS row FIELDTERMINATOR '|'
```

kann für Memgraph geändert werden zu

```
LOAD CSV FROM '/file.csv' WITH HEADER DELIMITER '|' AS row
```

Der Wechsel von Neo4j und Cypher hin zu einer anderen Datenbank oder Abfragesprache ist zweifelslos möglich. Dennoch sollte für einen Technologiewechsel etwas Zeit für die «Migration», Testing und Schulung eingeplant werden.

Beim Abschluss des Projektes mit der SwissDRG AG bestand kein spezieller Grund für einen Technologiewechsel. Die Auftraggeberin ist über den Neo4j Aura Account an die Aura-Umgebung des Datenmodells angebunden und hatte durch die User-Schulung im Vorfeld des User-Tests die Neo4j Tools kennengelernt. Die kostenlose Neo4j Cloud-Umgebung ermöglicht es nun, das Datenmodell ohne Risiko im Alltagseinsatz zu testen. Mit dem hohen Leistungsumfang, Marktanteil und Bekanntheitsgrad von Neo4j sowie einer grossen Community kann die SwissDRG AG zunächst auf eine risikoarme Standardlösung für den Betrieb des Medikamenten-Graphen setzen. Sobald eine Graph-Datenbank fest

in den Technologie-Stack der SwissDRG AG aufgenommen werden soll, muss eine ausführlichere Evaluation der GDBMS basierend auf der Vorauswahl gemäss Tabelle 7 (siehe Abschnitt 4.6) erfolgen.

Zum Abschluss der Bachelor-Thesis gibt es Anforderungen, die noch nicht bearbeitet wurden. So fehlen die Daten des gemeinsamen Bundesauschuss aus Deutschland vollständig. Es blieb am Ende schlicht keine Zeit mehr, die Daten aufzubereiten und in die Datenbank hinzuzufügen. Zudem ist im Laufe des Projektes die Idee entstanden, dass die Zusatzentgelte des deutschen DRG-Systems ebenfalls in das Graph-Modell eingefügt werden könnten. Diese zusätzliche Anforderung konnte ebenfalls noch nicht implementiert werden. Darüber hinaus wurde zwar der Orpha-Status gemäss EMA in die Datenbank eingefügt, aber der Orpha-Status von SwissMedic oder von OrphaNet konnte noch nicht ergänzt werden. Zudem konnten weder Ontologien oder Terminologien eingebunden werden, die die Indikationen besser gruppieren oder anderweitig präzisieren könnten. Das Datenmodell wurde nicht so formell und explizit formuliert, wie es sonst bei Ontologien der Fall ist. Der Knoten «Produkt» sollte in «Präparat» umbenannt werden, weil dies im medizinischen Sprachgebrauch üblicher ist. Insgesamt könnte man das Modell weiter verfeinern. Diese offenen Themengebiete und andere Verbesserungsideen können zu einem späteren Zeitpunkt ergänzt werden, um die Informationen für den Betrieb der Datenbank innerhalb der SwissDRG AG zu komplettieren.

Zukünftig liesse sich die Graph-Datenbank mit internen Daten der SwissDRG AG erweitern, um weitere Fragestellungen zu analysieren. Zu Testzwecken wurden z.B. stationäre Fälle in die lokale Neo4j-Umgebung eingefügt, um Kombinationen von hochteuren Medikamenten zu identifizieren. Die Fälle wurden als eigenes Label «Fall» und die Verabreichung hochteurer Medikamente als Relation vom «Fall» zur «Substanz» modelliert. Technisch wurde aus einer Tabelle im RDBMS eine Relation im GDBMS.

Die Abbildung 30 zeigt ein Netzwerk von stationären Fällen des Jahres 2024 für ein einzelnes Spital. Die stationären Fälle werden grau und Substanzen gelb dargestellt. Die Fälle sind in Clustern angeordnet, die teilweise nur über einen Fall in Beziehung stehen oder komplett unabhängig (disjunkt) sind. Dadurch lassen sich hochteure Substanzen identifizieren, die häufig in Kombination miteinander angewendet werden. Mit den graphischen Clustern lassen sich Wirkstoffkombinationen leicht identifizieren, sodass man bei der Weiterentwicklung des Tarifsystems zukünftig mehr Rücksicht auf Kombinationen und Behandlungskomplexe nehmen könnte.

Den Ansatz mit Fällen und deren Relationen könnte man für spezifische Analysen ausweiten. So wären die Kombinationen von Fällen mit ICD-10-Codes oder CHOP-Codes grundsätzlich auf gleiche Art und Weise möglich. Dabei sollte jedoch eine sinnvolle Vorselektion stattfinden, denn pro Jahr gibt es rund 1.2 Mio. stationäre Fälle in der Schweiz und der Upload aller Fälle mit allen Diagnosen und Prozeduren ist vermutlich nicht besonders zielführend. Mit Hilfe von parametrisierbaren Deep-Links könnten Anwender ausserdem von anderen Systemen direkt in eine Perspektive in Neo4j Explore/Bloom abspringen und sich spezifische Daten anzeigen lassen.

Neben den Medikamenten und Fällen könnte die SwissDRG AG eine intern implementierte Graph-Datenbank noch für ganz andere Themengebiete einsetzen. So könnte das IT-Netzwerk damit analysiert und überwacht werden oder ein KI-Pilotprojekt mit dem GraphRAG-Ansatz zu besseren Ergebnissen gelangen.

Zusammenfassend kann festgehalten werden:
- Das Graph-Modell mit umfangreichen Medikamentendaten wird in der SwissDRG AG eingesetzt,
- Die Datenbank hilft bei Recherchearbeiten über Medikamente und unterstützt die Weiterentwicklung der stationären Tarifsysteme,
- Die Neo4j Cloud-Umgebung wird zunächst für den Betrieb des Graph-Modells verwendet,
- Die Implementierung einer internen Graph-Datenbank, mit Neo4j oder einer anderen Datenbank, ist vorbereitet und möglich. Eine finale Entscheidung über die eingesetzte Technologie wird aber zu einem späteren Zeitpunkt fallen.

Ich selbst konnte durch das Bachelorprojekt unglaublich viel lernen und hoffe, mich auch weiterhin mit Datenbanken im Allgemeinen und Graph-Datenbanken im Speziellen beschäftigen zu können.

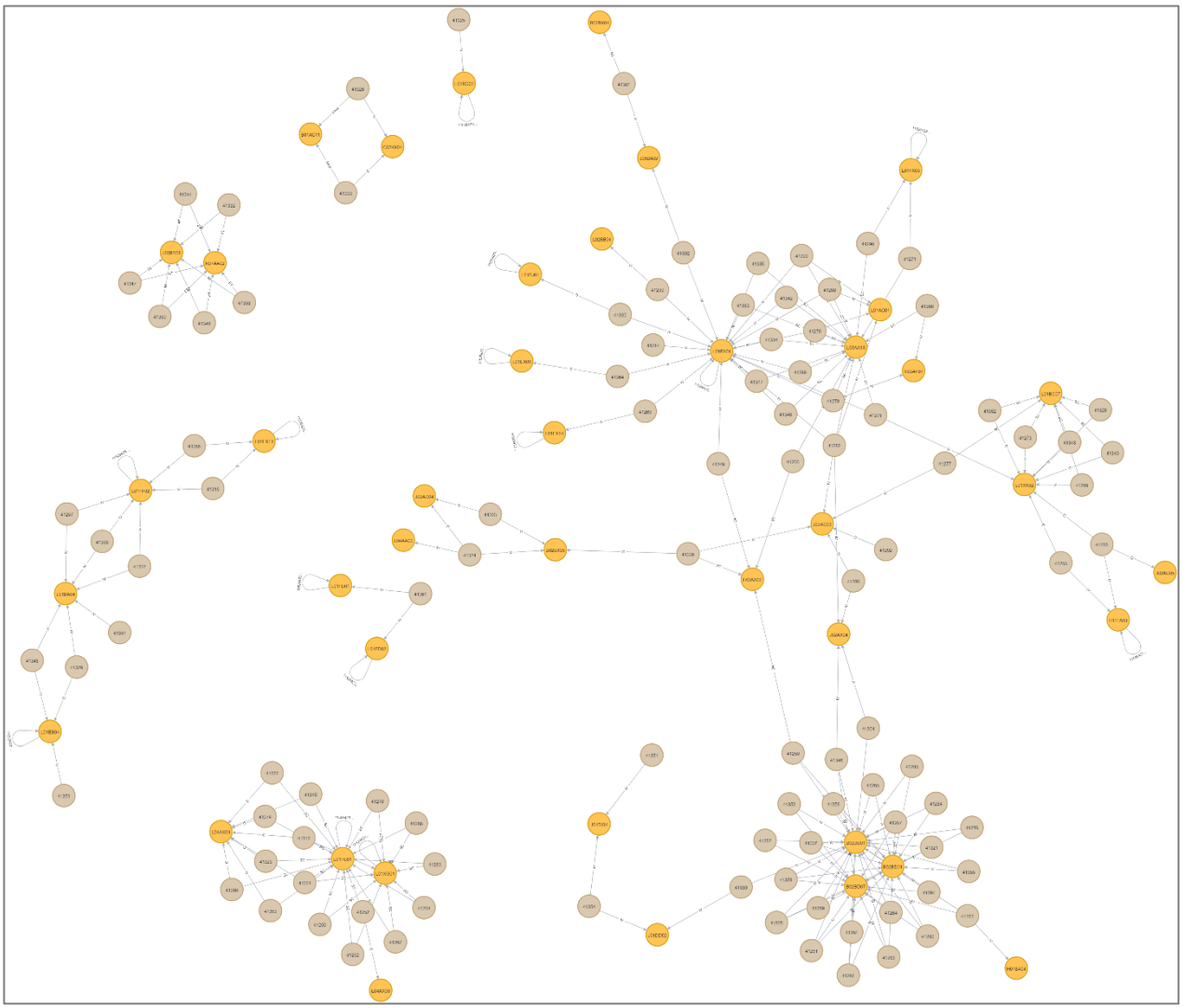


Abbildung 30: Analyse von Fällen und Substanzkombinationen

6 Abbildungsverzeichnis

7 Tabellenverzeichnis

8 Glossar

Bezeichnung	Erläuterung
BFH	Berner Fachhochschule
EMA	European Medicines Agency (Europäische Arzneimittelagentur) ist das EU-Pendant zu SwissMedic (oder der FDA in den USA)
FDA	Food and Drugs Administration ist das US-amerikanische Pendant zu SwissMedic (oder der EMA in der EU)
GDBMS	Graph-Datenbank-Managementsystemen, deren Daten in Knoten und Kanten modelliert werden
GQL	Graph Query Language ist eine Abfragesprache für GDBMS, die seit 2024 als ISO-Standard etabliert wurde. Sie ist, nach SQL, die zweite Abfragesprache, für die ein ISO-Standard erstellt wurde.
INSERM	Institut national de la santé et de la recherche médicale (Französisches Nationalinstitut für Gesundheit und medizinische Forschung)
Kanten	Werden auch Relationen genannt und verbinden die Knoten eines Graphen miteinander. Im LPG können Kanten eigenen Attribute/Properties haben.
Knoten	Wird auf engl. «node» genannt und repräsentiert einen Datenpunkt in einem Graphen. Im Zusammenhang mit LPG wird dies teilweise synonym zu Label verwendet.
LPG	Labeled Property Graph sind eine von zwei Graph-Typen, bei denen die Knoten mit Knotentypen (Labels) versehen sind und Attribute/Properties haben (das sind weitere Datenpunkte).
Many-to-Many-Relations	Wird auch n:m Relation genannt und stellt eine nicht eindeutige Verbindung von Daten dar, die in RDMS über Zwischentabellen modelliert werden
Neo4j	Markenbezeichnung einer nativen Graph-Datenbank, die von der Firma Neo4j Inc. hergestellt und weiterentwickelt wird
RDBMS	Relationale Datenbank-Managementsysteme, die Daten in Tabellen halten und in denen in der Regel mit SQL gearbeitet wird
RDF	Resource Description Framework ist ein W3C Standard für den Datenaustausch und Wissensrepräsentation im semantischen Web
SAI	Strukturierte Arzneimittelinformationen der Stiftung Refdata aufgeteilt in mehrere XML-Dateien sowie XSD-Files.
SL	Die Spezialitätenliste des Bundesamt für Gesundheit ist eine Liste mit Medikamenten und deren Preise, die von der obligatorischen Krankenpflegeversicherung übernommen werden.
SPARQL	Protocol and RDF Query Language ist eine Abfragesprache, um RDF Graphen zu traversieren, wobei gleichzeitig der Protokoll-Layer (http) und das Outputformat spezifiziert wird
SQL	Structured Query Language ist eine Abfragesprache für RDBMS, die sehr verbreitet und etabliert ist und in der Norm ISO/IEC 9075 spezifiziert wurde
Traversal	Wird auch Traversierung genannt und bezeichnet das Durchqueren eines Graphen oder Netzwerkes auf einem spezifischen Pfad
URI	Unified Resource Identificator dient der Identifizierung von abstrakten oder realen Ressourcen
VM	Virtuelle Maschine ist ein zusätzliches Rechnersystem innerhalb eines bereits lauffähigen Rechnersystems
ZE	Zusatzentgelt; wird zusätzlich zur stationären Fallpauschale gezahlt

9 Literaturverzeichnis

1. SwissDRG AG. Aufgaben und Verantwortung der SwissDRG AG [Online]. 2023 [aufgerufen am 06.03.2025]. Verfügbar unter: https://www.swissdrg.org/download_file/view/4598/2075.
2. SwissDRG AG. Aufgaben und Verantwortung [Online]. [aufgerufen am 06.03.2025]. Verfügbar unter: https://www.swissdrg.org/de/ueber-uns/organisation/aufgaben-und-verantwortung.
3. SwissDRG AG. Fallpauschalen-Katalog SwissDRG-Version 14.0 Abrechnungsversion (2025/2025) [Online]. [aufgerufen am 06.03.2025]. Verfügbar unter: https://www.swissdrg.org/de/akutsomatik/swissdrg-system-1402025/fallpauschalenkatalog.
4. SwissDRG AG. Klarstellung zur Abrechnung der Zusatzentgelte nach SwissDRG Abrechnungsversion 14.0 sowie zur Kodierung der Medikamente/Substanzen [Online]. 2024 [aufgerufen am 06.03.2025]. Verfügbar unter: https://www.swissdrg.org/download_file/view/5012/2253.
5. SwissDRG AG. Datenerhebung Archiv [Online]. [aufgerufen am 06.03.2025]. Verfügbar unter: https://www.swissdrg.org/de/akutsomatik/datenerhebung/archiv.
6. Erhebung 2026 (Daten 2025) [Online]. [aufgerufen am 06.03.2025]. Verfügbar unter: https://www.swissdrg.org/de/akutsomatik/datenerhebung/erhebung-2026-daten-2025.
7. SwissDRG AG. SwissDRG System 7.0/2018 Zusatzentgeltkatalog [Online]. [aufgerufen am 06.03.2025]. Verfügbar unter: https://www.swissdrg.org/de/akutsomatik/archiv-swissdrg-system/swissdrg-system-702018.
8. Norwegian Institute of Public Health, WHO Collaborating Centre for Drug Statistics Methodology. ATC/DDD Index [Online]. [aufgerufen am 06.03.2025]. Verfügbar unter: https://atcddd.fhi.no/atc_ddd_index/.
9. SwissDRG AG. Dokumentation EDV-Version 2025 [Online]. 2024 [aufgerufen am 06.03.2025]. Verfügbar unter: https://www.swissdrg.org/download_file/view/4994/2253.
10. SwissDRG AG. Antragsverfahren und Kriterien zur Antragsbearbeitung [Online]. [aufgerufen am 06.03.2025]. Verfügbar unter: https://www.swissdrg.org/de/akutsomatik/antragsverfahren.
11. Gillis A. What is a graph database? [Online]. Definition from WhatIs. [aufgerufen am 26.02.2025]. Verfügbar unter: https://www.techtarget.com/whatis/definition/graph-database.
12. Schäfer J, Tang M, Luu D, Bergmann AK, Wiese L. Graph4Med: a web application and a graph database for visualizing and analyzing medical databases. BMC Bioinformatics. 2022;23(1):537.
13. Schrodt J, Dudchenko A, Knaup-Gregori P, Ganzinger M. Graph-Representation of Patient Data: a Systematic Literature Review. J Med Syst. 2020;44(4):86.
14. Walke D, Micheel D, Schallert K, Muth T, Broneske D, Saake G, et al. The importance of graph databases and graph learning for clinical applications. Database J Biol Databases Curation [Online]. 2023;2023. Verfügbar unter: https://academic.oup.com/database/article/doi/10.1093/database/baad045/7222237.
15. Stothers JAM, Nguyen A. Can Neo4j Replace PostgreSQL in Healthcare? AMIA Summits Transl Sci Proc. 2020;2020:646–53.
16. Timón-Reina S, Rincón M, Martínez-Tomás R. An overview of graph databases and their applications in the biomedical domain. Database J Biol Databases Curation. 2021;2021:baab026.
17. BeyondVerse. Graph Algorithms: Traversals, Shortest Paths, and Beyond [Online]. Medium. 2023 [aufgerufen am 08.03.2025]. Verfügbar unter: https://medium.com/@beyond_verse/graph-algorithms-traversals-shortest-paths-and-beyond-671f611aa025.
18. Yoon BH, Kim SK, Kim SY. Use of Graph Database for the Integration of Heterogeneous Biological Data. Genomics Inform. 2017;15(1):19–27.
19. Neo4j Inc., Webber J. RDF vs. Property Graphs: Choosing the Right Approach for Implementing a Knowledge Graph [Online]. Graph Database & Analytics. 2024 [aufgerufen am 26.02.2025]. Verfügbar unter: https://neo4j.com/blog/rdf-vs-property-graphs-knowledge-graphs/.
20. How to Represent Simple Facts with RDF [Online]. OpenHPI; 2019 [aufgerufen am 08.03.2025]. (Linked Data Engineering (Semantic Web)). Verfügbar unter: https://www.youtube.com/watch?v=qgdzHQQj_vU.

21. How to Query RDFS SPARQL [Online]. OpenHPI; 2019 [aufgerufen am 08.03.2025]. (Linked Data Engineering (Semantic Web)). Verfügbar unter: https://www.youtube.com/watch?v=RoogS47Fp8o.
22. Barrasa J. How semantic is your graph? In San Fransisco; Verfügbar unter: https://www.youtube.com/watch?v=OVweE--RJqM.
23. DB-Engines Ranking [Online]. DB-Engines. [aufgerufen am 08.03.2025]. Verfügbar unter: https://db-engines.com/en/ranking/graph+dbms.
24. Top 10 Open Source Graph Databases in 2025 [Online]. GeeksforGeeks. 2025 [aufgerufen am 08.03.2025]. Verfügbar unter: https://www.geeksforgeeks.org/open-source-graph-databases/.
25. Neo4j Inc. Neo4j Deployment Center [Online]. Graph Database & Analytics. [aufgerufen am 08.03.2025]. Verfügbar unter: https://neo4j.com/deployment-center/.
26. Neo4j Inc. Neo4j Pricing [Online]. Graph Database & Analytics. [aufgerufen am 08.03.2025]. Verfügbar unter: https://neo4j.com/pricing/.
27. Rathle P. ISO GQL: A Defining Moment in the History of Database Innovation [Online]. Graph Database & Analytics. 2024 [aufgerufen am 08.03.2025]. Verfügbar unter: https://neo4j.com/blog/cypher-and-gql/gql-international-standard/.
28. ISO/TS 22218-1:2023 [Online]. ISO. [aufgerufen am 25.09.2024]. Verfügbar unter: https://www.iso.org/standard/72899.html.
29. Graph database. In: Wikipedia [Online]. 2025 [aufgerufen am 08.03.2025]. Verfügbar unter: https://en.wikipedia.org/w/index.php?title=Graph_database&oldid=1276475700.
30. The Apache Software Foundation. Apache TinkerPop: Data System Providers [Online]. [aufgerufen am 08.03.2025]. Verfügbar unter: https://tinkerpop.apache.org/providers.html.
31. Anadiotis G. Graph Database market update September 2024 [Online]. Medium. 2024 [aufgerufen am 26.02.2025]. Verfügbar unter: https://medium.com/@linked_do/graph-database-market-update-september-2024-4a213b482bf4.
32. Rathle P. The GraphRAG Manifesto: Adding Knowledge to GenAI [Online]. Graph Database & Analytics. 2024 [aufgerufen am 09.03.2025]. Verfügbar unter: https://neo4j.com/blog/genai/graphrag-manifesto/.
33. World Wide Web Consortium W3C. Web Ontology Language (OWL) [Online]. [aufgerufen am 15.03.2025]. Verfügbar unter: https://www.w3.org/OWL/.
34. Studer R, Benjamins VR, Fensel D. Knowledge engineering: Principles and methods. Data Knowl Eng. 1998;25(1):161-97.
35. Gruber TR. Toward Principles for the Design of Ontologies Used for Knowledge Sharing [Online]. Verfügbar unter: https://tomgruber.org/writing/onto-design.pdf.
36. Jupp S. Building a Repository of Biomedical Ontologies with Neo4j [Online]. 2016 [aufgerufen am 20.03.2025]. Verfügbar unter: https://www.youtube.com/watch?v=jZND2WYT4GE.
37. Vocabularies and Ontologies [Online]. 2019 [aufgerufen am 17.03.2025]. Verfügbar unter: https://www.youtube.com/watch?v=yX0TDFx7Obw.
38. SNOMED International. What is SNOMED CT [Online]. SNOMED International. [aufgerufen am 26.03.2025]. Verfügbar unter: https://www.snomed.org/what-is-snomed-ct.
39. Ghorbani A, Davoodi F, Zamanifar K. Using type-2 fuzzy ontology to improve semantic interoperability for healthcare and diagnosis of depression. Artif Intell Med. 2023;135:102452.
40. Orphanet [Online]. [aufgerufen am 09.03.2025]. Verfügbar unter: https://www.orpha.net/.
41. RD-CODE. Facility for the visualisation of the hierarchical organisation of the ORPHA nomenclature [Online]. [aufgerufen am 09.03.2025]. Verfügbar unter: https://www.rd-code.eu/wp-content/uploads/2021/01/Facility-for-the-visualisation-of-the-Hierarchical-organsation-of-the-ORPHA-Nomenclature.pdf.
42. RD-CODE. https://dataviz.orphacode.org tooling example based on API implementation [Online]. orphanet-rare-diseases-issues; 2021 [aufgerufen am 09.03.2025]. Verfügbar unter: https://github.com/orphanet-rare-diseases-issues/dataviz_poc1.

43. Orphanet. Orphadata_aggregated/Orphanet Ontologies at master · Orphanet/Orphadata_aggregated [Online]. [aufgerufen am 09.03.2025]. Verfügbar unter: https://github.com/Orphanet/Orphadata_aggregated/tree/master/Orphanet%20Ontologies.
44. Orphadata. ORDO SPARQL Endpoint – Orphadata [Online]. [aufgerufen am 09.03.2025]. Verfügbar unter: https://www.orphadata.com/ordo-sparql-endpoint/.
45. Siddiqi A. Using Neo4j to visualize medicines' class and their ingredients [Online]. Adnan's Random bytes. 2022 [aufgerufen am 19.02.2025]. Verfügbar unter: https://blog.adnansiddiqi.me/using-neo4j-to-visualize-medicines-class-and-their-ingredients/.
46. Droste O, Merz C. Testmanagement in der Praxis [Online]. Berlin: Springer Vieweg; 2019 [aufgerufen am 12.11.2023]. Verfügbar unter: https://doi.org/10.1007/978-3-662-49653-4.
47. Docker Inc. Install Docker Engine on Debian [Online]. Docker Documentation. 2025 [aufgerufen am 18.02.2025]. Verfügbar unter: https://docs.docker.com/engine/install/debian/.
48. Neo4j Inc. Aura [Online]. Graph Database & Analytics. [aufgerufen am 02.03.2025]. Verfügbar unter: https://neo4j.com/product/auradb/.
49. Neo4j Inc. Prerequisites and system requirements - Neo4j Bloom [Online]. Neo4j Graph Data Platform. [aufgerufen am 02.03.2025]. Verfügbar unter: https://neo4j.com/docs/bloom-user-guide/current/bloom-installation/bloom-prerequisites/.
50. Neo4j Inc. Getting started with Neo4j in Docker [Online]. Neo4j Graph Data Platform. [aufgerufen am 18.02.2025]. Verfügbar unter: https://neo4j.com/docs/operations-manual/5/docker/introduction/.
51. National Library of Medicine. Medical Subject Headings [Online]. National Library of Medicine; [aufgerufen am 05.03.2025]. Verfügbar unter: https://www.nlm.nih.gov/mesh/meshhome.html.
52. SwissDRG AG. Datenschutz und Nutzungsbedingungen [Online]. [aufgerufen am 02.04.2025]. Verfügbar unter: https://www.swissdrg.org/de/datenschutz-und-nutzungsbedingungen.
53. European Medicines Agency (EMA). Download medicine data [Online]. 2018 [aufgerufen am 21.05.2025]. Verfügbar unter: https://www.ema.europa.eu/en/medicines/download-medicine-data.
54. Stiftung Refdata. Nutzungsbestimmungen AIPS / SAI [Online]. [aufgerufen am 02.04.2025]. Verfügbar unter: https://www.refdata.ch/de/nutzungsbestimmungen-aips-sai.
55. Norwegian Institute of Public Health, WHO Collaborating Centre for Drug Statistics Methodology. ATC/DDD Index and Guidelines [Online]. [aufgerufen am 21.05.2025]. Verfügbar unter: https://atcddd.fhi.no/atc_ddd_index_and_guidelines/atc_ddd_index/.
56. Bundesamt für Gesundheit. Arzneimittel [Online]. [aufgerufen am 21.05.2025]. Verfügbar unter: https://www.bag.admin.ch/bag/de/home/versicherungen/krankenversicherung/krankenversicherung-leistungen-tarife/Arzneimittel.html.
57. National Library of Medicine. MeSH Linked Data [Online]. [aufgerufen am 05.03.2025]. Verfügbar unter: https://id.nlm.nih.gov/mesh/.
58. Stiftung Refdata. Häufig gestellte Fragen [Online]. [aufgerufen am 17.04.2025]. Verfügbar unter: https://www.refdata.ch/de/faq.
59. Stiftung Refdata. Die Geschichte [Online]. [aufgerufen am 17.04.2025]. Verfügbar unter: https://www.refdata.ch/de/ueber-uns/die-geschichte.
60. Stiftung Refdata. SAI – Herunterladen [Online]. [aufgerufen am 17.04.2025]. Verfügbar unter: https://sai.refdata.ch/download.
61. Norwegian Institute of Public Health, WHO Collaborating Centre for Drug Statistics Methodology. ATC alterations from 2005-2024 [Online]. [aufgerufen am 22.05.2025]. Verfügbar unter: https://atcddd.fhi.no/atc_ddd_alterations__cumulative/atc_alterations/.

10 Selbständigkeitserklärung

Ich bestätige mit meiner Unterschrift, dass ich meine vorliegende Bachelor-Thesis selbstständig und ohne Benutzung anderer als der im Literaturverzeichnis angegebenen Quellen und Hilfsmittel angefertigt habe. Sämtliche Textstellen, die nicht von mir stammen, sind als Zitate gekennzeichnet und mit dem genauen Hinweis auf ihre Herkunft versehen.

Ich bestätige weiterhin, dass ich bei der Erstellung dieser Studienarbeit keine von einer KI erzeugte Inhalte übernommen habe. Es wurde lediglich die automatische Rechtschreibprüfung von MS Word verwendet. Der erarbeitete Source Code zum Erstellung des Datenmodells wurde punktuell mit Hilfe generativer KI (Microsoft Copilot, Github Copilot) erstellt oder vereinfacht.

Name, Vorname: Christian Franke

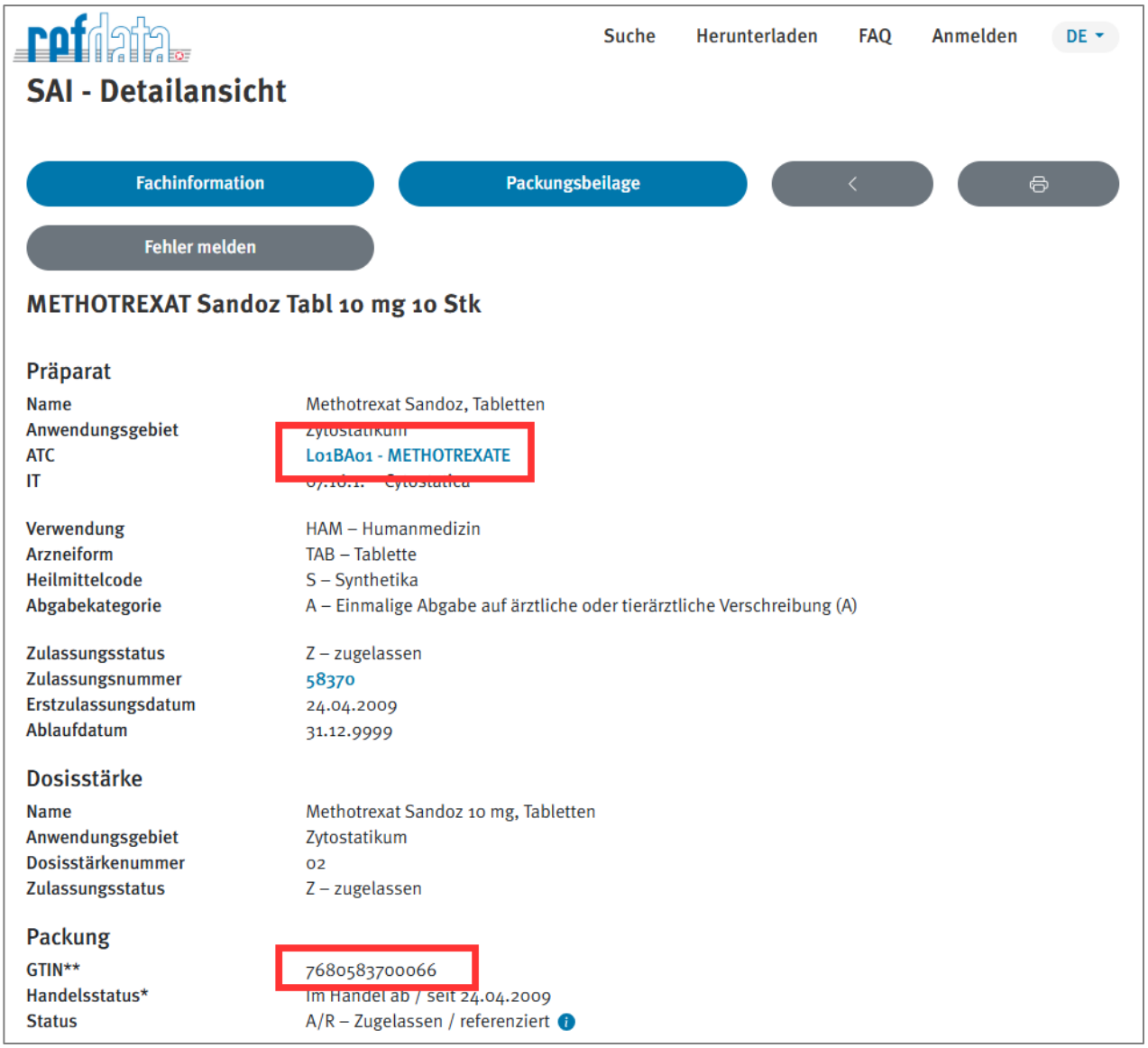
Ort, Datum: Bern, 09.06.2025

Unterschrift: 

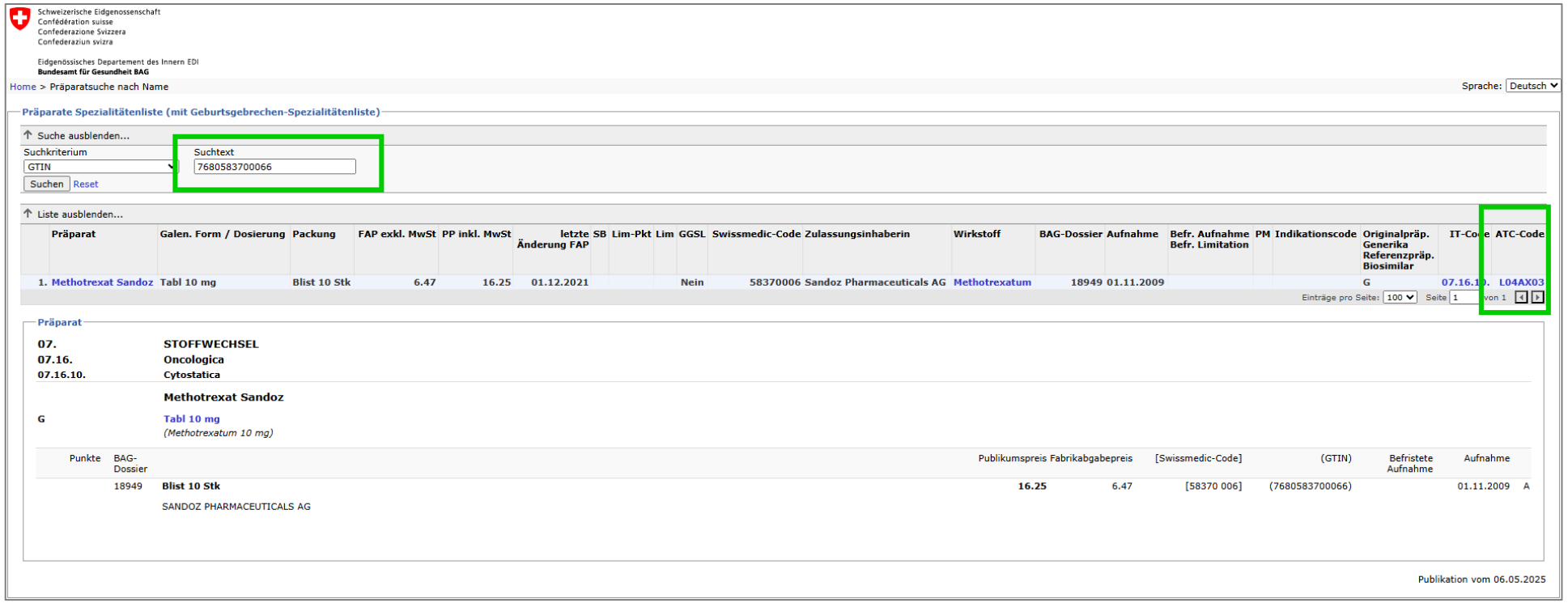
11 Anhang

11.1 Screenshots Methotrexat

Screenshot der strukturierten Arzneimittelinformationen (SAI) der Stiftung Refdata zeigen den alten ATC-Code (aufgenommen am 28.05.2025, 16:42 Uhr).



Screenshot der Spezialitätenliste (SL) des Bundesamts für Gesundheit zeigen den aktuellen ATC-Code (aufgenommen am 28.05.2025, 16:42 Uhr).



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement des Innern EDI
Bundesamt für Gesundheit BAG

Home > Präparatsuche nach Name

Sprache: Deutsch

Präparate Spezialitätenliste (mit Geburtsgebrechen-Spezialitätenliste)

Suche ausblenden...

Suchkriterium: GTIN

Suchtext: 7680583700066

Suchen | Reset

Liste ausblenden...

	Präparat	Galen. Form / Dosierung	Packung	FAP exkl. MwSt	PP inkl. MwSt	letzte Änderung FAP	SB	Lim-Pkt	Lim	GGSL	Swissmedic-Code	Zulassungsinhaberin	Wirkstoff	BAG-Dossier	Aufnahme	Befr. Aufnahme Befr. Limitation	PM	Indikationscode	Originalpräp. Generika Referenzpräp. Biosimilar	IT-Code	ATC-Code
1.	Methotrexat Sandoz	Tabl 10 mg	Blist 10 Stk	6.47	16.25	01.12.2021				Nein	58370006	Sandoz Pharmaceuticals AG	Methotrexatum	18949	01.11.2009				G	07.16.10.	L04AX03

Einträge pro Seite: 100 Seite 1 von 1

Präparat

07. STOFFWECHSEL
07.16. Oncologica
07.16.10. Cytostatica

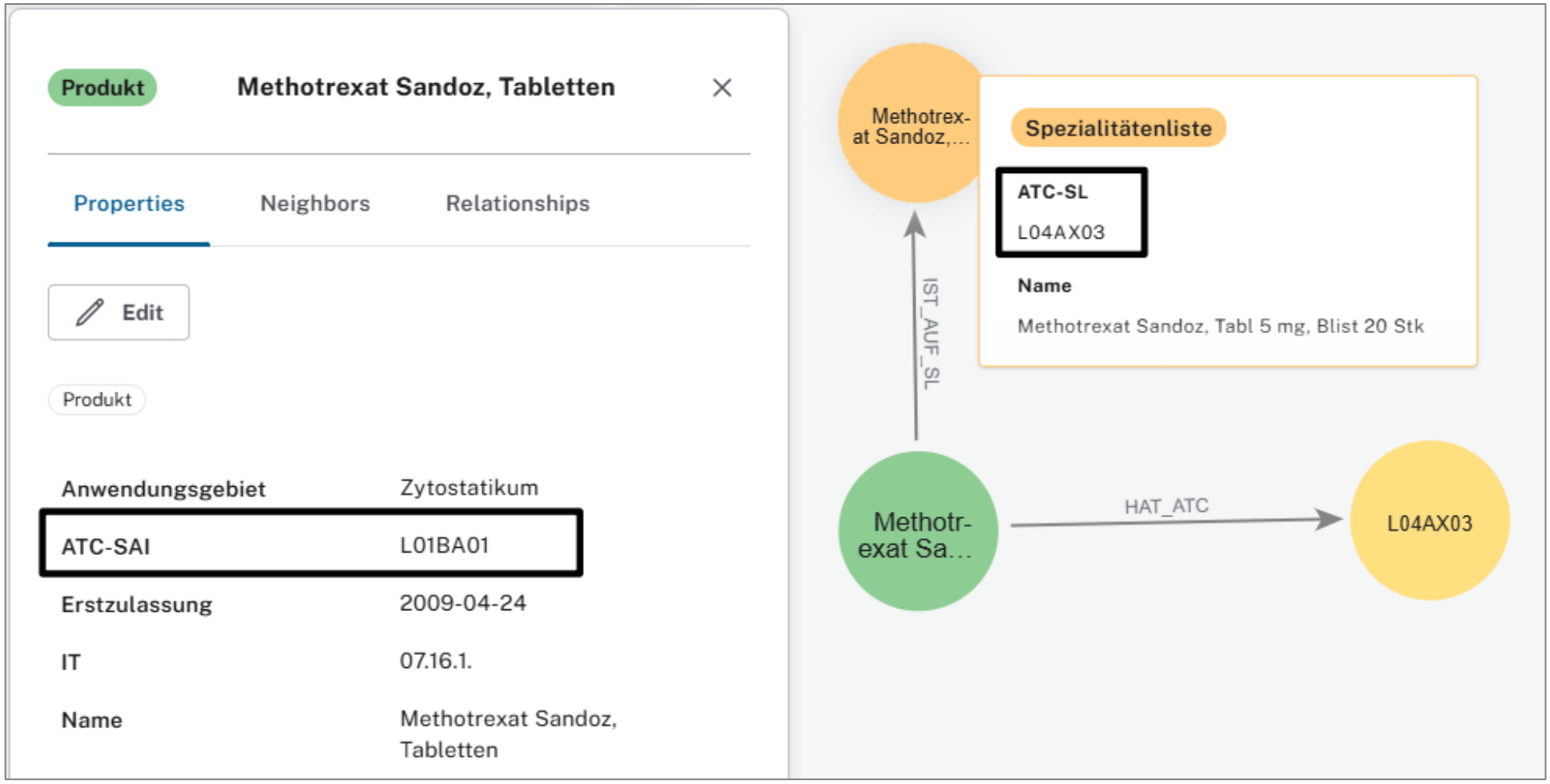
Methotrexat Sandoz

G Tabl 10 mg
(Methotrexatum 10 mg)

Punkte	BAG-Dossier		Publikumspreis	Fabrikabgabepreis	[Swissmedic-Code]	(GTIN)	Befristete Aufnahme	Aufnahme	
	18949	**Blist 10 Stk**	**16.25**	6.47	[58370 006]	(7680583700066)		01.11.2009	A
		SANDOZ PHARMACEUTICALS AG							

Publikation vom 06.05.2025

Screenshot des Datenmodells in Neo4j Explore: obwohl sich die ATC-Codes der SAI- und SL-Daten unterscheiden, wird die Relation der Methotrexat Sandoz Tabeletten mit dem korrekten ATC-Code L04AX03 hergestellt.



11.2 Preprocessing Schritte

Skript-Bezeichnung	Input (Ressource)	Beschreibung	Output
02_technisches_begleitblatt.R	Technisches_Begleitblatt_2025.xlsx	Excel-Datei wird von der SwissDRG Website heruntergeladen, die Daten werden in Deutsch, Französisch und Italienisch eingelesen. Anschliessend werden die Spaltenbezeichnungen vereinheitlicht und die zusammengeführten Daten als csv-Datei exportiert.	technisches_begleitblatt.csv
03_zusatzentgelte.R	ZE-EDV_V14.3_20241211-definitions.csv ZE_tarpsy_reha.csv	Die SwissDRG Zusatzentgelte (ZE) stehen als gezippte csv-Dateien zum Download auf der Website der SwissDRG AG bereit. Die Zusatzentgelte von TARPSY und ST Reha sind als Excel-Dateien vorhanden und wurden manuell in die gleiche Form wie die SwissDRG Zusatzentgelte gebracht. Die SwissDRG ZE werden heruntergeladen, entzippt, zusammen mit den ZE von TARPSY und ST Reha als Data-Frame eingelesen, nur die Medikamenten-Zusatzentgelte gefiltert und aggregiert.	ze_definitions.csv
04_ema_produkte.R	medicines_output_medicines_report_en.xlsx	Eine Excel-Datei wird von der EMA-Website heruntergeladen und die zugelassenen Humanarzneimittel filtert.	ema_products.csv
05_refdata.R	SAI/SAI-Praeparate.XML SAI/SAI-Adressen.XML	Eine zip-Datei mit den monatlichen Daten der strukturierten Arzneimittel werden von der Refdata-Website heruntergeladen (Achtung: Monats-Link aktualisieren!) und zwei relevante xml-Dateien extrahiert. Die Daten der xml-Dateien werden mit dem xml2-Package und XPath-Anweisungen in ein Data-Frame überführt. Alle Präparete, die nicht mehr zugelassen sind, werden dabei ignoriert. Eine Reihe von ATC-Codes wird aktualisiert, um die Datenqualität zu verbessern. Der Präparatename wird extrahiert, im dem der Text vor einem Komma als Kurzname übernommen wird. Mehrfach vorkommende Kurznamen werden gekennzeichnet. Die Zulassungsinhaberinnen werden aus SAI-Adressen.XML als Data-Frame extrahiert und dem Data-Frame mit den Präparaten hinzugefügt.	sai_praeparate.csv
06_parse_atc_ddd.py		ATC-Liste und die DDD-Angaben werden mit Hilfe eines Webscraping von https://atcddd.fhi.no/atc_ddd_index heruntergeladen. Der Webscraper durchsucht den DOM nach den HTML-Elementen, die die ATC-Codes, englischen Bezeichnungen und Tabellen mit DDD-Angaben enthalten. Dabei wird die Website rekursiv aufgerufen, beginnend bei ATC-Code mit einem Buchstaben und endend bei den siebenstelligen ATC-Codes. Ohne time.sleep(10) Anweisung läuft das Skript ca. 5 Minuten.	atc_list.csv atc_ddd.csv

07_bag_sl.R	Publications.xlsx	Die Excel-Datei wird von der BAG-Website heruntergeladen, die fünfstellige Zulassungsnummer extrahiert und das Datum der letzten Preisänderung mit dem SL-Einfügedatum aktualisiert, falls das letzte Preisdatum leer ist. Der Pfad zu der Excel-Datei ändert sich nicht, sodass das Preprocessing ohne monatliche Anpassung laufen gelassen werden kann.	sl.csv
		Die kumulierte Liste der ATC-Änderungen wurde der Website https://atcddd.fhi.no/atc_ddd_alterations__cumulative/atc_alterations/ entnommen und manuell aufbereitet, um die Fussnoten zu entfernen. Der Output bildet die Grundlage für die Substanzen, deren ATC-Codes sich in der Vergangenheit geändert hatten.	atc_alterations.csv
09_antraege.R	proposals ZE.xlsx proposals Medi.xlsx	In den Excel-Daten für Anträge aus den Bereichen Zusatzentgelt und Medi-Liste wurden die aktuellen ATC-Codes manuell aus dem Antragstext herausgesucht. Die Excel-Dateien wurden in R in einem gemeinsamen Data-Frame zusammengefasst und die abgelehnten oder nicht rechenbaren Anträge gefiltert.	antraege.csv

11.3 GitLab Repository

Im GitLab Repository https://gitlab.ti.bfh.ch/franc2/bachelor-thesis werden alle relevanten Unterlagen für das Projekt zusammengestellt. Die Struktur des Repository lautet wie folgt:

`/Deliverables`
- Hier befinden sich die Abgabedokumente als PDF-Dateien (Abschlussbericht, Abschlusspräsentation, Poster, Bookeintrag)
- Ausserdem ist das Datenbank-Backup `neo4j.dump` mit Daten vom Juni 2025 vorhanden, sodass das Backup in eine Neo4j-DB importiert werden kann
- Direktlink

`/Projektmanagement`
- Hier befinden sich administrative Dokumente für das Projekt.
  Diese bestehen aus dem Projektmanagementplan, dem Gantt-Chart als Excel-Datei, der Kooperationsvereinbarung und mehreren Dateien für die Zeiterfassung (R-Skript, das die Time Tracking Daten von der GitLab GraphQL API zieht und eine Grafik mit ggplot2 sowie csv-Export erstellt)
- Direktlink

`/Protokolle`
- Hier befinden sich zwei Markdown-Dateien mit den anti-chronologischen Protokollen einerseits mit Betreuer und Experten und andererseits mit den Stakeholdern der SwissDRG AG
- Direktlink

`/src/main`
- Hier befindet sich sämtlicher Code, der für die Erstellung des Graph-Datenmodells nötig ist
- Im Ordner `/data_model` sind die cypher-Skripte und insb. die finale Datei `99_load_all_into_neo4j.cypher` zu finden
- Im Ordner `/documentation` befinden sich das `Benutzerhandbuch.MD` und das technische Handbuch als `README.MD` sowie die Screenshots der Testdokumentation in Unterordnern
- Im Ordner `/import` befinden sich die csv-Dateien, mit denen das Datenmodell in Neo4j aufgebaut wird. Bis auf `geloeschte_substanzen.csv` und `atc_alterations.csv` werden alle Import-Dateien mit den Preprocessing-Skripten erstellt
- Im Ordner `/perspectives` befindet sich eine Perspektive für Neo4j Bloom/Explore, die gespeicherte Abfragen enthält und die Knotendarstellung optisch verfeinert
- Im Ordner `/preprocessing` sind Skripte enthalten, um die Import-Dateien zu erstellen
- Direktlink

`/src/resources`
- In diesen Ordner werden Dateien des Preprocessings zwischengespeichert und dort befinden sich manuell aufbereitete Zwischendateien
- Durch das Preprocessing werden die vorhandenen Dateien teilweise überschrieben und/oder gelöscht
- Direktlink

`/src/test`
- Hier befindet das `Test-Protokoll.md`, die Szenarien für den User-Test sowie Dateien, die für Entwicklung und Testing des Datenmodells angefallen sind
- In Unterordnern befinden sich die Dokumente je Themengebiet
- Direktlink