Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektro- und Informationstechnik

Studiengang Industrie-4.0-Informatik

Bachelorarbeit

von

Erik **Barthelmann**

Erstellung eines Datenmodells für die Integration unterschiedlicher Datenquellen in einem Werksstandort mittels Anchor Modeling und der Realisierung in SDI

Creating a data model for the integration of different data sources in a plant location with Anchor Modeling

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektro- und Informationstechnik

Studiengang Industrie-4.0-Informatik

Bachelorarbeit

von

Erik **Barthelmann**

Erstellung eines Datenmodells für die Integration unterschiedlicher Datenquellen in einem Werksstandort mittels Anchor Modeling und der Realisierung in SDI

Creating a data model for the integration of different data sources in a plant location with Anchor Modeling

Bearbeitungszeitraum: von 15. März 2023
bis 14. August 2023

1. Prüfer: Prof. Dr.-Ing. Christoph P. Neumann

2. Prüfer: Prof. Dr. Fabian Brunner

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektro- und Informationstechnik

Ostbayerische Technische Hochschule
Amberg-Weiden

Bestätigung gemäß § 12 APO

Name und Vorname
des Studenten: **Barthelmann, Erik**

Studiengang: **Industrie-4.0-Informatik**

Ich bestätige, dass ich die Bachelorarbeit mit dem Titel:

Erstellung eines Datenmodells für die Integration unterschiedlicher Datenquellen in einem Werksstandort mittels Anchor Modeling und der Realisierung in SDI

selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Datum: 14. August 2023

Unterschrift:

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektro- und Informationstechnik

Ostbayerische Technische Hochschule
Amberg-Weiden

Bachelorarbeit Zusammenfassung

Student (Name, Vorname): **Barthelmann, Erik**
Studiengang: Industrie-4.0-Informatik
Aufgabensteller, Professor: Prof. Dr.-Ing. Christoph P. Neumann
Durchgeführt in (Firma): SIEMENS AG
Betreuer in Firma: Herr Dickas Heiko
Ausgabedatum: 15. März 2023 Abgabedatum: 14. August 2023

Titel:

Erstellung eines Datenmodells für die Integration unterschiedlicher Datenquellen in einem Werksstandort mittels Anchor Modeling und der Realisierung in SDI

Zusammenfassung:

Durch die Digitalisierung und Vernetzung von Maschinen und Anlagen entstehen immer mehr Daten. Dabei entstehen Datenquellen in verschiedenen Formaten und mit unterschiedlichen Strukturen. Liegen die Daten in einem einheitlichen Format vor, kann die Analyse erleichtert werden. Dies ist die Aufgabe der Datenintegration, die schon seit Jahren in der Informatik erforscht wird. Es wurden verschiedene Ansätze und Applikationen entwickelt, die für die Datenintegration verwendet werden können. SDI (Semantic Data Interconnect) ist eine Applikation, welche einige dieser Ansätze implementiert. Die von Siemens entwickelte und über Insights Hub bereitgestellte Applikation vereinheitlicht Daten und ermöglicht das Erstellen von Abfragen auf diesen. Um die Daten einheitlich abzufragen, wird ein gemeinsames Datenmodell benötigt. Mit den Anforderungen der heutigen Industrie hat sich das Anchor Modeling entwickelt. Diese Modellierungsmethode sorgt für eine stabile Schnittstelle, die gleichzeitig einfach zu erweitern ist. Untersucht wurde die Anwendbarkeit von Anchor Modeling für die Datenintegration in SDI. Dabei wurde ein Anwendungsfall aus der Industrie betrachtet, bei dem Daten aus verschiedenen Quellen integriert und bereitgestellt werden sollen. Bei diesem Anwendungsfall handelt es sich um Daten, die benötigt werden, um Produkte rückverfolgen zu können. Für diesen Anwendungsfall soll ein Anchor-Model-basiertes Datenschema entworfen und in SDI implementiert werden.

Schlüsselwörter: Datenintegration, Datenmodellierung, Anchor Modeling

Inhaltsverzeichnis

Symbole, Formelzeichen und Einheiten

CSV Comma Separated Values
IDL Integrated Data Lake
IOT Internet of Things
JSON JavaScript Object Notation
MLFB Maschinen-lesbare Fabrikbezeichnung
OWL Web Ontology Language
RDF Resource Description Framework
RDFS Resource Description Framework Schema
SDI Semantic Data Interconnect
SLR Systematic Literature Review
SQL Structured Query Language
URI Uniform Resource Identifier
W3C World Wide Web Consortium
XML Extensible Markup Language

Kapitel 1

Einleitung

In der Industrie werden kontinuierlich Daten produziert, weitergeleitet und verarbeitet. Mit der Vernetzung durch Industrie 4.0 und dem Internet of Things (IOT) ist die Menge der zu verarbeitenden Daten jedoch explodiert (vgl. [15]). Diese Daten werden auf verschiedene Arten erzeugt und haben meist auch anwendungsspezifische Formate. Durch diese Vielfalt an Datenquellen ist es schwierig, die Daten in angemessener Art und Weise zu analysieren und zu verarbeiten, da diese erst miteinander verknüpft werden müssen (vgl. [4]). Deswegen setzt sich das Feld der Datenintegration „*die Konsolidierung der Daten mit dem Ziel, möglichst vollständige, einheitliche und korrekte Datenbestände zur Verfügung zu stellen*“([14, S.18]) als Aufgabe. Die Datenintegration hat, seit ihren Anfängen in den 80ern (siehe [43]) schon viele verschiedene Lösungen entwickelt, um Daten zu integrieren (vgl. [14, S.13]).

Es hat sich gezeigt, dass die Semantik ein wichtiger Aspekt ist, der auch bei der Datenintegration beachtet werden muss. Verschiedene Begriffe können in unterschiedlichen Kontexten andere Bedeutungen haben und somit zu Missverständnissen führen (vgl. [44, S.222]). Selbst im natürlichen Sprachgebrauch gibt es solche Phänomene, wie das Homonym ‚Bank‘ welches eine Sitzgelegenheit oder ein Geldinstitut benennen kann. Ontologien sind konzeptuelle Modelle, die die Bedeutung von Begriffen und Konzepten beschreiben, und dadurch bei diesen Problemen helfen können. Durch sie können Begriffe und Konzepte in einen semantischen Zusammenhang gebracht werden. Aufgrund dieser Eigenschaften haben Ontologien auch ihren Weg in die Datenintegration gefunden (vgl. [44, S.225 f.]).

Der von Siemens entwickelte Service Semantic Data Interconnect (SDI) ist ein System, welches einige Ansätze für die Datenintegration implementiert. Darunter fällt der einheitliche Zugriff auf verschiedene Datenquellen, sowie die Möglichkeit, Ontologien zu erstellen und mit Datenquellen zu verknüpfen. Da dieser Service über Insights Hub, ein cloud-basiertes Ökosystem von Siemens, bereitgestellt wird, ist es in der Lage, mit anderen Services von Insights Hub zu kommunizieren. Dies ermöglicht eine einfache und integrierte Nutzung des Services innerhalb des Ökosystems (vgl. [22]).

Das Erstellen einer Ontologie ist eine schwierige und vielschichtige Aufgabe (vgl. [44, S.159]). Ähnlich wie beim Erstellen eines Datenmodells gibt es verschieden Ansätze

und Möglichkeiten, Sachverhalte zu modellieren. Zusätzlich zu den Anforderungen, die aus der Begründung der Datenintegration hervorgehen, kommen noch weitere durch die Weiterentwicklung der Industrie hinzu. Durch die schnelle und ständige Veränderung einer Industrie 4.0 Umgebung (vgl. [48, S.1]), müssen die Datenmodelle und Ontologien einfach erweiterbar und veränderbar sein. Dabei ist es wichtig bereits bestehende Systeme durch diese Änderungen nicht zu beeinflussen. Das Modell muss also stabil sein.

Mit der Entwicklung von Anchor Modeling, wurde ein Ansatz entwickelt, um Datenmodelle zu erstellen, die sich leicht verändern lassen. Lars Rönnbäck entwickelte dieses Modellierung basierend auf Anforderungen für Data Warehouses (vgl. [2]). Ein Data Warehouse ist eine etablierte Lösung für die Datenintegration, bei der die Daten in einem Datenspeicher abgelegt werden und von dort aus abgerufen werden können (vgl. [30, S.8]). Durch ihren hohen Grad an Normalisierung, bieten Anchor Models, ein stabiles Datenmodell.

Anchor Modeling bietet die Stabilität (vgl. [40, S.1248 f.]), die von heutigen Industrieumgebungen gefordert wird. Aufgrund dieser Eigenschaften soll es in dieser Arbeit für die Datenintegration genutzt werden. Dabei soll überprüft werden, ob sich Anchor Models eignen, um als ein Datenmodell für die Bereitstellung zu dienen. Als Integrationssystem wird SDI verwendet. Damit soll eine Ontologie erstellt werden, die das Anchor Model implementiert. Weiterhin soll ermittelt werden, ob SDI die nötigen Anforderungen und Funktionen bietet, um solche Modelle zu realisieren. Dafür soll ein Datenschema für einen gegebenen Anwendungsfall erstellt und in SDI umgesetzt werden.

Der Anwendungsfall basiert auf der Rückverfolgung von Produkten in einer Produktionsumgebung. Dabei werden verschiedene Informationen zu einem Produkt gesammelt, damit die einzelnen Materialien und Hersteller ermittelt werden können. Für diesen Anwendungsfall existieren mehrere Datenquellen. Diese Datenquellen sollen miteinander verknüpft und deren Inhalt über eine Schnittstelle bereitgestellt werden. Diese Daten können dann für verschiedene Anwendungen, wie z.B. Rückverfolgung, Visualisierungen oder Qualitätskontrolle, genutzt werden. Anhand dieses beispielhaften Anwendungsfalls soll die Eignung von Anchor Modeling und SDI für die Datenintegration überprüft werden. Zudem stellt das Datenschema eine Grundlage für die Entwicklung eines Werksstandortweiten Datenschema dar. Deshalb ist es wichtig, dass das erstellte Schema stabil ist und sich leicht erweitern lässt.

Aufbau der Arbeit

Zunächst werden die Grundlagen für die Arbeit erläutert. Dazu wird auf Datenintegration, Ontologien und Anchor Modeling eingegangen. Diese Konzepte werden grundlegend erklärt und es wird die Motivation der einzelnen Themengebiete verdeutlicht. Des Weiteren wird die Applikation SDI vorgestellt. Dazu wird ihre Funktionsweise erläutert und die einzelnen Oberflächen und Konzepte beschrieben.

Danach beginnt der Prozess der Datenmodellierung. Dafür wird mit einem Überblick

über den Anwendungsfall begonnen, um die Anforderungen an das Datenschema zu ermitteln. Danach wird eine Datenanalyse durchgeführt, um den Aufbau und die Struktur der Daten verstehen und bewerten zu können. Im Anschluss wird das Datenschema mittels Anchor Modeling erstellt. Dabei werden die einzelnen Modellierungsschritte gezeigt, begründet und ggf. mit Alternativen erweitert.

Nachdem ein fertiges Schema erstellt wurde, wird es in SDI implementiert. Dabei wird das komplette Vorgehen zur Implementierung des Schemas in SDI gezeigt. Es werden Schwierigkeiten und Probleme aufgelistet und diskutiert. Für einige Probleme werden auch verschiedene Lösungsvorschläge erarbeitet.

Im Anschluss wird es eine Diskussion über die Arbeit geben. Dabei wird auf die Ergebnisse der Arbeit eingegangen und diese bewertet. Dabei wird auch auf die Grenzen der Arbeit sowie die Bewertung der Methodik eingegangen. Außerdem werden weiter Aspekte und Alternativen aufgezeigt, die nicht in der Arbeit behandelt wurden. Zum Schluss wird das erstellte Schema sowie die Umsetzung in SDI mittels einer Fitness for Use Analyse ausgewertet. Dabei handelt es sich um eine qualitative Analysetechnik zur Bewertung eines Produktes (vgl. [32, S.52 f.]).

Kapitel 2

Grundlagen

Zur Klärung der Grundlagen sollen Begrifflichkeiten definiert und erläutert werden, die für das Verständnis der Arbeit wichtig sind. Zusätzlich sollen Konzepte und Technologien aus den verschiedenen Bereichen, in denen sich die Arbeit bewegt, vorgestellt werden. Dazu wird zuerst das Thema Datenintegration behandelt. Dabei werden die Grundlagen, Rollen, Aufgaben und Herausforderungen innerhalb von Datenintegration erläutert. Ontologien werden heutzutage genutzt, um Wissen zu repräsentieren und stellen ein nützliches Mittel bei der Datenintegration dar (vgl. [39, S.162]). Deshalb wird beschrieben, wie sich Ontologien entwickelt haben und wie diese aufgebaut sind. Außerdem wird differenziert, welche Arten von Ontologien es gibt und wie Ontologien in SDI zu betrachten sind. Danach wird das Tool SDI vorgestellt. Dabei werden die verschiedenen Funktionen und die Nutzungsweise der Applikation gezeigt. Zum Schluss wird noch das Thema Anchor Modeling behandelt. Dort wird erklärt aus welchen Komponenten ein Anchor Model besteht und wie es aufgebaut ist. Des Weiteren werden Vor- und Nachteile sowie die Besonderheiten der Modellierungsmethode erläutert.

2.1 Datenintegration

Die Datenintegration stellt auch in der heutigen Zeit noch eine Herausforderung dar. Es entsteht eine immer größer werdende Menge aus unterschiedlichen Daten, Datenquellen und Datenformaten. Der Grund dafür ist die Unmenge an Daten, welche ständig durch Sensoren, Applikationen, von Menschenhand oder durch Maschinen produziert werden (vgl. [14, S.16 ff.]). Für Menschen, sowie auch für Maschinen, ist es oft nicht leicht, mit vielen unterschiedlichen Formaten zu arbeiten. Es bedarf neben dem Wissen über Aufbau und Struktur der Nutzdaten, auch Wissen über Metadaten, wie Speicherorte und Gültigkeit der Nutzdaten. Um die Arbeit mit den verschiedenen Daten zu erleichtern, werden zentrale Datenbanken eingesetzt (vgl. [9, S.9]). Datenbanken ermöglichen und regeln einen einheitlichen Zugriff auf die Daten und stellen eine Schnittstelle zur Verfügung. Nutzenden, die über wenig Kenntnis von Aufbau und Struktur der Rohdaten verfügen, ermöglicht diese Schnittstelle, Daten zu analysieren.

Es erleichtert die Verknüpfung von Daten zur Gewinnung neuer Erkenntnisse. Auch kann die Entwicklung von Applikationen durch eine stabile Schnittstelle vereinfacht werden (vgl. [1, S.1]).

2.1.1 Aufgaben

Die eigentliche Aufgabe der Datenintegration ist es, Daten aus unterschiedlichen, heterogenen Datenquellen zu verschmelzen (vgl. [14, S.47 f.]). Das bedeutet, verschiedene Datenquellen (unabhängig von der Form der Daten) werden analysiert und in einer einzelnen Datenquelle integriert. Diese Datenquelle stellt eine einheitliche Schnittstelle bereit, von welcher aus alle Nutzer auf die Daten zugreifen können (vgl. [9, S.11]). Heterogene Datenquellen sind Datenquellen, welche unterschiedliche Datenformate, -strukturen und -modelle beinhalten können (vgl. [9, S.8]). Zwei unterschiedliche Tabellen, die identisch aufgebaut sind und denselben Inhalt beschreiben (die Werte können sich unterscheiden), sind nicht heterogen. Über das zentrale Datensystem sollen Entwickler und Datenwissenschaftler (engl. Data Scientist) in der Lage sein, auf alle Daten zuzugreifen. Dafür muss das System in der Lage sein, auf die verschiedenen Datenquellen zuzugreifen. Der Aufbau der Rohdaten muss nur für das Integrationssystem bekannt sein, welches dafür sorgt, diese Rohdaten in einer angemessenen und einheitlichen Form bereitzustellen (vgl. [9, S.10 ff.]).

Für die Implementierung des Datenintegrationssystems gibt es verschiedene Architekturen. In einer möglichen Architektur nimmt das System die Daten in einem dedizierten Speicher auf und kann dadurch auf die Daten zugreifen. Bei diesem Ansatz spricht man oft von einem Data Warehouse. Eine weitere Möglichkeit ist die sogenannte virtuelle Integration. Bei dieser Form werden die Daten nicht an einem zentralen Ort gespeichert. Das Integrationssystem weiß, wo die Daten liegen und kann diese von Maschinen oder anderen Quellen abrufen, wenn sie benötigt werden (vgl. [9, S.10 ff]). Über das System können dann Abfragen und andere Datenanalyse-Techniken genutzt werden, um die Daten zu verarbeiten. Die Zentralisierung und Vereinheitlichung bringt einige Vorteile mit sich. Eine bessere Kommunikation zwischen Nutzern der Daten wird durch die Verwendung von einheitlichen Begriffen und Datenstrukturen gewährleistet (vgl. [17, S.301f.]). Zudem können Daten aus vorher getrennten Domänen miteinander verknüpft und analysiert werden, was ohne ein zentrales Schema mit viel Zeit und Aufwand verbunden ist.

Um dieser Aufgabe zu begegnen, benötigt es sorgfältiger Vorbereitung und Planung. Datenquellen müssen analysiert werden, um die Struktur und die Semantik der Daten zu verstehen. Oft ist einfaches Wissen über Daten und Datenstrukturen nicht ausreichend und man benötigt Expertenwissen aus der Domäne. Zwei Verfahren, das Schema Matching (dt. Schema Vergleich) und das Schema Mapping (dt. Schema Übersetzung), sollen bei dem Integrationsprozess unterstützen. Sie helfen dabei, Verknüpfungen zwischen den Datenquellen zu finden und diese zu beschreiben.

Das Schema Matching bezeichnet das Auffinden von Verknüpfungen zwischen den Datenquellen. Dabei geht es nicht um die genaue Beschreibung der Verknüpfung, sondern um die allgemeine Existenz derer. Es gibt keinen Algorithmus, der alle richtigen

Beziehungen zwischen Datenquellen finden kann. Daher gibt es für diesen Schritt verschiedene Mechanismen und Methoden, die bei der Findung solcher Beziehungen helfen sollen (vgl. [20, S. 25]). Mit einem sogenannten Matcher versucht man Beziehungen zwischen den Datenquellen zu identifizieren (vgl. [9, S. 128]). Einen Matcher kann man entweder auf den Inhalt der einzelnen Spalten anwenden oder auf die Metadaten, also die Spaltennamen und Datentypen (vgl. [20, S.27 ff.]). Ebenso ist eine Kombination aus beiden Ansätzen möglich. Oft wird ein Matcher durch einen String- und Value-Matching-Algorithmus implementiert. Es könnte zḂ. die Levenshtein-Distanz genutzt werden, um die Ähnlichkeit von Strings zu bestimmen. Die Levenshtein-Distanz gibt an, wie viele Operationen nötig sind, um einen String in einen anderen zu transformieren. Es können ebenso Wörterbücher genutzt werden, um die Ähnlichkeit von Begriffen zu bestimmen, welche domänenspezifisch abgestimmt sind (vgl. [20, S.29, S.34]). Ein Matcher gibt eine Menge von möglichen Beziehungen zurück. Jeder Beziehung wird eine Gewichtung zugeordnet. Die Gewichtung gibt an wie sicher der Matcher ist, dass die Beziehung tatsächlich existiert (vgl. [9, S.129 f.]). Meist werden mehrere unterschiedliche Matcher eingesetzt, um die Beziehungen zu identifizieren. Die Ergebnisse der einzelnen Matcher werden zu einem Gesamtergebnis zusammengeführt. Dabei werden den Ergebnissen der einzelnen Matcher Gewichte zugeordnet und daraus ein Endergebnis erstellt (vgl. [9, S.128]). Die erkannten Beziehungen können darauf untersucht werden, ob bestimmte (bekannte) Gesetzmäßigkeiten verletzt werden oder unerlaubte Beziehungen vorhanden sind (vgl. [9, S.135]). Zum Schluss muss entschieden werden, welche der gefundenen und erlaubten Beziehungen tatsächlich genutzt werden (vgl. [9, S.143 f.]). Beim Schema Matching werden mittlerweile Methoden des Machine Learnings eingesetzt, um den Prozess zu automatisieren (vgl. [20, S.33]).

Nachdem die Beziehungen identifiziert wurden, müssen diese genauer definiert werden. Das ist die Aufgabe des Schema Mapping (vgl. [9, S.129]). Hierbei wird definiert, welche Operationen auf den Datenquellen nötig sind, um die gewünschten Beziehungen zu erhalten. Dafür können Verbunde (engl. Join), Vereinigungen (engl. Union), Aggregation oder auch Filterungen genutzt werden. Mit Verbunden können Tabellen über eine gemeinsame Spalte verknüpft werden. Vereinigungen fügen Tabellen zusammen, die die gleiche Struktur haben. In dieser Arbeit werden, anstelle der deutschen Begriffe Verbund und Vereinigung, die englischen Begriffe Join und Union verwendet. Eventuell müssen die Tabellenstrukturen geändert oder komplexe Strukturen aufgespalten werden (vgl. [9, S.152 f.]). Die Schwierigkeit bei dieser Aufgabe liegt darin, dass es unterschiedliche Möglichkeiten gibt, um die gewünschten Beziehungen zu erhalten. Es ist also nicht immer klar, welche Operationen die korrekten sind (vgl. [9, S.129]). Deswegen erfordert diese Aufgabe ein tiefes Verständnis der Datenquellen und der gewünschten Ergebnisse. Der Vorgang kann aber mithilfe eines Algorithmus unterstützt werden. Ein allgemeiner Algorithmus untersucht die Menge aller möglichen Operationen und schlägt die, für ihn am wahrscheinlichsten wirkende, vor. Die Wahrscheinlichkeit wird durch verschiedene Ansätze bestimmt, die bei der Erstellung von solchen Verknüpfungen erkannt wurden. Das Mapping, wenn auch heuristisch, liegt einfachen Prinzipien der Datenmodellierung zugrunde. Einige Verknüpfungen werden bevorzugt, da sie eher den Normalisierungs-/Denormalisierungsprinzipien entsprechen. Verknüpfungen lassen sich durch Joins oder Unions realisieren, die je

nach Aufbau und Struktur der Datenquellen zu bevorzugen sind. Eine Join Operationen wird bevorzugt, wenn die Datenquellen über Schlüssel verknüpft werden können. Schlüssel sind Spalten, die zur Identifizierung eines Datensatzes genutzt werden. Ein Primärschlüssel ist ein Schlüssel, der eindeutig einen Datensatz identifiziert (vgl. [41, S.31]). Ein Fremdschlüssel ist ein Schlüssel, der auf einen anderen Datensatz verweist (vgl. [41, S.38]). Meist ist er in dem verwiesenem Datensatz der Primärschlüssel. Unions werden genutzt, wenn ein Sachverhalt auf unterschiedliche Wegen berechnet werden muss. Dies könnte der Fall sein, wenn z.B. das Gehalt einer studentischen Hilfskraft anders berechnet wird, als das eines Professors. Sollen beide Gehälter in einer Datenbank integriert werden, müssen beide Berechnungen mittels einer Unionen in die Datenbank eingepflegt werden (vgl. [9, S.153 f.]). Diese Prinzipien werden in den Algorithmus implementiert, sodass dieser eine fundierte Entscheidung trifft. Dem Nutzer ist es dabei möglich, die Prinzipien für den entsprechenden Anwendungsfall anzupassen und dem Algorithmus Feedback zu geben. Die Eingabe für den Algorithmus ist das Ergebnis des Schema Matching (vgl. [9, S.156]).

2.1.2 Rollen

Der Prozess der Datenintegration ist ein komplexer Prozess, der die Zusammenarbeit verschiedener Personen und Rollen benötigt (vgl. [9, S.8]). Angefangen bei den Daten, die aus der Maschine kommen oder durch einen Prozess erzeugt werden, müssen im ersten Schritt diese Daten abgreifbar sein. Jemand muss die Daten also in einer bestimmten Form zur Verfügung stellen. Daten, welche nicht in elektronischer Form vorliegen, müssen digitalisiert werden. Im schlimmsten Fall sind die Daten nicht mal vorhanden (vgl. [9, S.8 f.]) und müssen erst, durch Sensoren oder per Hand, aus dem Prozess extrahiert werden.

Bei der Erstellung, Digitalisierung und Speicherung von Nutzdaten muss darauf geachtet werden, dass diese Daten so verwaltet sind, dass sie gut genutzt werden können. Dazu gehört die eventuelle Bereinigung und Umstrukturierung der Daten. Es wird eine strukturelle Ordnung benötigt, um auf die Daten zuzugreifen. Bei der Datenintegration geht es speziell darum, einen einheitlichen Zugriff auf die verschiedenen Datenquellen zu ermöglichen. Die Bereitstellung, Pflege und Wartung dieses Systems wird von einem Data Engineer übernommen. Dieser ist dafür zuständig, dass die Daten angemessen in das System integriert werden und die Datenquellen angebunden sind. Er sorgt dafür, dass die Daten so strukturiert sind, dass sie von anderen Nutzergruppen verwendet werden können (vgl. [11]). Um die Daten in ein einheitliches Format zu bringen, muss der Data Engineer mit den Domänenexperten zusammenarbeiten, um ein korrektes und vollständiges Datenmodell zu erstellen. Die Experten können zudem dabei helfen, wichtige Konzepte und Zusammenhänge zu identifizieren, die in ein Datenschema einfließen können. Dieses Datenmodell ist die Basis für jegliche weitere Nutzung der Daten. Daher ist eine enge Zusammenarbeit und ein hoher Anspruch an Datenqualität erforderlich. Sind die Nutzdaten im zentralen System vorhanden, ist es die Aufgabe des Data Scientist, diese Daten zu analysieren. Hierzu nutzt er die ihm bekannten Methoden, um Hypothesen mithilfe der Daten zu überprüfen, neue Erkenntnisse zu gewinnen und die Daten zu visualisieren (vgl. [45]).

2.1.3 Herausforderungen

Es gibt unterschiedliche Schwierigkeiten, die beim Integrieren der Datenquellen beachtet werden müssen. Diese Probleme beruhen meist auf der Heterogenität der Datenquelle, so wie den unterschiedlichen Möglichkeiten der Darstellung und Modellierung von Daten. Sie können nur bedingt automatisiert gelöst werden und müssen dabei stets von Experten kontrolliert werden. In der Literatur wird zwischen drei verschiedenen Arten von Problemen bei der Datenintegration unterschieden.

Um ein syntaktisches Problem handelt es sich, wenn die Datenquellen unterschiedliche Datenformate besitzen oder unterschiedliche Datenmodelle nutzen. Eine Datenquelle kann beispielsweise Daten in einer relationalen Datenbank speichern, während eine andere Datenquelle die Daten in einer Comma Separated Values (CSV)-Datei oder JavaScript Object Notation (JSON)-Datei ablegt. Zur Lösung dieses Problems können sogenannte Wrapper eingesetzt werden. Wrapper kapseln die Datenquelle und geben sie nach außen in einem einheitlichen Format weiter. Dafür wird oft Extensible Markup Language (XML) oder Resource Description Framework (RDF) genutzt (vgl. [44, S.216]).

Strukturelle Konflikte entstehen aufgrund unterschiedlicher Modellierung der Daten. Es ist möglich, dass dieselben Daten in einer Datenquelle in einer Tabelle gespeichert sind und in einer anderen Datenquelle in mehreren Tabellen abgespeichert sind (vgl. [9, S. 7, f.]). Tabelle 2.1 und Tabelle 2.2 stellen einen solchen Konflikt dar. Während in Tabelle 2.1 das Feld *Adresse Hauptsitz* genutzt wird, um die gesamte Adresse mit Straße, Hausnummer, Postleitzahl und Ort zu speichern, wird in Tabelle 2.2 die Adresse in mehrere Felder aufgeteilt. Im Sinne der Normalisierung wäre der richtige Schritt, die Adressen in Tabelle 2.1 in mehrere Felder aufzuteilen; eine nicht triviale Aufgabe. Wie in Reihe 2 von Tabelle 2.1 zu sehen ist, sind die Adressen nicht einheitlich aufgebaut. Manche Straßen- und Ortsnamen enthalten Leerzeichen, die die Trennung erschweren. Es ist nicht automatisch klar, welche Teile der Adresse in welches Feld übernommen werden sollen.

Lieferanten ID	Name	Adresse Hauptsitz
1	Reifen & CO	Innenstadtring 3, 80331 München
2	Schrauben GmbH	Industriestraße 46 10115 Berlin
3	Hans Müller	Kanton Straße, 92224 Amberg

Tabelle 2.1: Eine Tabelle in der die Adresse in einer Spalte gespeichert ist

ID	Lieferanten	Straße	PLZ	Ort
1	Holzbauten	Sattelgasse 4	80331	München
2	Schrauben GmbH	Industriestraße 46	10115	Berlin
3	Elektro Schmidt	West-Alle 5	92224	Amberg

Tabelle 2.2: Eine ähnliche Tabelle mit anderer Struktur

Auch können Probleme entstehen, wenn verschiedene Datenquellen unterschiedliche Begriffe für denselben oder gleiche Begriffe für unterschiedliche Sachverhalte nutzen.

Unterschiedlichen Datenquellen haben verschiedene Intentionen für denselben Begriff. Dazu kann auch die Verwendung von Abkürzungen und Fachbegriffen gezählt werden, deren Bedeutung sich nicht unmittelbar erschließen lässt (vgl. [44, S. 215, f.]). Bei dieser Art von Problemen spricht man von semantischen Problemen. Ein Beispiel: Ein Autohaus speichert unter dem Begriff Auto nur PKWs (die sie verkaufen). Ein Autoverleih hingegen speichert unter dem Begriff Auto, sowohl PKWs als auch Wohnwagen und Transporter oder auch LKWs. In diesem Fall ist die Intention des Begriffs *Auto* in beiden Datenquellen unterschiedlich. Die Intention des Autohauses ist hier in der Intention des Autoverleihs enthalten. Die Intention des Autoverleihs enthält jedoch zusätzlich Objekte, die nicht in der Intention des Autohauses enthalten sind. In diesem Fall spricht man dann von einer Subsumption (vgl. [44, S. 224 f.]). Das kann mit einer Teilmengenbeziehung aus der Mengenlehre verglichen werden. Die Intention des Autohauses ist eine Teilmenge der Intention des Autoverleihs. Einen allgemeineren Fall stellt die Überlappung dar, bei der gewisse Punkte übereinstimmen und anderen sich unterscheiden. In seltenen Fällen kann es zu einer Disjunktheit der Konzepte kommen. Das bedeutet, dass die Intentionen der Begriffe keine Gemeinsamkeiten aufweisen (siehe Beispiel Homonym , Bank'). Ein anderes Problem der Semantik ist eine mögliche Inkonsistenz der Nutzdaten, welche zu Widersprüchen führt. Widersprüche können durch Nutzdaten entstehen, die zu unterschiedlichen Zeitpunkten erhoben wurden und ihre Gültigkeit bereits verloren haben (vgl. [44, S. 225]). So ist es in manchen Situationen auch interessant, den Nutzdaten eine zeitliche Dimension hinzuzufügen, wenn dies möglich ist.

Um die strukturellen und semantischen Probleme der Datenintegration zu lösen, bedarf es, neben der Bereitstellung durch einen Wrapper, zusätzliche Arbeit. Die Konflikte müssen erstmal erkannt und danach aufgelöst werden. Es hat sich gezeigt, dass die Ontologien aus dem Bereich der Semantik ein gutes Mittel sind, um solche Konflikte zu erkennen und die Daten aus semantischer Sicht zu beschreiben. Ontologien bieten die Möglichkeit, den Nutzdaten Metadaten hinzuzufügen und dadurch deren Inhalt besser zu beschreiben. Damit lassen sich dann allgemeine Modelle erstellen, auf denen Zusammenhänge zwischen Nutzdaten abgebildet werden können (vgl. [44, S.216]).

2.2 Ontologien und Semantik

Ontologien sind ein Teilgebiet der Philosophie, die sich mit der Zusammensetzung des Seins beschäftigt (vgl. [6, S.4 ff]). Semantik ist ein Teilgebiet der Philosophie, welches sich mit der Bedeutung von Zeichen und Symbolen beschäftigt (vgl. [13]). In der Informatik werden diese zwei Gebiete zusammengeführt und bilden die Grundlage für die semantische Datenintegration. Dabei geht es darum, die Bedeutung von Daten zu verstehen und zu verarbeiten. Dafür werden die eigentlichen Nutzdaten mit zusätzlichen Metadaten erweitert, welche die Nutzdaten beschreiben.

In der Philosophie ist eine Ontologie eine Theorie über das Sein. Sie beschäftigt sich mit der Wirklichkeit und der Struktur des Existierenden (vgl. [19, S.199]). Dabei geht es um die Ordnung von Objekten und Konzepten. Man versucht sie damit in Kategorien einzuteilen (vgl. [6, S.5]). Eine Kategorie kann dabei auf verschiedene Arten definiert

werden. Entweder durch die Aufzählung aller Elemente, die zu der Kategorie gehören oder durch beschreibende Eigenschaften. Diese beschreibenden Eigenschaften werden Differntiae genannt und bilden eine Grundlage zur Entscheidung, ob ein Element zu einer Kategorie gehört oder nicht (vgl. [44, S.14 f.]). Eine Ontologie ist ein Netzwerk von Kategorien, die in Beziehung zueinander stehen.

Die Philosophie versucht, umfassende Kategoriensysteme zu entwickeln, die allen Objekten eine Kategorie zuordnet. In der Informatik hingegen ist der Nutzen von Ontologien anwendungsbezogen. Es soll nicht die ganze Welt, sondern nur ein Ausschnitt beschrieben werden, der für die Anwendung relevant ist (vgl. [44, S.11 f.]). Da mit Ontologien Konzepte und Begriffe in Zusammenhang gesetzt werden können, eignet es sich gut um bei semantischen Problemen in der Datenintegration zu unterstützen (vgl. [44, S.215]). Sie werden dort als ein neutrales Datenmodell genutzt (vgl. [44, S.225]). Dadurch können die unterschiedlichen Bedeutungen auf die einzelnen tatsächlich genutzten Begriffe in den Domänen-Daten abgebildet und eine einheitliche Sprache definiert werden. Dadurch lässt sich besser über die Domänen-Daten kommunizieren und sie können von verschiedenen Parteien genutzt werden (vgl. [33, S.1]). Mit Ontologien lassen sich auch Einschränkungen definieren. Durch Interferenz können eventuelle Konflikte frühzeitig erkannt werden (vgl. [44, S.232 f.]). Da in der Informatik Maschinen diese Ontologien lesen und verstehen sollen, muss eine formale Sprache definiert werden. Dafür wurde die Web Ontology Language (OWL) entwickelt (vgl. [44, S.97 ff.]).

Die OWL Spezifikation wurde im Februar 2004 vom World Wide Web Consortium (W3C) veröffentlicht. Im Jahre 2009 wurde OWL2 veröffentlicht, welche OWL um neue Funktionalitäten erweitert. OWL2 ist rückwärts kompatibel, was bedeutet, dass alle Ontologien, die mit OWL definiert wurden, auch in OWL2 gültig sind (vgl. [46]). Es handelt sich um eine formale Sprache, die genutzt wird, um Ontologien zu beschreiben. Sie basiert auf RDF und Resource Description Framework Schema (RDFS) (vgl. [27]). Beides sind Sprachen zur Beschreibung von Nutzdaten. RDF wird genutzt, um Aussagen über Ressourcen zu formalisieren und für Maschinen interpretierbar zu machen. Diese Aussagen werden, in Form von Tripeln, bestehend aus Subjekt, Prädikat und Objekt gespeichert. Das Subjekt ist die Ressource, die beschrieben werden soll. Ein Prädikat beschreibt die Eigenschaft des Subjekts, die definiert werden soll. Das Objekt ist der Wert, der dem Prädikat zugeordnet werden soll. Dabei kann es sich um eine andere Ressource oder ein Literal, also einen konkreten Wert, handeln. Die Tupel die in einem RDF Dokument definiert werden, bilden zusammen einen gerichteten Graphen. Das Subjekt und Objekt sind zwei Knoten, das Prädikat ist die Kante, die die beiden Knoten verbindet. Dieser Graph bildet das eigentliche RDF Modell, welches die Ressourcen beschreibt. Um die unterschiedlichen Ressourcen ansprechen und unterscheiden zu können, werden Uniform Resource Identifier (URI) genutzt. Eine URI ist ein eindeutiger Bezeichner für eine physische oder abstrakte Ressource, die von Website-Adressen inspiriert wurde (vgl. [5]). Sowohl Subjekt, Prädikat und Objekt werden dabei mit ihrer URI angegeben. Bestimmten URIs werden in der RDF Spezifikation spezifische Bedeutungen zugewiesen. Für die Prädikate kann eine eigene Terminologie definiert werden, um Ressourcen zu beschreiben. Dafür muss jedem

eigen definierten Prädikat eine URI zugeordnet werden. Mithilfe des RDF Modell können Maschinen die Ressourcen besser interpretieren und Abfragen auf ihnen ausführen (vgl. [26]). RDF kann somit Ressourcen beschreiben, allerdings fehlen zur Beschreibung von kompletten Ontologien noch andere passende Sprachkonstrukte. OWL definiert diese Sprachkonstrukte und kann somit, auch in Verbindung mit RDF, Ontologien beschreiben (vgl. [27]).

Mit OWL können Konzepte und Relationen definiert werden. Konzepte lassen sich auf verschiedene Weisen definieren. Man kann sie durch ihre Eigenschaften oder durch ihre Beziehungen zu anderen Konzepten abgrenzen. Ein Konzept kann durch seinen Oberbegriff definiert werden, aber auch durch sein Gegenteil oder eine Disjunktion aus anderen Konzepten. Ein Beispiel: Ein Konzept *Weiblich* wird erstellt und diesem Objekte zugewiesen, welche als weiblich definiert sind. Als Komplement wird das Konzept *Männlich* definiert. So kann kein Objekt, welches dem Konzept *Weiblich* angehört, dem Konzept *Männlich* angehören. Relationen können zwischen zwei oder mehreren Konzepten sowie zwischen Konzepten und Datentypen definiert werden. Relationen besitzen immer einen Werte- und Zielbereich, wodurch eingeschränkt werden kann, welche Objekte Teil einer Relation sein können. Werden Relationen zwischen Konzepten erstellt, werden sie als Objekt-Eigenschaften (engl. Object Properties) bezeichnet. Eine mögliche Objekt-Eigenschaft wäre *hat_Vater*. Diese Relation beschreibt die Beziehung zwischen einer Person und ihrem Vater. Es ist möglich, diese Relation durch weitere Eigenschaften wie Funktionalität, Reflexivität, Symmetrie und Transitivität, einzuschränken. Beispielsweise ist es nicht sinnvoll, die Relation *hat_Vater* als transitiv zu betrachten. Der Vater des Vaters einer Person ist nämlich der Großvater und nicht der Vater. Eine andere Objekt-Eigenschaft, wie *ist_Vorfahre*, kann hingegen durchaus transitiv sein. Der Vorfahre eines Vorfahren einer Person ist ebenfalls ein Vorfahre. Objekt-Eigenschaften können auch in Relation mit anderen Objekt Eigenschaften gesetzt werden. Es ist mögliche eine Teilmengenbeziehung zwischen zwei Eigenschaften zu definieren. Die Relation *hat_Vater* kann als eine Teilmenge der Relation *hat_Elternteil* definiert werden. Neben den Objekt-Eigenschaften gibt es noch Daten-Eigenschaften (engl. Data Properties). Daten-Eigenschaften beschreiben die Relation zwischen einem Konzept und einem Datentyp. Ein Beispiel für eine Daten-Eigenschaft ist *hat_Alter*. Diese Relation beschreibt die Beziehung zwischen einer Person und ihrem Alter, einer Zahl. Konzepte und Relationen können durch Einschränkungen genauer beschrieben werden. Mit einer Werte-Einschränkung kann definiert werden, dass für eine bestimmte Menge von Objekten eine spezifische Eigenschaft vorhanden sein muss. Diese Werte-Einschränkung ist Ähnlich zu den All- und Existenzquantoren in der Prädikatenlogik (vgl. [21]). Diese Quantoren geben an, dass eine bestimmte Aussage für alle oder mindestens ein Objekt aus einer Menge gültig ist (vgl. [10, S.28]). Mit einer Kardinalitäts-Einschränkung kann man definieren, wie viele Instanzen einer Eigenschaft ein Objekt besitzen darf. Damit kann die Existenz so einer Eigenschaft erzwungen, verboten oder eingeschränkt werden. Bei der Eigenschaft *hat_Elternteil* kann es sinnvoll sein, die Eigenschaft mit einer maximalen Kardinalität von zwei zu beschränken. Dann kann jedes Objekt maximal zwei Elternteile besitzen (vgl. [21]).

Das Erstellen von Ontologien wird als iterativer Prozess beschrieben (vgl. [33, 44]). Ähn-

lich wie in der Software Entwicklung, gibt es auch bei der Erstellung von Ontologien einiges zu beachten. Es ist notwendig, vorab eine Anforderungsanalyse durchzuführen, die definiert, welchen Bereich und welche Details die Ontologie abbilden soll. Abschließend müssen wichtige Begriffe aus der Domäne extrahiert und entschieden werden, welche Konzepte und Relationen modelliert werden sollen. Die Begriffe müssen dahingehend bewertet werden, ob sie für die Domäne relevant und für eine Modellierung geeignet sind. Mögliche Kriterien sind: Intuitivität, Erweiterbarkeit und Pflegbarkeit (vgl. [33, S.5]). Genauere Entwurfsmuster, wie bei der Software-Entwicklung, gibt es beim Erstellen von Ontologien nicht. Auch wenn bereits einige Richtlinien und typische Muster gefunden wurden, die beim Erstellen von Ontologien helfen. Das Erstellen von Ontologien ist ein zeitaufwendiger Prozess, der viel Wissen über die Domäne, Ontologien und OWL erfordert. Mittlerweile gibt es verschiedene Tools und Editoren, die bei der Erstellung von Ontologien unterstützen und eventuelle Konflikte erkennen können (vgl. [44, S.160 ff.]).

Diese Arbeit verwendet ebenfalls den Begriff Ontologie. Dabei ist der Begriff an verschiedenen Stellen unterschiedlich zu verstehen. Im Zusammenhang mit dem Tool SDI wird der Begriff Ontologie genutzt, um eine Menge von Klassen zu beschreiben, die ein logisches, objektorientiertes Modell über einer Domäne darstellen. In diesem Zusammenhang ist der Begriff nicht als absolut zu sehen, wie die eben erklärte formale Definition von Ontologien mit OWL. Bei Unklarheiten wird der Begriff an entsprechender Stelle in der Arbeit genauer differenziert.

2.3 SDI

SDI ist eine Applikation von Insights Hub (früher Mindsphere). Diese App ermöglicht es, Datenquellen zu vereinheitlichen und miteinander zu verknüpfen. Insights Hub ist eine cloud-basierte IOT-Umgebung, die von Siemens bereitgestellt wird. Sie bietet verschiedene Lösungen für die Verwaltung, Sammlung, Analyse und Visualisierung von Daten und Maschinen. Eine Maschine kann mit Insights Hub verbunden werden, um verschieden Daten zu sammeln und sie über eine Cloud zur gemeinsamen Nutzung verfügbar zu machen. Dazu zählen u. a. Prozessdaten, Maschinendaten und Metadaten. Diese Daten können in anderen, von Insights Hub angebotenen, Applikationen genutzt werden. So können Automatisierungen auf Basis von NodeRed erstellt werden, aber auch Visualisierungen mittels einer Tableau ähnlichen Oberfläche. NodeRed ist ein Low-Code Ansatz zur Erstellung einfacher Hausautomatisierungen. Dort können vorgefertigte Funktionen in einer grafischen Oberfläche nacheinander geschaltet werden, um eine gewünschte Aufgabe zu automatisieren (vgl. [34]). Kenntnisse über eine Programmiersprache sind nicht notwendig. Tableau ist ein Framework zur Visualisierung von Unternehmensdaten ([47]). In Zusammenarbeit mit Siemens können auch weitere, nutzerspezifische Applikationen angeboten werden. SDI ist eine dieser Applikationen und kann für die strukturelle und syntaktischen Integration von Daten genutzt werden. Mit SDI können Datenquellen registriert und mithilfe von Ontologien verknüpft werden. Das Erstellen und Ausführen von Abfragen ermöglicht es auf die Daten zuzugreifen.

Damit die Daten mit SDI bereitgestellt werden können, müssen diese vorher registriert werden. Dazu können Datenregister, in der App als Data Registry bezeichnet, erstellt werden. Im Bild 2.1 ist das Menü zu sehen, mit welchem Datenregister verwaltet und einzeln aufgelistet werden. Im Bild ist zu erkennen, dass Datenregister einen sogenannten Data Tag und einen Source Namen besitzen. In der Übersicht werden außerdem Datentyp und Datenquelle angezeigt. Zum jetzigen Zeitpunkt unterstützt SDI JSON-, CSV-, XML- und Parquet Dateien. Datenregistern können jederzeit Daten hinzugefügt oder gelöscht werden. Werden Daten zu einem Datenregister hinzugefügt, wandelt SDI diese in ein relationales Datenmodell um. Das dabei entstehende Datenschema kann in der Applikation eingesehen und in Abfragen verwendet werden. SDI stellt also Daten in einem einheitlichen Format zur Verfügung, wodurch syntaktische Konflikte gelöst werden.

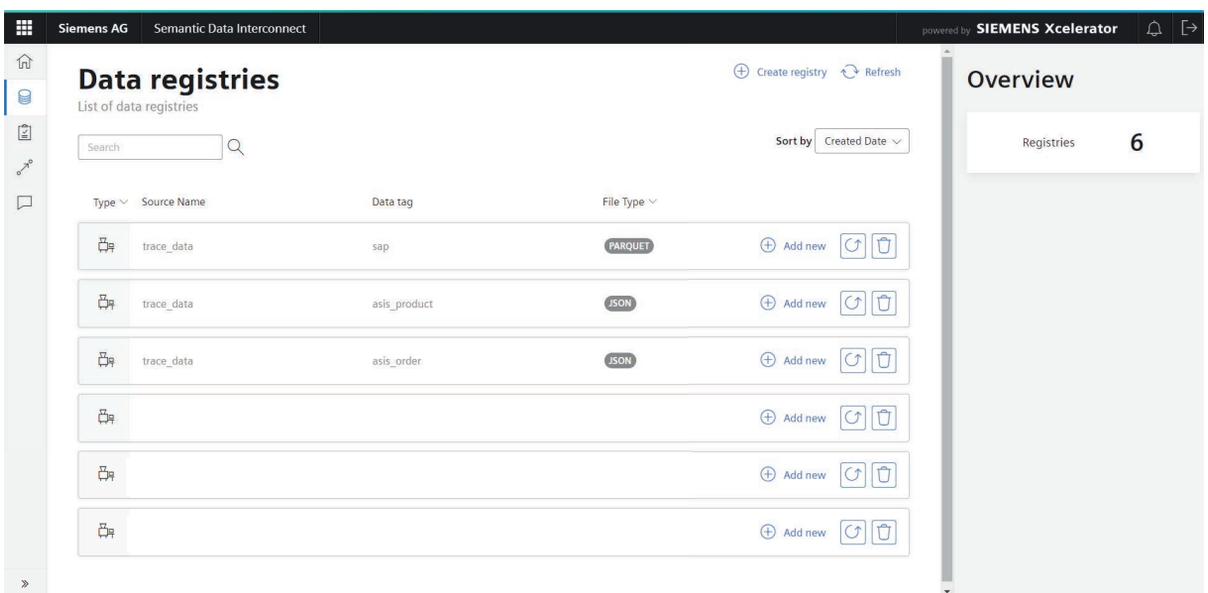

Abbildung 2.1: Data Registry Menü

Zur Erstellung eines Registers öffnet sich das in Bild 2.2 dargestellte Menü. In diesem Menü kann dem Datenregister einen Source Namen sowie ein Data Tag gegeben werden. Das Feld Source Name bezieht sich auf den Namen der Herkunft der Datenquelle. Der Source Name kann wie ein Namespace genutzt werden, um Datenquellen besser zu organisieren. Der Data Tag ist eine weitere Spezifizierung der Datenquelle. Source Name und Data Tag bilden zusammen den Bezeichner, mit dem die Datenquelle später in Abfragen angesprochen wird. Dieser Bezeichner setzt sich folgendermaßen zusammen: <*Source name*>_<*Data tag*>. Direkt darunter wird die Strategie ausgewählt, die beim Hochladen der Daten angewendet werden soll. Die Strategie wird in der App als Upload Strategy bezeichnet. SDI bietet zwei unterschiedliche Strategien an. Die erste ist die Erweiterung (in der App Append Strategy). Werden neue Daten zur Datenquelle hinzugefügt, werden die neuen Daten an die bereits Vorhandenen angehangen. Besitzen die neuen Daten ein anderes Schema wie die zuvor Verwendeten, wird das Schema in SDI erweitert. Die zweite Möglichkeit ist das Ersetzten (in der App Replace Strategy). Werden neue Daten zur Datenquelle hinzugefügt, werden

die Alten gelöscht und durch die Neuen ersetzt. Das Schema wird bei jeder neuen Datenquelle neu erstellt. Um welches Datenformat es sich bei der Datenquelle handelt, muss ebenfalls angegeben werden. Zusätzlich zum Datentyp ist es auch möglich, ein Muster anzugeben. Dieses Muster wird genutzt, um nur bestimmte Dateien in SDI aufzunehmen. Wird eine Datei hochgeladen, wird der Name der Datei mit dem Muster verglichen. Stimmt der Name mit dem Muster überein, wird die Datei in SDI aufgenommen. Mit den restlichen Feldern lassen sich Metadaten hinzufügen oder der Pfad zur Datenquelle angeben. Die Datenquelle kann auf einem weiteren Weg bereitgestellt werden, welcher zu einem späteren Zeitpunkt beschrieben wird. Sobald alle Informationen eingetragen sind und das Datenregister die gewünschten Konfigurationen hat, wird das Datenregister erstellt und bekommt eine ID zugeteilt. Anschließend muss das Datenregister mit Daten befüllt werden. Dazu wird es mit einem Ordern im Integrated Data Lake (IDL) verknüpft.

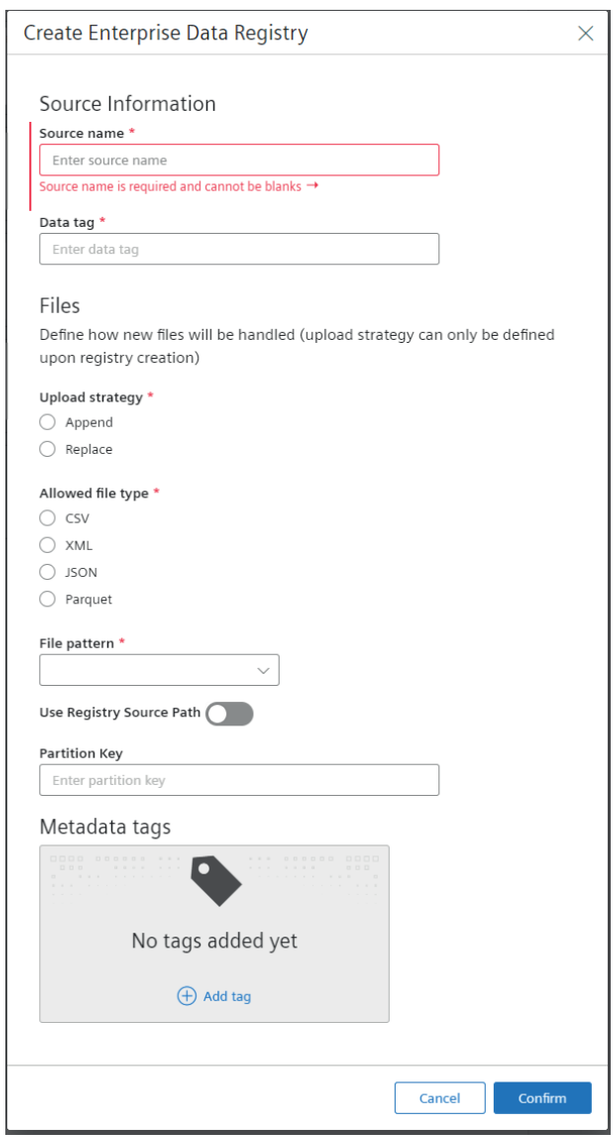

Abbildung 2.2: Data Registry Hinzufügen

IDL ist ein weiterer Service von Insights Hub. Dieser Service bietet Möglichkeiten zur Nutzung und Verwaltung eines Data Lake. In einem Date Lake werden Daten im Rohformat gespeichert. Diese liegen dort so lange, bis sie von einem Prozess abgeholt und weiterverarbeitet werden. Ein Data Lake muss skalierbar sein, unterschiedliche Datenformate unterstützen und eine einfache Möglichkeit bieten, Daten zu speichern und zu verwalten (vgl. [29]). Im IDL werden die Daten als Objekte in einer Verzeichnisstruktur gespeichert, wie es von Dateiverwaltungssystemen von Computern bekannt ist. Ein Ordner kann als Ausgangspunkt für Daten genutzt werden, die in SDI verarbeitet werden sollen. Damit SDI diese Nutzdaten finden kann, muss im IDL ein Ordner mit dem Namen *sdi* angelegt werden. Unterordner, die in diesem Ordner liegen, können als Datenquellen registriert werden. Um einen Ordner mit einer Datenquelle zu verknüpfen, werden Metadaten genutzt. In IDL lassen sich Metadaten für Ordner und Dateien hinzufügen. Der gewünschte Ordner erhält ein Metadaten-Tag, der die ID des zugehörigen Datenregisters enthält. Werden Dateien, des im Datenregister angegebenen Typs, in diesen Ordner hochgeladen, werden sie von SDI verarbeitet. Bei der Verarbeitung wird ein Schema erstellt und die Nutzdaten werden, entsprechend der gewählten Strategie, in SDI hinzugefügt. Das dabei entstandene Schema beschreibt den Aufbau der zugrundeliegenden Datenquelle. Die Nutzdaten müssen dafür in eine relationale Form gebracht werden. Dazu werden auch eventuelle verschachtelte Strukturen aufgelöst. Die Nutzdaten mit dem dazugehörigen Schema können weiterverarbeitet werden.

Die Verknüpfungen, die man mit SDI erstellen kann, werden über Ontologien realisiert. Mit diesen Ontologien lassen sich Klassen und Eigenschaften definieren, die auf den Datenquellen abgebildet werden. Dadurch wird es ermöglicht ein einheitliches Datenschema zur Verfügung zu stellen und die Daten auf eine Geschäftsebene zu heben. So können bestimmten Begrifflichkeiten oder Abkürzungen in den Datenquellen eindeutigen und verständlichen Bezeichnern zugeordnet werden. Dies erleichtert das Erstellen von Abfragen auf den Nutzdaten durch Nutzergruppen ohne spezifisches Hintergrundwissen. Eine SDI-Ontologie besitzt einen Namen und kann über SDI erstellt, verändert und gelöscht werden. Im folgenden Menü (Siehe Abbildung 2.3) werden die erstellten Ontologien aufgelistet und sie können verwaltet werden.

Sie können als OWL- oder JSON-Datei zur Verfügung gestellt werden. OWL und JSON sind Spezifikationen für Beschreibungssprachen. JSON ist eine allgemeine Spezifikation, die für die Beschreibung von Daten genutzt wird. OWL ist eine Spezifikation, die für die Beschreibung von Ontologien genutzt wird. Der Unterschied liegt hier rein im strukturellen Aufbau der Dateien, beide Spezifikationen können zur Beschreibung von Ontologien genutzt werden. Diese Dateien müssen ein bestimmtes Format aufweisen, damit die Ontologien von SDI richtig gelesen werden. Das Format bedingt die Definition der verschiedenen Schemata der Datenquellen, sowie Klassen, deren Eigenschaften und die Abbildungen zwischen den Schemata und den Klassen.

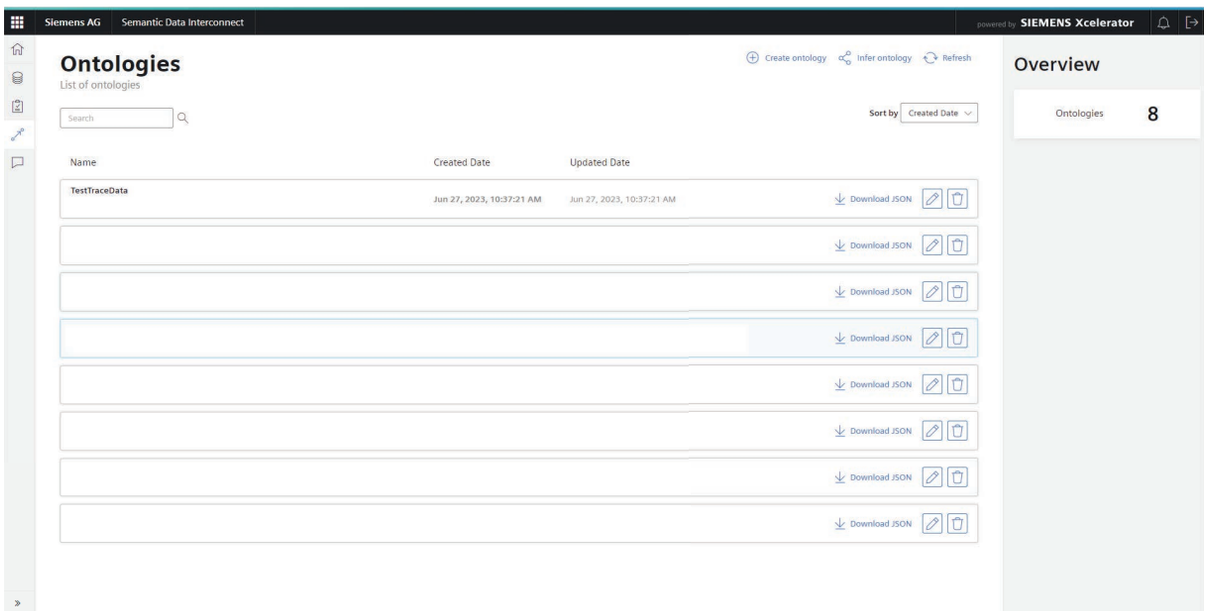

Abbildung 2.3: Ontologie Menü

SDI bietet zudem eine vereinfachte Möglichkeit, diese JSON-Datei zu erstellen. Dazu wird eine Benutzeroberfläche bereitgestellt, mit der SDI-Ontologien definiert werden können. Hierzu werden die Datenquellen-Schemata ausgewählt, mit welchen eine SDI-Ontologie erstellen werden soll. In Bild 2.4 ist abgebildet, wie Klassen erstellt und ihnen Eigenschaften zugeordnet werden. Eine Klasse besitzt einen eindeutigen Namen und kann mehrere Eigenschaften aufweisen. Diese Eigenschaften haben einen Namen, einen Datentyp, eine Abbildung und einen Abbildungsnamen. Die Abbildung gibt an, welche Spalte des Datenquellen-Schemas mit der Eigenschaft verknüpft werden soll. Es können mehrere Spalten angegeben werden, die in der Abbildung mit einem Komma getrennt werden. Dadurch können mehre Datenquellen miteinander verknüpft werden. Der Abbildungsname muss ein eindeutiger Bezeichner für die Verknüpfung sein. Der Datentyp einer Eigenschaft muss mit dem Datentyp der Spalte in dem Datenquellen-Schema übereinstimmen. Hat man alle Klassen definiert, erstellt SDI eine JSON-Datei, welche die eben definierte Ontologie enthält. Diese Datei muss abgespeichert werden und kann anschließend in SDI hochgeladen werden, um die SDI-Ontologie zu erstellen.

Eine weitere Funktionalität die SDI bietet, ist das Erstellen von Abfragen (in der App als Queries bezeichnet). Abfragen können genutzt werden, um Daten aus den Datenquellen zu extrahieren. Die Abfragen in SDI basieren auf Structured Query Language (SQL) und benutzen die gleiche Syntax. Einige Operationen, die in SQL möglich sind, werden nicht unterstützt. Darunter der * Operator, der alle Spalten einer Tabelle auswählt. Abfragen können entweder auf den Datenquellen-Schemata oder auf den SDI-Ontologien erstellt werden. Werden Abfragen auf den Datenquellen-Schemata ausgeführt, werden die einzelnen Schemata mit ihrem Bezeichner der Form <*Source Name*>_<*Data Tag*> angesprochen. Soll auf die einzelnen Spalten eines Datenquellen-Schemas zugegriffen werden, wird die Punktnotation verwendet. Auf den Schema Bezeichner folgt, getrennt durch einen Punkt (.), der gewünschte Spaltenname (<*Source*

Name>_<*Data Tag*>.<*Spaltenname*>). Werden Abfragen auf SDI-Ontologien ausgeführt, werden die Klassen und Eigenschaften der SDI-Ontologien angesprochen. Diese werden, ebenfalls mit der Punktnotation angegeben. Sowohl die Klassen als auch die verschiedenen Schemata können verknüpft, gefiltert und gruppiert werden.

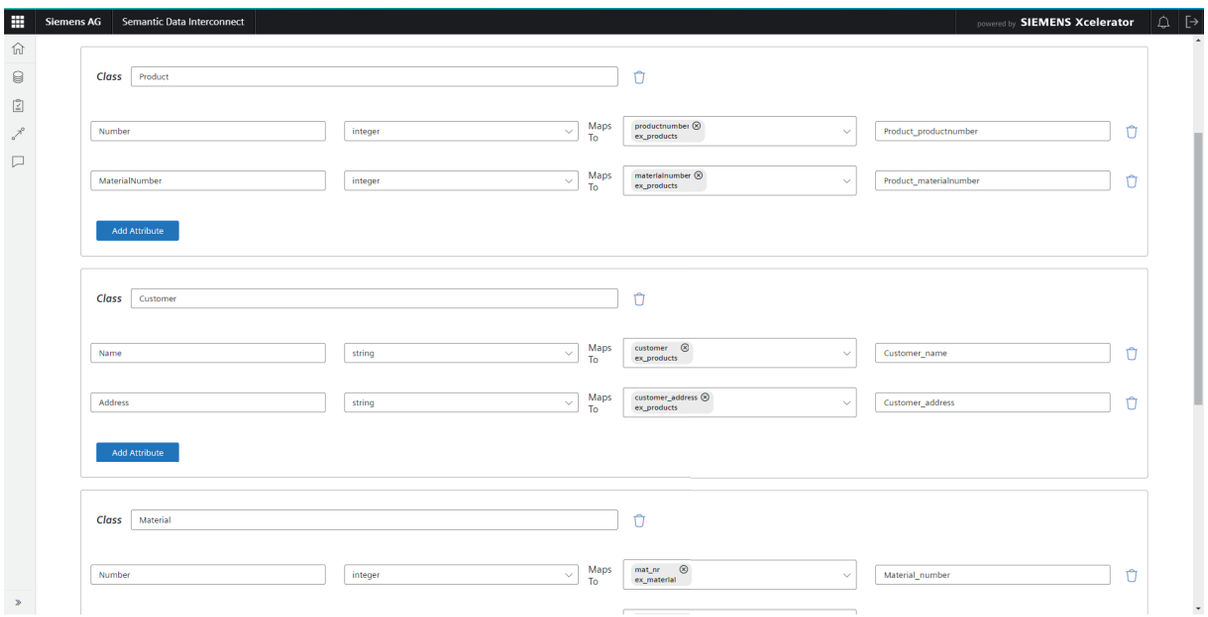

Abbildung 2.4: Eine Ontologie Erstellen

Im Menü aus Abbildung 2.5 werden die Abfragen erstellt. Es kann eingestellt werden, ob es sich um eine physische Abfrage (Physical Query) handelt, welche direkt auf den Datenquellen-Schemata ausgeführt wird. Oder aber, ob es sich um eine Geschäftsabfrage (Business Query) handelt, die auf den SDI-Ontologien ausgeführt wird. Die Abfrage benötigt außerdem einen Namen, der nicht eindeutig sein muss. Zusätzlich kann eine Beschreibung hinzugefügt werden. Die Abfrage selbst wird in einem Textfeld eingegeben. Dies ist im Menü unter der Überschrift Statement zu finden. Zusätzlich kann eingestellt werden, ob eine dynamische Abfrage erstellt werden soll. Dynamische Abfragen erlauben die Parametrisierung der Abfrage und die persönliche Anpassung der Ergebnistabelle. Die Parameter werden in der Abfrage mit einer besonderen Syntax markiert und können bei einem Aufruf der Abfrage angegeben werden. Zusätzlich ist es möglich, Aliase mitzugeben, die den Ergebnisspalten einen anderen Namen geben. Dynamische Abfragen können wiederholt, mit unterschiedlichen Parametern, ausgeführt werden. Das Gegenstück, die sogenannte statische Abfrage, wird einmalig beim Erstellen der Abfrage ausgeführt und kann nicht mehr verändert werden. Werden den Datenquellen neue Daten hinzugefügt, so wird die statische Abfrage neu ausgeführt und die Ergebnisse werden aktualisiert. Ganz unten im Menü lässt sich einstellen, wie das Ergebnis der Abfrage gespeichert werden soll. Wird hier nichts ausgewählt, wird das Ergebnis als HTTP Response zurückgegeben. Das bedeutet, das Ergebnis wird gespeichert und bei einem Abruf der Abfrage als HTML Seite zurückgegeben. So kann das Ergebnis über eine API direkt abgerufen und weiterverarbeitet werden. Die andere Möglichkeit ist das Speichern des Ergebnisses in einer Datei. Hierzu wird im

Menü *Save result as a file* ausgewählt und ein Pfad im IDL angegeben, indem die Datei gespeichert werden soll. Die Datei wird in einem Ordner mit dem Zeitstempel der Ausführung als Parquet Datei gespeichert und kann dann weiterverarbeitet werden.

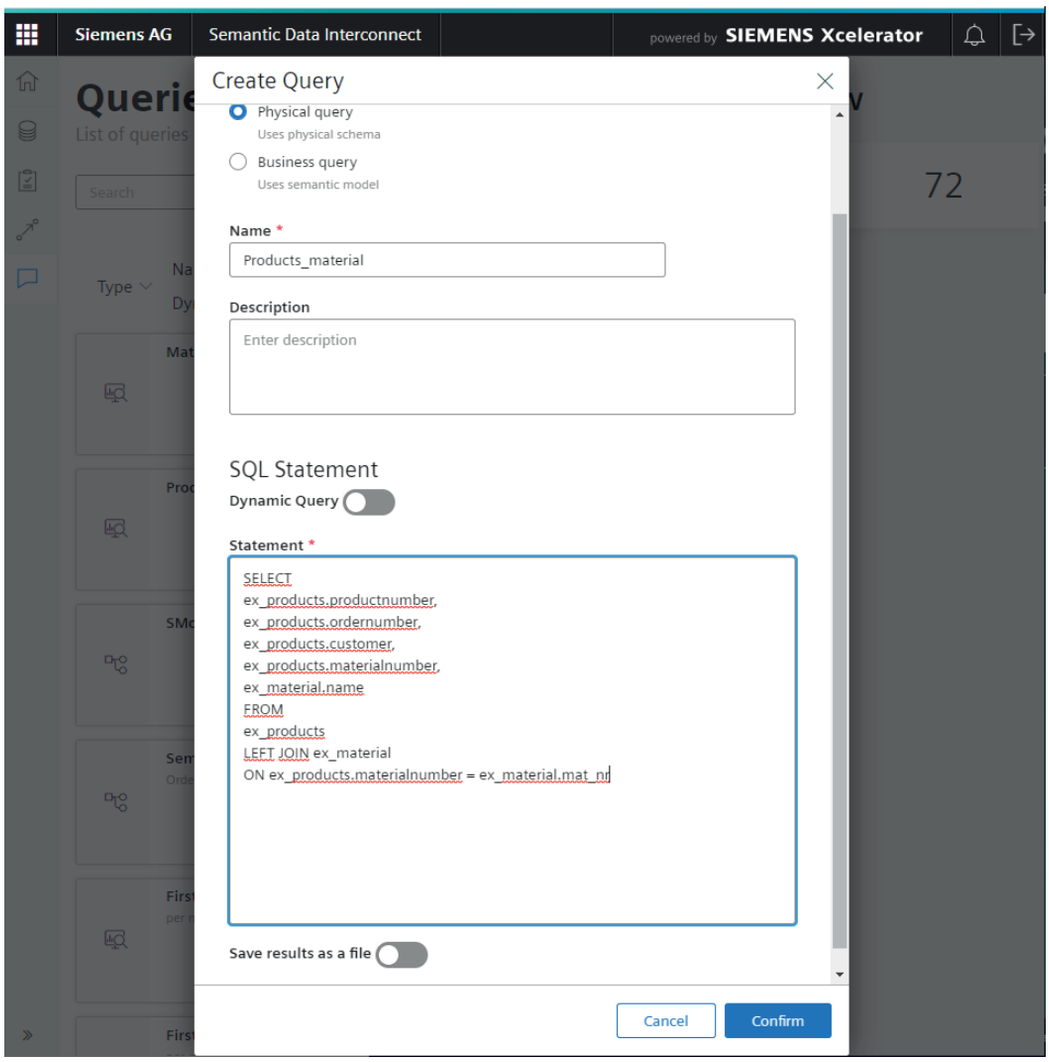

Abbildung 2.5: Eine Abfrage erstellen

Mit SDI lassen sich verschiedene Datenquellen registrieren. Diese werden in ein einheitliches Format gebracht und können mit Abfragen durchsucht und bereitgestellt werden. SDI bietet somit Lösungen für die Datenintegration an. Durch die Schemata der Datenquellen wird ein einheitliches Datenmodell verwendet. Somit werden die syntaktischen Probleme der Datenintegration aufgelöst. Mit SDI-Ontologien können Datenquellen miteinander verknüpft und den Spalten neue Bezeichner vergeben werden. Dies hilft dabei, die strukturellen und zum Teil auch semantischen Herausforderungen zu lösen.

2.4 Anchor Modeling

Anchor Modeling ist eine Technik zur Datenmodellierung und wurde von Lars Rönnbäck entwickelt und veröffentlicht. Sie wird genutzt, um ein logisches Datenmodell zu erstellen. Damit ist es unabhängig von der physischen Datenbank. Dieses Modellierungssystem wurde entwickelt, damit Veränderungen in der Datendomäne leichter in das Modell eingebunden werden können (vgl. [40, S.1229]). In der Literatur gibt es keine Übersetzung für die im Anchor Modeling verwendeten Begriffe. Deshalb werden im Folgenden und im Rest dieser Arbeit die englischen Begriffe Anchor, Attribute, Knot und Tie verwendet.

Im Anchor Modeling gibt es verschiedene Komponenten, die genutzt werden können, um bestimmte Sachverhalte zu modellieren. Ein Anchor modelliert Entitäten und Domänenobjekte. Ein Anchor kann verschiedene Attributes besitzen, wobei ein Attribute eine Eigenschaft eines Anchors darstellt. Attributes haben einen bestimmten Datentyp, einige sind mit Knots verbunden. Knots sind Attributes, die immer einen Wert aus einem gewissen (oft kleinen) Wertebereich annehmen. Knots können für verschiedene Attributes genutzt werden. In Programmiersprachen sind Knots vergleichbar mit Enums, die eine Aufzählung von möglichen Werten darstellen und einen eigenen Typ definieren. Ein Tie ist eine Verbindung zwischen zwei oder mehreren Anchors. Dabei wird jedem Anchor eine gewisse Rolle in der Verbindung zugeteilt. Zwei Anchors können auch mehrfach in Verbindung stehen, wenn sie unterschiedliche Rollen einnehmen. Zusätzlich zu ihrer statischen Form können Attributes und Ties auch als historisiert oder temporal modelliert werden. Dadurch wird den Attributes und Ties eine zeitliche Dimension hinzugefügt, die beschreibt, zu welchem Zeitpunkt ein Wert des Attributes oder Ties gültig ist. Wie diese Komponenten grafisch repräsentiert werden, wird in Tabelle 2.3 gezeigt. Jede Komponente hat ein eigenes Symbol, das in einem Modell genutzt wird. Ein Anchor wird als ausgefülltes, rotes Viereck repräsentiert. Ein roter Kreis (nicht ausgefüllt) stellt ein Attribute dar. Ein Knot wird als nicht ausgefülltes, graues Quadrat mit abgerundeten Ecken dargestellt. Eine graue, ausgefüllte Raute wird verwendet um einen Tie zu repräsentieren. Zusätzlich haben Attributes und Ties noch eine Variante, die genutzt wird, wenn sie historisiert sind. Dann haben sie eine zusätzliche Umrandung um ihr Symbol (vgl. [40, S.1231 ff.])

Tabelle 2.3: Die verschiedenen Symbole die in einem Anchor Model genutzt werden.

Das Besondere am Anchor Modeling ist die Übertragung des logischen Modells in ein physisches Modell. Bei der Überführung wird darauf geachtet, dass das physische Modell in sechster Normalform ist (vgl. [38]). Eine Datenbank ist in sechster Normalform, wenn sie keine nicht trivialen Verbundabhängigkeiten (engl. Join-Dependencies) erfüllt. Eine Verbundabhängigkeit ist eine Menge aus Teilmengen der Spalten einer Tabelle. Die Verbundabhängigkeit ist erfüllt, wenn man aus den Teilmengen die Tabelle wiederherstellen kann. Eine Verbundabhängigkeit ist trivial, wenn sie eine Teilmenge mit allen Spalten der Tabelle enthält. Anders ausgedrückt ist eine Tabelle in sechster Normalform, wenn sie sich nicht weiter in kleinere Tabellen unterteilen lässt, ohne dabei Daten zu verlieren. Normalerweise ist es nicht von Bedeutung, eine Datenbank in sechster Normalform zu haben. Da Anchor Modeling aber mit historischen Daten arbeitet, ist die starke Normalisierung wichtig um Anomalien zu vermeiden. So kann jedes Attribute, unabhängig von anderen Attributes, historisiert werden (vgl. [8, S. 173 ff.]). Dazu wird jeder Anchor, Attribute, Tie und Knot in eine eigene Tabelle überführt (vgl. [40, S.1237 f.]). Somit existieren keine nicht-trivialen Verbundabhängigkeiten mehr, da alle Tabellen minimal sind. Die Tabellen bestehen meist aus einen künstlichen Primärschlüssel und einem oder mehreren Fremdschlüsseln (vgl. [40, S-1238]). Dadurch wird eine gewisse Unabhängigkeit zwischen den Komponenten erreicht, die es erleichtert, ein Schema zu erweitern. Jede Erweiterung enthält dabei das vorherige Schema in sich (vgl. [40, S.1248]). Dadurch, dass ein Eintrag nicht in einer Reihe, sondern auf mehrere Tabellen aufgeteilt ist, können Null Werte vermieden werden. Wenn ein Attribute zu einem Eintrag nicht existiert, dann existiert dieser Eintrag in der Attribute-Tabelle nicht und muss nicht mit einem Null-Wert dargestellt werden. Zudem können Daten dadurch einfacher aktualisiert werden. Sind bestimmte Daten erst zu einem späteren Zeitpunkt vorhanden, können sie in der entsprechenden Tabelle hinzugefügt werden, ohne dass dabei bereits existierende Tabelleneinträge verändert werden müssen (vgl. [40, S.1248]). Das sind nur einige Vorteile, welche durch die starke Normalisierung erreicht werden können. Weiter sind in Lars Rönnbäcks Publikation aufgelistet ([40]).

Neben den Vorteilen weist eine starke Normalisierung ebenso Nachteile auf. Aufgrund der Voraussetzung, dass keine nicht-trivialen Verbundabhängigkeiten existieren, sind die Tabellen sehr klein, wodurch sich die Anzahl der Tabellen erhöht. Einfache Abfragen, wie ein SELECT, müssen durch Verknüpfungen über viele Tabellen ausgeführt werden. Es können dafür Views genutzt werden, um die Verknüpfung nur einmalig definieren zu müssen (vgl. [40, S.1241]). Diese erlauben es die komplexen Abfragen mit einem Namen zu versehen und die Abfrage so zu abstrahieren. Trotzdem werden für die Abfrage Join-Operationen benötigt, welche komplex und langsam sein können. Durch Tabellen Eliminierung wird versucht, die Abfrage zu vereinfachen und nicht benötigte Tabellen aus einer Abfrage zu entfernen. So ist es möglich, dass die Abfragen, trotz der Vielzahl an Tabellen, schnell ausgeführt werden können (vgl. [40, S.1243]).

Damit die Vorteile des Anchor Models auch erreicht werden können, existieren einige Richtlinien für die Erstellung solcher Modelle. In den Richtlinien wird beschrieben, wann die einzelnen Komponenten genutzt, eine Historisierung verwendet und wie Verbindungen modelliert werden sollen. In Kapitel 3.3 werden diese Richtlinien genauer

betrachtet und der Aufbau des Schemas an diesen ausgerichtet.

Zum Erstellen von Anchor Models wird ein offizielles Online-Tool der Entwickler angeboten. Auf deren Website[1] lassen sich Modelle einfach erstellen und exportieren. Der Editor benutzt ebenfalls die in Tabelle 2.3 vorkommenden Symbole zur Darstellung.

[1] `www.anchormodelling.com/modeler/latest`

Kapitel 3

Konzeption des Datenmodells

Ein allgemeines Datenmodell für einen Industriestandort zu entwickeln, ist eine komplexe und umfangreiche Aufgabe. Deshalb wird in dieser Arbeit die Vorgehensweise am Beispiel eines Anwendungsfalles dargelegt. Die Umsetzung erfolgt zugeschnitten auf den spezifischen Anwendungsfall, die Überlegungen und Schlussfolgerungen können auf andere Anwendungsfälle übertragen werden. Das Ziel ist es, ein Datenschema zu erstellen, in dem die Nutzdaten aus unterschiedlichen Quellen integriert und als stabile Schnittstelle für Anwendungen genutzt werden kann. Für die Entwicklung des Datenschemas soll Anchor Modeling genutzt werden, da dieser Ansatz durch eine hohe Normalisierung eine stabile Schnittstelle bieten kann. Dieses Schema soll anschließend als SDI-Ontologie umgesetzt werden.

Im folgenden Kapitel wird zuerst der betrachtete Anwendungsfall beschrieben und Anforderungen an das Datenschema abgeleitet. Anschließend werden die bereitgestellten Daten analysiert und ausgewertet. Zum Schluss wird ein Datenschema entworfen, welches als Grundlage für die Implementierung dient.

3.1 Der Anwendungsfall

In den vergangenen Jahren ist die Komplexität von Versorgungsketten kontinuierlich gestiegen. Dadurch steigt das Risiko für Fehler und Unterbrechungen in den Versorgungsketten (vgl. [37, S.1]). Um in Krisensituation schnell reagieren zu können, ist es wichtig, die entstandenen Probleme zurückverfolgen zu können. Dies ermöglicht es, schneller Ursachen zu finden und Entscheidungen in Krisensituation zu treffen (vgl. [37, S.12]). Auch bei Siemens soll die Rückverfolgbarkeit von Produkten verbessert werden. Dazu werden Produktionsdaten gekennzeichnet und mit Genealogie-Daten verknüpft. Die Genealogie ist ein „*Forschungsgebiet, das sich mit der Herkunft und den Verwandtschaftsverhältnissen bestimmter Personen, [...] befasst*“ [12]. Es handelt sich also um Daten, welche die Herkunft eines Produktes beschreiben. Diese Daten sind bereits, durch ein anderes System, erfasst und stehen darüber zur Verfügung. Alleine sind diese Daten nicht ausreichend, da sie nur minimale Informationen enthalten. Um die Genealogie-Daten nutzen zu können, müssen diese mit anderen Daten verknüpft

werden. Diese Daten liegen in einem SAP System vor und beinhalten Auftrags- und Produktionsdaten. Das Ziel des Anwendungsfalles ist es, die Genealogie- und Produktionsdaten in einer Anwendung zu visualisieren und den Nutzern zur Verfügung zu stellen. Zusammengefasst liegen zwei unterschiedliche Datenquellen vor, die miteinander verknüpft werden und für eine Anwendung bereitgestellt werden sollen. Die Bereitstellung der Daten erfolgt über ein Datenschema, welches als Schnittstelle für die Anwendung dient. Dieses Datenschema sollte wiederverwendbar, stabil und einfach erweiterbar sein. Durch die Verwendung von Anchor Modeling kann ein stabiles und leicht erweiterbares Datenschema erstellt werden. Die Wiederverwendbarkeit des Schemas muss durch den zuständigen Data Engineer sichergestellt werden. Das erstellte Schema soll dann in SDI umgesetzt werden. Dazu können die SDI-Ontologien benutzt werden, um die Daten in das erstellte Schema zu übertragen. Über die Abfragen können die benötigten Daten extrahiert und für die Anwendung bereitgestellt werden.

3.2 Rohdatenanalyse

Für die Entwicklung eines Datenschemas ist es wichtig, die Daten zu kennen, die in dem Schema integriert werden sollen. Zur Gewinnung eines Überblicks wurden die Daten des Anwendungsfalles mittels deskriptiver Datenanalyse analysiert. Für die Analyse der Rohdaten wurde die Python Bibliothek Polars genutzt. Polars stellt verschiedene Klassen und Methoden zur Verfügung, um mit Tabellen und Spalten zu arbeiten. Andere Bibliotheken wie Pandas oder Arrows können ebenfalls genutzt werden. Polars bietet eine bessere Performance und ist für die Arbeit mit großen Datenmengen optimiert (vgl. [36]). Polars nutzt DataFrames, um Daten zu repräsentieren. DataFrames werden als Tabellen interpretiert, die aus Spalten und Zeilen bestehen. Die Spalten können unterschiedliche Datentypen aufweisen, die Zeilen sind immer identisch aufgebaut. In Polars wird eine Spalte als Series bezeichnet. Polars bietet verschiedene Methoden, um mit DataFrames und Series zu arbeiten und sie zu manipulieren.

Es gibt drei unterschiedliche Datenquellen, welche im Folgenden als Auftrag, Produkt und SAP bezeichnet werden. Die Daten liegen im Rohformat als JSON- und Parquet Dateien vor. JSON ist ein Format zur Beschreibung von Objekten aus der Programmiersprache Java-Script und wird häufig genutzt, um Daten zwischen Anwendungen auszutauschen (vgl. [23]). Die Auftrags- und Produkt Dateien sind JSON-Dateien. Sie beinhalten die Genealogie-Daten. Da die Daten im JSON-Format vorliegen, handelt es sich hierbei um semistrukturierte Daten. Parquet ist ein Dateiformat zur Speicherung von Daten in tabellarischer Form. Parquet ist ein spalten-orientiertes Format und wird häufig genutzt, um große Datenmengen zu speichern (vgl. [35]). Die SAP Dateien sind Parquet Dateien. Sie beinhalten Auftrags- und Produktionsdaten. Parquet Dateien beinhalten strukturierte Daten, da sie ihre Daten in einer tabellarischen Form speichern.

Um die Rohdaten zu analysieren, müssen diese in die Python Umgebung geladen werden. Mit Polars lassen sich die Rohdaten über eine einfache *read Funktion* in einen DataFrame laden. Nachdem die Rohdaten eingelesen sind, kann sich deren Aufbau

angeschaut werden. Die Auftrags- und Produkt Dateien besitzen dabei das in Tabelle 3.1 und Tabelle 3.2 aufgezeigte Schema.

Produkt	
ProductNumber	*str*
ProductionOrderNumber	*str*
CustomerOrderNumber	*str*
CustomerOrderPositionNumber	*str*
GenealogyData	*list[struct[4]]*
ReferencedProductionOrders	*list[struct[2]]*

Tabelle 3.1: Schema der Produkt-Datei

Auftrag	
ProductionOrderNumber	*str*
GenealogyData	*list[struct[4]]*

Tabelle 3.2: Schema der Auftrag Datei

Beide Dateien enthalten ähnliche Felder. Die Felder `ProductionOrderNumber` und `GenealogyData` sind in beiden Dateien vorhanden. Die Felder `ProductNumber`, `CustomerOrderNumber`, `CustomerOrderPositionNumber` und `ReferencedProductionOrders` sind nur in der Produkt-Datei vorhanden. Bei `GenealogyData` und `ReferencedProductionOrders` handelt es sich um Listen, die weitere JSON-Objekte enthalten. Der Aufbau dieser Datenstrukturen ist in Tabelle 3.3 und Tabelle 3.4 zu sehen.

Die Genealogie-Daten beinhalten die Felder `AssembledDateTime`, `MaterialNumber`, `GenealogyTraceType` und `TraceData`. Einträge des Typs `ReferencedProductionOrders` beinhalten die Felder `ProductionOrderNumber` und `DeliveryDate`. Die Struktur `ReferencedProductionOrders` wird verwendet, um auf andere Einträge der Genealogie-Daten zu verweisen.

GenealogyData	
AssembledDateTime	*str*
MaterialNumber	*str*
GenealogyTraceType	*int*
TraceData	*str*

Tabelle 3.3: Schema der GenealogyData

ReferencedProductionOrders	
ProductionOrderNumber	*str*
Delivery Date	*str*

Tabelle 3.4: Schema der ReferencedProductionOrder

Die Schemata, die SDI nutzt, arbeiten nicht mit verschachtelten Strukturen. Diese Strukturen werden von SDI aufgelöst und in einzelne Spalten aufgeteilt. Das Schema, wie es von SDI erstellt wird ist in Tabelle 3.5, am Beispiel der Auftrags-Daten, zu sehen. Es ist zu erkennen, dass die einzelnen Felder der Genealogie-Datenstruktur jeweils in eine eigene Spalte umgewandelt wurde. Um zu verstehen, wie SDI die verschachtelten Daten aufspaltet, wurde in Python eine Aufspaltung der Strukturen durchgeführt. Dazu wurden die Auftrags-Daten umgeformt und das Ergebnis mit den relationalen Daten, welche in SDI liegen, verglichen. Um die relationalen Daten aus SDI zu extrahieren, wurde eine statische Abfrage genutzt. Das Ergebnis dieser Abfrage enthält die von SDI umgeformten Daten.

Auftrag	
productionordernumber	*str*
genealogydata_materialnumber	*str*
genealogydata_assembleddatetime	*str*
genealogydata_genealogytracetype	*long*
genealogydata_tracedata	*str*

Tabelle 3.5: Schema der Auftrag Datei in SDI

Um diese Struktur zu erreichen, müssen folgende Umformungen durchgeführt werden: Zuerst muss die Liste aufgelöst werden. Dazu kann für jeden Eintrag in der Liste eine neue Zeile erstellt werden. Dadurch entstehen mehr Zeilen, als im originalen Datensatz und die Spalte `ProductionOrderNumber` wird, je nachdem wie viele Einträge in der Liste vorhanden sind, mehrfach dupliziert. Danach muss die Struktur aufgelöst werden, indem für jedes Feld in der Struktur eine neue Spalte erstellt wird. Die Daten weisen danach die gleiche Struktur vor, wie das von SDI erstellte Schema. Vergleicht und sortiert man beide Datensätze, sind diese identisch. Listing 3.2 zeigt die Umsetzung dieser Umformungen mit Polars in Python. Der daraus entstehende Datensatz `orders_ex` hat identische Struktur wie der Datensatz, der von SDI erstellt wurde. Der einzige Unterschied ist die Reihenfolge und die Namen der Spalten, welche in Polars einfach angepasst werden können. Analog werden diese Strukturen ebenfalls in den Produkt-Daten aufgelöst, nur das hier zwei Listen vorhanden sind.

Bei den SAP-Daten gibt es einen anderen Aufbau. Die Tabelle enthält keine Listen und keine Strukturen und besteht aus einzelnen Feldern mit einfachen Datentypen. Die Daten sind bereits in strukturierter Form vorhanden. Das Schema der SAP-Daten kann im Anhang A nachgeschlagen werden. In der Tabelle sind viele Spalten vorhanden,

die Abkürzungen als Namen nutzen. In allen drei Datenquellen werden meist Strings genutzt, um die Informationen zu repräsentieren.

```
      import polars as pl
      #Lesen der Auftrags Dateien und erstellen eines Polars DataFrames
  ‘orders‘

      orders_ex = orders.explode('GenealogyData')

      orders_ex = orders.with_columns([
        pl.col('ProductionOrderNumber'),
        pl.col('GenealogyData')
          .struct.field('AssembledDateTime')
          .alias('AssembledDateTime'),
        pl.col('GenealogyData')
          .struct.field('MaterialNumber')
          .alias('MaterialNumber'),
        pl.col('GenealogyData')
          .struct.field('GenealogyTraceType')
          .alias('GenealogyTraceType'),
        pl.col('GenealogyData')
          .struct.field('TraceData')
          .alias('TraceData')
      ])

```

Listing 3.1: Auflösung verschachtelter Strukturen der Auftrags-Daten

Nachdem der Aufbau der Rohdaten bekannt ist, wird im Weiteren der Inhalt dieser untersucht. Dabei wird auf verschiedene Dimensionen der Datenqualität eingegangen. Dazu zählen unter anderem die Vollständigkeit, die Genauigkeit und die Konsistenz der Daten (vgl. [31]). Eine hohe Datenqualität ist wichtig, um Daten effizient und korrekt nutzen zu können. Fehlende oder fehlerhafte Datensätze können zu falschen Ergebnissen führen und einem Unternehmen schaden. Sind die Daten inkonsistent, ist die Arbeit mit ihnen zeit- und kostenintensiv (vgl. [28]). Die Analyse wurde für jeden Datensatz einzeln durchgeführt. Durch den ähnlichen Aufbau der Produkt- und Auftrags-Daten konnte, bei der Analyse der beiden Datensätze, ähnlich vorgegangen werden. Aus diesem Grund werden die Ergebnisse für diese beiden Datenquellen zusammen vorgestellt. Bei Unterschieden in der Vorgehensweise wird dies im Text angegeben. Die Analyse der SAP-Daten wird separat vorgestellt, da diese eine andere Struktur haben.

Angefangen mit den Auftrags-Daten, wurde die Datenquellen auf Duplikate untersucht. Duplikate bringen verschiedene Nachteile mit sich, wenn sie in Daten vorhanden sind. Zum einen nehmen sie Speicherplatz weg, der für andere Daten genutzt werden könnte. Zum anderen können sie zu falschen Ergebnissen führen, wenn sie nicht erkannt werden (vgl. [28]). Die `GenealogyData`-Series besteht aus Listen, die nicht direkt auf Duplikate untersucht werden kann. In den Auftrags-Daten existiert eine weitere Spalte. Die Spalte `ProductionOrderNumber`, welche keine Duplikate enthält. Die Datenquelle

enthält keine ausgewiesene Schlüsselspalte, was in JSON nicht vorgesehen. Es existiert also kein dedizierter Schlüssel, der die Einträge eindeutig identifiziert. Nichtsdestotrotz könnte die Spalte `ProductionOrderNumber` als Schlüsselkandidat genutzt werden, da diese eindeutig ist. Diese Eindeutigkeit muss aber durch die Anwendung und durch den Anbieter der Datenquelle sichergestellt werden, da JSON solche Konstrukte nicht durchsetzt.

Werden die Auftrags-Daten in SDI geladen, verändern sie ihre Struktur, wie oben beschrieben. Daher wird die Anzahl der Duplikate in den umstrukturierten Daten untersucht. In den umstrukturierten Auftrags-Daten gibt es mehr Einträge als vorher. Zudem gibt es dadurch auch Duplikate. In den relationalen Auftrags-Daten lassen sich 9.878 Duplikate von 13.078 Einträgen finden. Dies entspricht 75,55 % der Auftrags-Daten. Betrachtet man die einzelnen Spalten, finden sich Duplikate in jeder Spalte. In den Spalten `ProductionOrderNumber`, `TraceType` und `MaterialNumber` sind die meisten Duplikate vorhanden. Die, vorher eindeutige Spalte `ProductionOrderNumber`, enthält nun 12.528 Duplikate (95,8 %). Die Anzahl der Duplikate in der Spalte `ProductionOrderNumber` ist abhängig von der Anzahl der Genealogie-Daten aus der Liste. In der Spalte `TraceType` sind 99,9 % Duplikate vorhanden. Das liegt daran, weil es nur wenige verschiedene Werte für `TraceType` gibt, weshalb sie sich oft wiederholen. Ein möglicher Grund für die hohe Anzahl an Duplikaten in `MaterialNumber` (99,7 %) ist denkbar, dass für viele Produkte dieselben Materialien verwendet werden oder auch, dass mit den Genealogie-Daten bisher nur bestimmte Materialien in einem kleinen Wertebereich nachverfolgt werden. Betrachtet man die Spalten der Genealogie-Daten, finden sich hier ebenfalls Duplikate. Es sind also schon komplette Genealogie-Daten doppelt vorhanden. Von den 550 Einträgen in den originalen Auftrags-Daten, besitzen 460 Einträge eine Liste aus Genealogie-Daten, die Duplikate enthalten. Durch die Duplikate in den relationalen Auftrags-Daten können diese nicht eindeutig identifiziert werden. Es gibt keine Menge von Spalten, die als Schlüssel genutzt werden könnte. Weder die Originalen, noch die relationalen Auftrags-Daten enthalten Null-Werte. Es ist in jeder Spalte für jeden Eintrag ein Wert vorhanden.

Als Nächstes werden die Produkt-Daten untersucht. Hier wird ähnlich vorgegangen, wie bei den Auftrags-Daten. Zuerst wird der Datensatz auf Duplikate untersucht. Hier finden sich ebenfalls zwei Spalten, die nicht direkt auf Duplikate untersucht werden können. Die Spalten `GenealogyData` und `ReferencedProductionOrders` sind Listen, die nicht direkt berücksichtigt werden. Berücksichtigt man bei der Untersuchung aber alle restlichen Spalten, finden sich auch hier keine Duplikate. Das bedeutet, dass mindestens eine der Spalten in den Produkt-Daten eindeutig ist. Untersucht man hingegen jede Spalte einzeln auf Duplikate, kann erkannt werden, dass es sich nur um eine Spalte handelt, die keine Duplikate enthält. Die Anzahl der Duplikate in den einzelnen Spalten ist Tabelle 3.6 zu entnehmen. Die hohe Anzahl der Duplikate von `CustomerOrderPositionNumber` lässt sich damit erklären, dass es sich um eine Positionsnummer innerhalb eines Auftrages handelt. Diese Nummern sind also nur pro Auftrag eindeutig und kommen somit häufig wiederholt in den Produkt-Daten vor. Auch hier gibt es keine dedizierte Schlüsselspalte. Die Spalte `ProductNumber` kann als Schlüsselkandidat genutzt werden, da diese eindeutig ist.

Spalte	Anzahl Duplikate	Duplikate in %
ProductNumber	11994	100 %
ProductionOrderNumber	8218	68,52 %
CustomerOrderNumber	9978	83,19 %
CustomerOrderPositionNumber	11947	99,61 %

Tabelle 3.6: Duplikate in den Spalten der Produkt-Daten (Einträge pro Spalte: 11994)

Da auch die Produkt-Daten verschachtelte Strukturen enthalten, müssen die Produkt-Daten umstrukturiert werden, sobald sie in SDI geladen werden. Bei einer Analyse der umgeformten Produkt-Daten fällt auf, dass weniger Duplikate vorhanden sind als in den umgeformten Auftrags-Daten. Bei Betrachtung des gesamten Datensatzes finden sich lediglich vier Duplikate. Es gibt auch hier keine möglichen Schlüsselkandidaten, die einen Eintrag eindeutig identifizieren. Die umstrukturierten Produkt-Daten enthalten allerdings, im Gegensatz zu den umstrukturierten Auftrags-Daten, Null-Werte. In der Spalte `ReferencedProductionOrders` sind 6069 Null-Werte vorhanden. Bei insgesamt 31210 Einträgen entspricht das 19,45 % der Daten. Die `ReferencedProductionOrders` verweist auf einen anderen Auftrag, falls dieser für das Nachverfolgen benötigt wird. Möglich ist auch, dass kein Auftrag existiert, auf den verwiesen werden muss. Daher ist die Existenz der Null-Werte in dieser Spalte nicht ungewöhnlich. Trotzdem muss dies kontrolliert werden, da es sich um Anomalien handeln kann und Null Werte gesondert behandelt werden müssen. Die Wahl eines alternativen Ansatzes wäre ebenfalls denkbar, sodass keine Null-Werte entstehen, wenn keine Referenz benötigt wird.

In beiden Datenquellen werden Datumswerte als Strings repräsentiert. Da es in JSON kein Datentyp für Datum und Zeit gibt, ist dies eine mögliche Lösung. Die Einträge haben alle dasselbe Format und können leicht umgewandelt werden, falls benötigt.

In den SAP-Daten gibt es keine Duplikate und auch keine Null-Werte. Die SAP-Daten besitzen einen künstlichen Schlüssel, der die Daten eindeutig identifiziert. Die Bezeichnung der Spaltennamen ist kryptisch. Es werden eine Vielzahl an Abkürzungen benutzt, aus denen sich nicht eindeutig der Inhalt der Spalte erschließen lässt.

Zusammenfassend lässt sich sagen, dass die Produkt- und Auftrags-Daten viele Dateneinträge enthalten. Das liegt hauptsächlich daran, wie die Rohdaten aufgebaut sind und wie die verschachtelten Datenstrukturen wieder aufgespalten werden. Vorhandene Duplikate mit unklarer Herkunft müssen identifiziert werden, um effizient die Rohdaten zu speichern und zu verarbeiten. Eine Möglichkeit ist es, die Strukturen in verschiedene Datenquellen aufzuteilen und sie mittels Fremdschlüsseln zu verknüpfen. Dies ist direkt aus den JSON-Daten möglich, sodass die eigentliche Datenquelle nicht verändert werden muss. Zur Realisierung bedarf es der Entwicklung eines separaten Transformators, welcher die entsprechend aufteilt, kennzeichnet und verknüpft. Eine Aufgabe, die den Umfang der vorliegenden Arbeit überschreitet und daher nicht in diesem Rahmen realisierbar ist. Mögliche Transformationen auf den Datensätzen werden in einem späteren Kapitel (4) genauer beschrieben.

3.3 Datenmodellierung

Das Ziel dieser Datenmodellierung ist ein Schema zu erstellen, welches von anderen Systemen genutzt werden kann, um einheitlich auf die Nutzdaten zuzugreifen. Dafür muss das Schema stabil sein. Es sollte also einfach erweiterbar sein, ohne dabei bestehende Anbindungen an das Schema zu verändern. Außerdem sollte es gut verständlich sein und eine Grundlage bieten, um mit den Nutzdaten zu arbeiten. Für die Datenmodellierung wird Anchor Modeling genutzt, damit das Schema leicht erweiterbar ist. Um die Vorteile, die Anchor Modeling bietet zu erhalten, wurde sich an die in [40, S. 1235 ff.] beschriebenen Richtlinien gehalten. In den Richtlinien wird beschrieben, wie Anchor und Attributes gewählt werden, wie Ties aufgebaut sein sollten, wann Knots genutzt werden und wann eine Attribute- oder Tie-Historisierung notwendig ist. Damit auf die zugrundeliegenden Nutzdaten zugegriffen werden kann, müssen die Entitäten aus der Domäne und den Datenquellen extrahiert und in das Schema übertragen werden. Dafür wurden die Ergebnisse der Datenanalyse sowie die Datenquellen selbst und auch die Anforderungen, die sich aus dem Anwendungsfall ergeben, genutzt. Für diese Arbeit wurde mit den Anwendern zusammengearbeitet, um die Beziehungen zwischen den Daten besser zu verstehen. Daraus hat sich folgendes Datenschema (Abbildung 3.1) ergeben. In den folgenden Abschnitten wird das Datenschema genauer beschrieben und erläutert, wie die Entitäten und Attributes ausgewählt wurden. Dazu wird auf passende Richtlinien oder andere Designmuster eingegangen, die gewählt wurden.

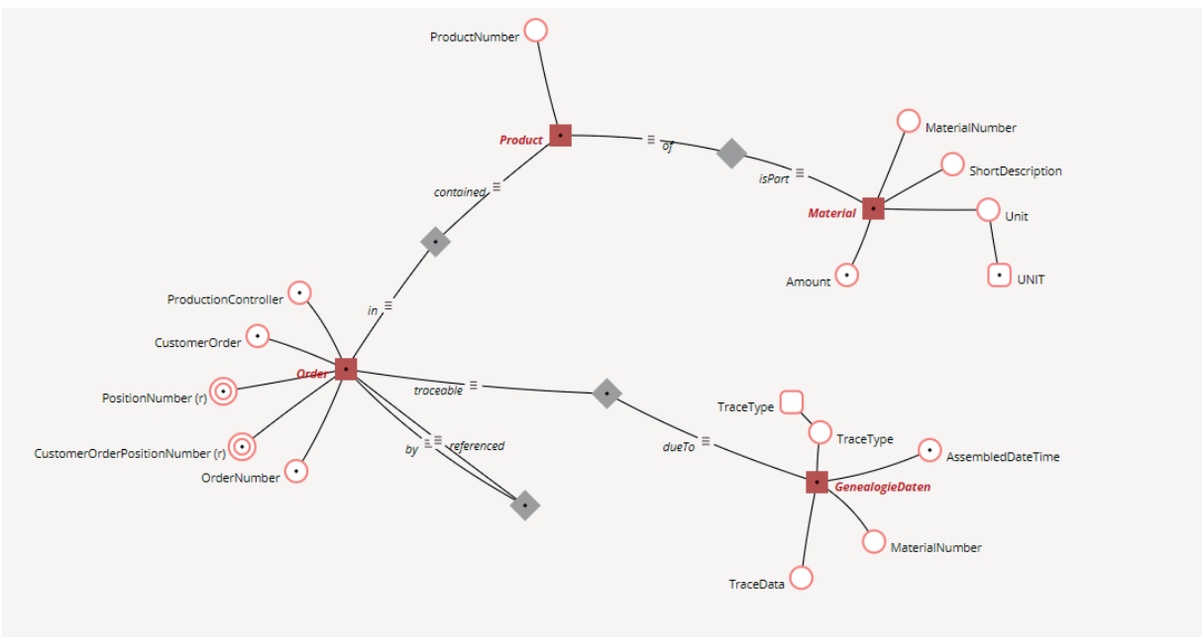

Abbildung 3.1: Das erstellte Schema

3.3.1 Anchors und Attributes

Anchors repräsentieren die Entitäten in einem Datenmodell. Entitäten können Objekte, Personen, Orte oder auch Konzepte sein. Eine bekannte Methode aus der Software-Entwicklung ist es, nach Substantiven in den Anforderungen und Anwendungsfällen zu suchen. Angepasst auf die Datenmodellierung, wurde nach Substantiven in den Datenquellen gesucht. Insbesondere in den Schemata der Datenquellen. Dort finden sich verschiedene Begriffe, wie z. B. `Product`, `Order`, `GenealogyData`, `Customer` oder auch `Material`. Aber auch weniger spezielle Begriffe, wie z. B. `Number`, `Position`, `Trace` oder `Date`. Schaut man die Datenquellen genauer an, erkennt man, dass diese zum Großteil aus den Genealogie-Daten bestehen. Bei den Genealogie-Daten handelt es sich um Strukturen, die aus einer Menge an Eigenschaften bestehen. Dieser Aufbau stellt eine gute Grundlage dar, um Genealogie-Daten als eine Entität zu modellieren. Die dazugehörenden Attributes sind die Eigenschaften in den Genealogie-Daten.

Da es im Anwendungsfall um die Nachverfolgbarkeit von Produkten geht, erscheint es sinnvoll, dass Produkt als eine Entität zu modellieren. Mögliche Attributes dafür sind in den Datenquellen vorkommende Felder, wie die `ProductNumber` oder die in den SAP-Daten vorhandene Maschinen-lesbare Fabrikbezeichnung (MLFB). Eventuell können auch weitere Attributes gefunden werden. Diese können erweitert werden, wenn sie benötigt werden. Ähnlich ist es beim Anchor `Auftrag`. Die verschiedenen Datensätze enthalten allesamt Auftragsnummern und ein Datensatz ist mit dem Begriff `Order` benannt. Bezüglich des Anwendungsfalles ist es sinnvoll, einen Auftrag zu modellieren. Die gelieferten Produkte können dann über die Auftragsnummer ausfindig gemacht werden. Mögliche Eigenschaften können aus den Rohdaten gezogen werden, wie z. B. die `OrderNumber` oder die `PositionNumber`. Position und Number waren zwei mögliche Kandidaten für Entitäten, die in den Rohdaten gefunden wurden. Nachdem die Rohdaten genauer betrachtet wurden, wurde festgestellt, dass es sich bei diesen beiden Begriffen eher um Attributes handelt. Weitere typische Eigenschaften eines Auftrags sind z. B. der dazugehörige Kunde und Prozessleiter. Diese Eigenschaften konnten in den Datenquellen gefunden werden und als Attribute des Auftrags modelliert werden.

Nachdem die Informationen zum Kunden im Auftrag gespeichert sind, stellt sich die Frage, ob die Kunden als Entität modelliert werden sollte. In den Rohdaten gibt es zwei verschiedenen Informationen zum Kunden, die Kundenauftragsnummer und die Kundenauftragsposition. Beide Eigenschaften beschreiben eher einen Auftrag als den Kunden selbst. Daher wurde entschieden, dass der Kunde nicht als Entität modelliert wird. Bedarf es zu einem späteren Zeitpunkt weitere Anforderungen, die eine Modellierung des Kunden erfordern, kann das Schema, ohne großen Aufwand, erweitert werden.

Die Entität `Materialien` lässt sich allerdings nicht eindeutig identifizieren. In den Genealogie-Daten kommt zwar eine Materialnummer vor, diese wurde aber als ein Attribute der Genealogie-Daten identifiziert. Aber das Vorhandensein dieser Eigenschaft deutet darauf hin, dass es sich dabei um eine Art Fremdschlüssel handelt, welcher auf ein Material verweist. In den SAP-Daten liegen weitere Informationen vor, welche die

Entität Material beschreiben, wie zḂ. eine Materialbeschreibung, die benötigte Menge und Einheit. Mit Absprache der Anwender konnte festgestellt werden, dass auch Materialien eine wichtige Entität ist, da diese mit der Rückverfolgung auch erkannt werden sollen.

Das Konzept Trace, welches als Möglichkeit für eine Entität genannt wurde, wird in diesem Schema nicht als Entität modelliert. Dieses Konzept wird im Schema (Abbildung 3.2) bereits durch die Genealogie-Daten abgebildet. Zudem ist das Wort Genealogie genauer definiert und beschreibt die Herkunft von Produkten. Trace lässt sich mit Spur übersetzen (z. B. Die Spur eines Autos), kann aber auch im Zusammenhang mit einer kleinen Menge (z. B. eine Spur von Salz) verwendet werden.

Mit diesen Entitäten entsteht das folgende Schema, welches in Abbildung 3.2 dargestellt ist. Es enthält die Anchor Order, Product, Material und GenealogyData. Die Attributes des Auftrags-Anchors sind OrderNumber, PositionNumber, CustomerOrder, CustomerOrderPositionNumber und ProductionController. Die Attributes des Genealogie-Daten-Anchors entsprechen den Feldern der Genealogie-Daten. Der Material-Anchor besitzt die Atributes MaterialNumber, ShortDescription, Amount und Unit. Der Produkt-Anchor besitzt lediglich das Attribute ProductNumber. Wie schon im Grundlagen Kapitel 2 erwähnt, ist die Erstellung eines Datenschemas ein iterativer Prozess. Die Entitäten und Attributes können sich im Laufe der Zeit noch ändern, sowohl während als auch nach der Erstellung des Schemas.

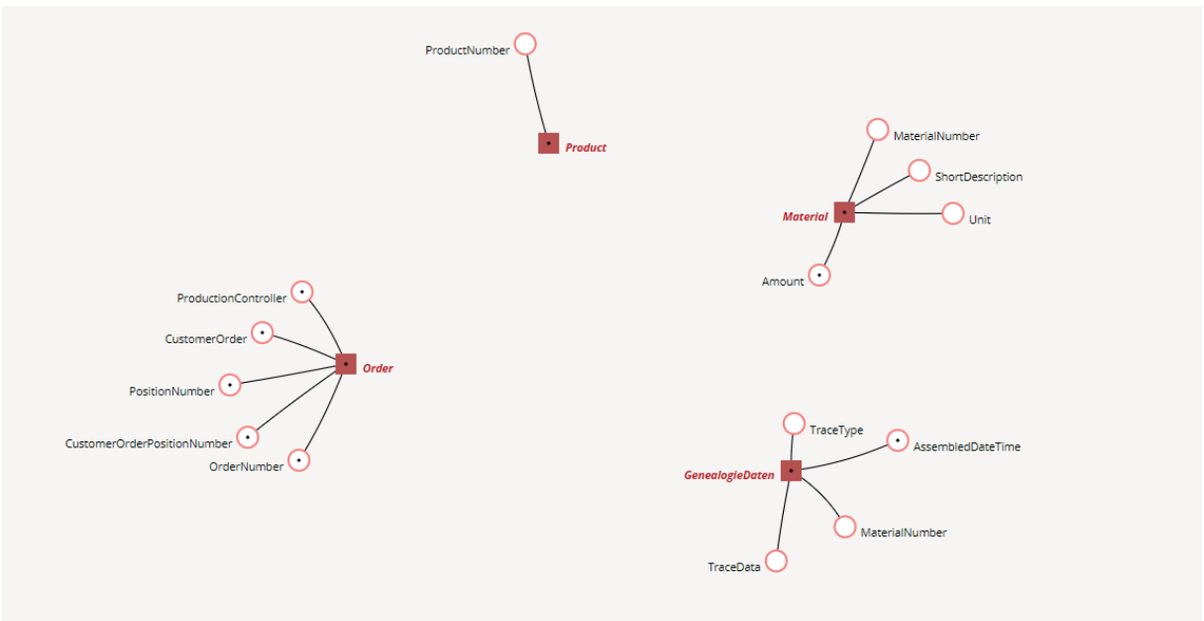

Abbildung 3.2: Die Anchors und Attributes des Datenschemas

3.3.2 Ties

Ties werden genutzt, um Beziehungen zwischen den Entitäten zu modellieren. Sie sollten möglichst einfach gehalten sein, damit Anomalien vermieden werden können ([40]). Meist können Beziehungen zwischen Entitäten aufgrund ihres Verhaltens oder

ihrer Interaktion identifiziert werden. Einige Beziehungen müssen erst gefunden oder genauer beschrieben werden, bevor sie erkannt werden.

In den Produkt-Daten gibt es, neben den Genealogie-Daten, eine weitere verschachtelte Struktur: die `ReferencedProductionOrders`. In dieser Struktur ist eine Referenz zu einer Auftragsnummer enthalten. Mit dieser Referenz können die dazugehörigen Genealogie-Daten in den Auftrags-Daten gefunden werden. Diese werden ebenfalls über eine Auftragsnummer identifiziert. Diese Beziehung besteht zusammengefasst zwischen einer Auftragsnummer und einer Auftragsnummer. Beides sind Eigenschaften des Auftrag-Anchors. Es kann eine Tie modelliert werden, welche diese Beziehung abbildet und zwischen dem Auftrag-Anchor und sich selbst liegt. Im Anchor Model wird jedem Ende einer Tie eine Rollenbezeichnung zugeordnet. Für den Anwendungsfall wird für ein Ende die Bezeichnung `referenziert` und für die andere `von` verwendet. Beide Enden des Tie beziehen sich auf den Auftrag-Anchor. Diese Tie liest sich wie folgt: ‚Ein Auftrag referenziert von einem Auftrag'. Einen Identifier für diesen Tie gibt es nicht, da an beiden Enden des Ties derselbe Schlüssel mehrfach vorkommen kann. Das liegt daran, weil weder die Spalte `ProductionOrderNumber`, noch die Spalte `ReferencedProductionOrders` in den Produkt-Daten eindeutig sind. Diese Verknüpfung ist aber ebendiese, welche im Anwendungsfall modelliert wird und somit auch für dessen Tie gültig ist.

Weitere Beziehungen können aus den Rohdaten abgeleitet werden. Die Genealogie-Daten liegen als Liste in einem Objekt mit einer Auftragsnummer vor. Somit kann eine Tie zwischen dem Auftrag-Anchor und Genealogie-Daten modellieren. Auch aus Sicht des Anwendungsfalles macht diese Beziehung Sinn, da die Genealogie-Daten dabei helfen sollen, Produkte und Materialien eines Auftrags zurückzuverfolgen. Die Rollenbezeichnungen für diesen Tie sind am Ende des Auftrags `rückverfolgbar` und am Ende der Genealogie-Daten `durch`. Da einem Auftrag mehrere Genealogie-Daten gehören können, kann `durch` nicht als Identifier genutzt werden. Eine Instanz der Genealogie-Daten hingegen, kann nicht zu verschiedenen Aufträgen gehören, dadurch kann die Rolle `rückverfolgbar` als Identifier genutzt werden.

Weitere Rollen können durch Analyse des Anwendungsfalles und der Domäne gefunden werden. Zum Beispiel ist es sinnvoll anzunehmen, dass zu einem Auftrag auch ein Produkt gehört und ein Produkt aus Materialien besteht. So lassen sich zwei weitere Ties modellieren: Eine Beziehung zwischen Auftrag und Produkt und eine zwischen Produkt und Material. Die Beziehung zwischen Auftrag und Produkt lässt sich in den SAP-Daten wiederfinden. In dieser Tabelle gibt es neben der Spalte der Auftragsnummer auch eine, welche eine MLFB enthält. Mithilfe der MLFB, ist es mögliche ein Produkt zu identifiziert. Die Rolle in diesem Tie ist am Ende des Auftrags `in` und am Ende des Produkts `ist enthalten`. Je nachdem, wie man die Entität `Produkt` modelliert, gibt es verschieden Möglichkeiten für einen Identifier. Wenn der Produkt-Anchor alle möglichen und unterschiedlichen Produkte enthält, stellt er eine Art Produktkatalog dar. Ist dies der Fall, kann keine der beiden Rollen als Identifier genutzt werden. Das liegt daran, dass mehrere Produkte zu einem Auftrag gehören können, aber dieselben Produktarten auch in anderen Aufträgen vorkommen können. Wenn der Produkt-Anchor aber jedes hergestelltes Produkt beinhaltet, also eine Art

Produktionstagebuch ist, kann dasselbe Produkt nicht in mehreren Aufträgen vorkommen. Da es sich in diesem Fall nicht um ein Produktkatalog handelt, sondern um eine Übersicht aller hergestellten Produkte, wird der Produkt-Anchor wie in der zweiten Möglichkeit beschrieben, modelliert. Ein anderer Ansatz ist es, diese zwei Sichtweisen in unterschiedlichen Anchors zu modellieren.

Die Rolle zwischen Produkt und Material lässt sich mit ähnlichen Argumenten wie die Rolle zwischen Auftrag und Produkt identifizieren und modellieren. Die Rolle am Ende des Produkt-Anchor wird als `von` und am Ende des Material-Anchor als `ist Bestandteil` bezeichnet. Auch hier stellt sich wieder die Frage, wie der Material-Anchor modelliert wird. Entweder repräsentiert er alle möglichen verbrauchten/benötigten Materialien je Auftrag oder repräsentiert einen Materialkatalog. Bei diesem Tie ist ein weiterer Aspekt zu beachten. In den SAP-Daten gibt es verschiedene Spalten, die Materialien beschreiben. Darunter eine Materialnummer, eine Materialbeschreibung, die benötigte Menge und die Einheit. Modelliert man den Material-Anchor als einen Materialkatalog, dann handelt es sich um eine many-to-many Beziehung zwischen Produkt und Material. Das bedeutet, dass ein Produkt aus mehreren Materialien besteht und ein Material in mehreren Produkten vorkommen kann. Es ist zudem möglich, dass dieselben Materialien in verschiedenen Produkten in einer anderen Menge gebraucht werden. Möchte man die Menge und Einheit zu einem Material wissen, ist dies abhängig von der Kombination aus Produkt und Material. Bei einer many-to-many Beziehung muss dies als Eigenschaft des Ties modelliert werden. Um so etwas zu modellieren, gibt es im Anchor Modeling zwei Möglichkeiten. Es kann die Tie um einen (oder im Allgemeinen auch mehrere) Knot oder um einen (oder mehrere) Anchor erweitert werden. Ein Knot wird normalerweise genutzt, um einen Zustand oder beschränkten Wertebereich zu modellieren. Da die Eigenschaft der Beziehung mehrere Attribute (Menge und Einheit) aufweist, muss ein Anchor genutzt werden. Dieser Anchor wird der Tie hinzugefügt und modelliert die Eigenschaften der Beziehung. Durch den zusätzlichen Anchor können aber Anomalien entstehen. Sind Einheit und Menge nicht vorhanden, kann die Beziehung nicht modelliert werden, selbst wenn Produkt und Material bekannt sind. Modelliert man den Material-Anchor hingegen, wie den Produkt-Anchor, als eine Liste aller benutzten (oder benötigten) Materialien, kann ein Eintrag aus dem Material-Anchor nicht zu verschiedenen Einträgen aus dem Produkt-Anchor gehören. Das liegt daran, dass jeder Eintrag aus dem Produkt-Anchor ein produziertes Produkt repräsentiert, für welches genau dieses Material benutzt wurde. Dadurch ist die Beziehung zwischen Produkt und Material eine one-to-many Beziehung. One-to-many, weil ein Produkt aus mehreren Materialien bestehen kann. In einer one-to-many Beziehung ist es möglich, die Eigenschaft der Beziehung, im vorliegenden Fall die Menge und Einheit, auf die Seite des many-Anchors zu modellieren. Dadurch bleiben die Eigenschaften Einheit und Menge im Anchor Material erhalten und die Tie zwischen Produkt und Material ist so einfach wie möglich gehalten.

Schaut man die Auftrags- und Produkt-Daten weiter an, erkennt man in den Genealogie-Daten die Spalte `MaterialNumber`. Da im Schema dieser Arbeit ein Anchor vorhanden ist, der Material modelliert, könnte eine Beziehung mit einer Tie modelliert werden.

Der entwickelte Material-Anchor modelliert jedoch keinen Materialkatalog, eine eindeutige Zuordnung ist nicht möglich. Würde der Anchor dies tun, würde ein Tie als Beziehung eher modelliert werden können. Um von den Genealogie-Daten auf die dazugehörigen Materialien zu gelangen, muss über den Auftrag und das Produkt gegangen werden. Dies ist später über die Abfragen möglich und wird deswegen hier nicht als direkte Beziehung modelliert. Dadurch entsteht das neue, erweiterte Schema, welches in Abbildung 3.3 zu sehen ist. Das Schema wurde um Ties zwischen `Order` und `Product`, `Product` und `Material`, `Order` und `Order` sowie `Order` und `GenealogyData` erweitert.

Bei der Modellierung von Ties wurde deutlich, dass es verschiedene Herangehensweisen gibt, wie Daten repräsentiert werden können. Welche Methode für ein Schema am besten geeignet ist, hängt von verschiedenen Faktoren, wie den Anforderungen, technischen Gegebenheiten und der Art der Daten ab. Die Entscheidung für eine konkrete Modellierung muss deswegen immer sorgfältig getroffen werden und die verschiedenen Möglichkeiten gegeneinander abgewogen werden.

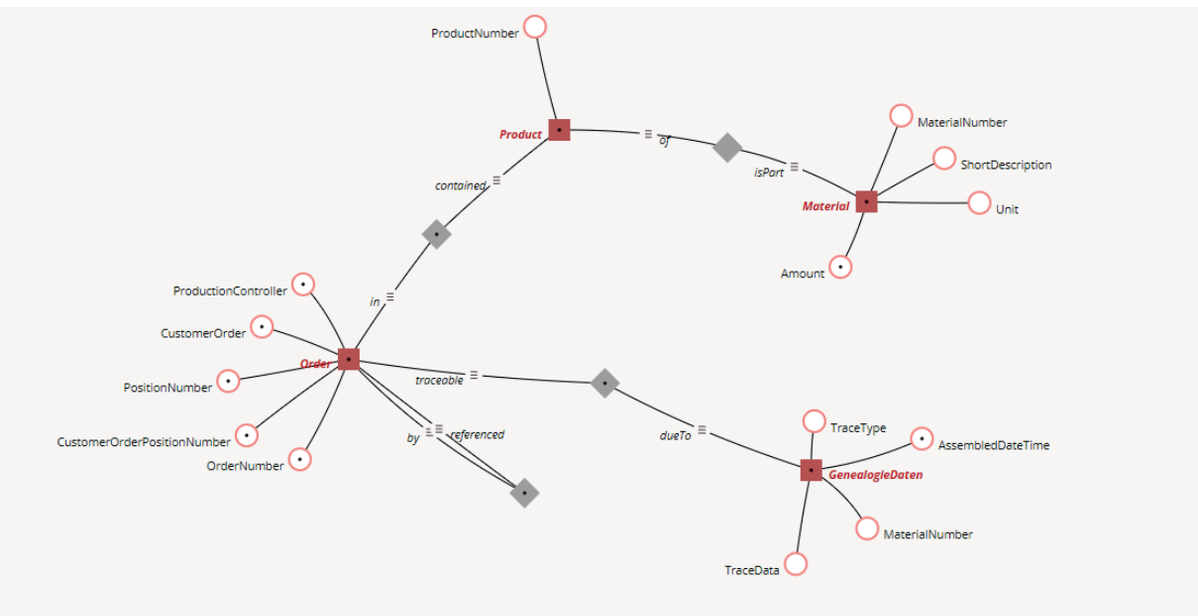

Abbildung 3.3: Das Schema erweitert durch Ties

3.3.3 Knots

Knots werden dann eingesetzt, wenn eine Eigenschaft einen Wert aus einer begrenzten Menge annehmen kann. Sie können auch genutzt werden, um einen Zustand zu modellieren. Modellieren Knots einen Zustand, sollen die möglichen Werte sich gegenseitig ausschließen.

Betrachtet man die Rohdaten genauer, stößt man auf das Feld `TraceType` in den Genealogie-Daten. Dieses Feld beschreibt, um welche Art von Genealogie-Daten es sich handelt und wie diese zu interpretieren sind. Die möglichen Werte beschränken

sich auf einen eingeschränkten Wertebereich, den der Ganzzahlen. Aufgrund dieser Eigenschaften des Feldes `TraceType` wird es als ein knotted Attribute modelliert. Ein knotted Attribute ist ein Attribute, welches mit einem Knot verknüpft ist. Im jetzigen Zustand des Schemas bringt diese Modellierung bis auf die Normalisierung keine weiteren Vorteile. Durch die Normalisierung wird jedoch festgelegt, dass der Wertebereich des Feldes `TraceType` auf die möglichen Werte beschränkt ist. Sind in diesem Feld andere Werte vorhanden, so kann dies erkannt werden. Ein weiterer Vorteil kommt zum Vorschein, sobald das Attribut `TraceType` in anderen Anchors genutzt wird. Dann kann der Knot an verschiedenen Stellen im Schema genutzt und dadurch Redundanzen vermieden werden.

Eine weitere Möglichkeit für ein knotted Attribut ist das Attribut Einheit des Material-Anchors. Es existiert eine begrenzte Anzahl an möglichen Einheiten, die genutzt werden können. Diese Anzahl ist jedoch sehr hoch, was gegen die Modellierung als Knot spricht. In diesem Fall hilft ein Knot jedoch dabei, die Einheiten zu vereinheitlichen. So können Einheiten ein unterschiedliches Format aufweisen, wie z. B. `kg` und `Kilogramm`, wodurch Anomalien entstehen können. Durch die Modellierung mit einem Knot existiert nur ein Eintrag für die Einheit Kilogramm und diese wird an entsprechender Stelle referenziert. Die Darstellung der Einheit, ausgeschrieben oder abgekürzt, kann in der Anwendung erfolgen. Dadurch konnte das Schema um die zwei Knots `Unit` und `TraceType` erweitert werden (siehe Bild 3.4).

Weitere Modellierung von Knots können je nach Anforderungen und Aufbau von neuen Daten und Datenquellen erfolgen. Soll zu einem späteren Zeitpunkt ein Attribut als knotted Attribut modelliert werden, ist dies nur bedingt möglich. Es können neue knotted Attribute modelliert werden, welches die analogen Daten des zu ersetzenden Attributs enthält. Dadurch ist das alte Attribut nicht mehr notwendig und kann entfernt werden. Es ist somit von Vorteil, wenn bei der Modellierung frühzeitig darauf geachtet wird, welche Attributes mit einem Knot versehen werden können.

3.3.4 Historisierung

Attributes und Ties können in einem Anchor Model auch historisiert werden. Das bedeutet, dass sich Einträge mit der Zeit ändern können und der alte Wert ungültig wird. Dadurch wird eine Versionierung der Daten ermöglicht, da die alten Werte nicht verloren gehen, sondern als ungültig markiert werden.

Geht man die einzelnen Attributes in dem bisher erstelltem Schema durch, finden sich nur wenige Attributes, die sich mit der Zeit ändern können. Die Auftragsnummer, Material- oder Produktnummern sind normalerweise statisch und werden nicht ungültig. Es ist möglich, dass eine neue Version eines Produkts auf den Markt kommt. Dann wird eine neue Produktnummer vergeben und das alte Produkt wird nicht mehr produziert. Dadurch wird die alte Produktnummer nicht ungültig. Die einzigen Attributes, die sich mit der Zeit verändern, sind die Attributes `Auftragsposition` und `Kundenauftragsposition`. Diese beschreiben den aktuellen Fortschritt eines Auftrags. Wird eine Position in einem Auftrag abgeschlossen, so wird die Position erhöht und die alte Positionsnummer ist nicht mehr gültig. Folglich werden diese zwei Attributes

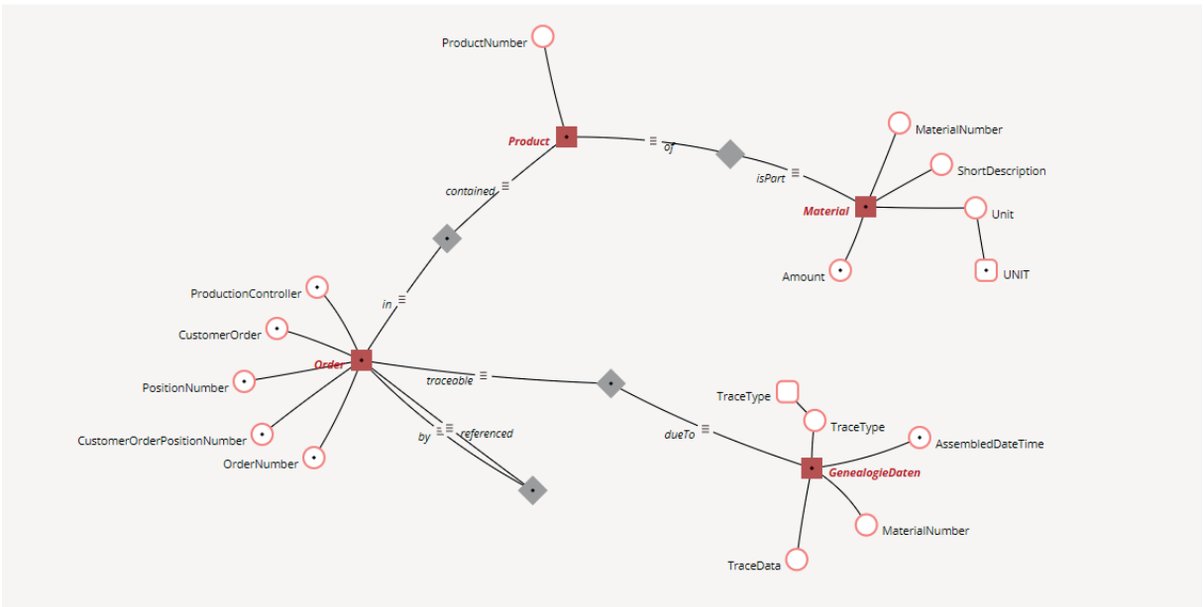

Abbildung 3.4: Erweiterungen des Schemas durch Knots

in dem Schema als historisiert modelliert. Ein Vorteil ist zudem eine mögliche, spätere Analyse von z. B. der Zeit zwischen den Änderungen der Auftragspositionen.

Bei Betrachtung der Ties lassen lässt sich keiner finden, der eine Historisierung gerechtfertigt. Die meisten Daten in diesem Schema stellen die Aufträge und die Verbindung mit ihren konkreten, den wirklich produzierten, Produkten dar. Es wird kein abstrakter Produktkatalog angeboten, bei dem sich die Produktinformationen ändern können. Deswegen ist in diesem Schema wenig Historisierung notwendig. Bei der Erweiterung des Schemas um eine Historisierung ist kein großer Aufwand notwendig, da den Attributes und Ties problemlos ein Zeitstempel hinzugefügt werden kann. Das fertige Schema ist in Bild 3.1 zu sehen.

Beim Erstellen eines Datenschemas stellen sich viele Fragen und Herausforderungen, die es zu lösen gilt. Dabei gibt es häufig verschiedene Herangehensweisen, die von Experten und Nutzern sorgfältig abzuwiegen sind. Die Erstellung eines Datenschemas ist ein agiler und iterativer Prozess, der sich über die gesamte Lebenszeit eines Datenschemas erstreckt. Es wird ständig erweitert und verändert, wenn sich Anforderungen ändern oder neue Erkenntnisse gewonnen werden. Das erstellte Schema basiert auf Überlegungen zu dem dazugehörigen Anwendungsfall und den zugrunde liegenden Rohdaten, welche abstrahiert werden sollen. Es stellt eine Grundlage zur Erstellung einer einheitlichen Schnittstelle für die Nutzdaten dar, die von verschiedenen Anwendungen genutzt werden können und erweiterbar ist.

Kapitel 4

Umsetzung in SDI

In diesem Kapitel werden die Vorgehensweisen und Möglichkeiten erkundet, um ein Datenmodell in SDI umzusetzen. Für die Umsetzung wird die Funktion von SDI verwendet, Ontologien zu erstellen. Der Aufbau der SDI-Ontologie soll der des, in Kapitel 3, erstellten Schemas entsprechen. Dazu werden die Anchor des Schemas auf Klassen in der SDI-Ontologie abgebildet. Die Attributes der einzelnen Anchors beziehen sich auf die Eigenschaften der Klassen einer SDI-Ontologie. Dabei können die einzelnen Eigenschaften mit den Spalten der von SDI bereitgestellten Schemata der Nutzdaten verknüpft werden.

Als Erstes wird erklärt, wie die Datenquellen im SDI registriert und strukturiert erstellt werden können. Danach wird das entworfene Datenschema in SDI umgesetzt. Dazu werden Ontologien genutzt. Dabei werden auch verschiedene Probleme, die dabei auftreten, erläutert und diskutiert. Danach werden unterschiedliche Ansätze vorgestellt, um die aufgetretenen Probleme zu lösen.

4.1 Registrierung der Datenquellen

Als Erstes wurden für die drei Datenquellen (Produkt, Auftrag, SAP) jeweils ein neues Datenregister angelegt. Dabei ist es sinnvoll, die drei Datenquellen mithilfe des Source Names zu gruppieren, da diese speziell für den Anwendungsfall zusammen- und bereitgestellt wurden. Da es sich hierbei um Daten handelt, die zur Nachverfolgung genutzt werden, wurde der Name `trace_data` gewählt. Der Source Name dient als Namespace für den Anwendungsfall `trace_data`, also eine Gruppierung von Daten. Diese Gruppierung hat keinerlei Auswirkung auf die Sichtbarkeit der Daten, gibt ihnen aber eine semantische Struktur. Die Data Tags können direkt aus den Namen der Datenquellen abgeleitet werden. Der Data Tag für die Produkt-Daten ist z. B. `product`. Damit ist auch definiert, wie die Daten in einer physischen Abfrage angesprochen werden können. Die Datenquelle `Auftrag` kann in Abfragen mit `trace_data_orders` angesprochen werden.

Bei den Datenquellen handelt es sich hauptsächlich um Produktionsdaten. Diese sind

also meist unvollständig und können immer durch neue Daten erweitert werden. Bei der Erstellung des Datenregisters wird deshalb die Erweiterung als Strategie zum Hochladen neuer Dateien gewählt. Die Ersatzstrategie würde sich bei dieser Art von Daten nicht eignen. Bei Daten, wie einem Produktkatalog, werden Datensätze meist im Ganzen verändert. Diese Art von Daten würden die Nutzung der Ersatzstrategie besser rechtfertigen.

Sind die drei Datenregister angelegt, müssen diese mit einer Datenquelle verknüpft werden. Dafür wird für jede Datenquelle ein Ordner im IDL angelegt und wie in den Grundlagen (2.3) mit den erstellten Datenregistern verknüpft. Danach können die einzelnen Dateien in den zugehörigen Ordnern hochgeladen werden. SDI beginnt nun automatisch mit der Erstellung der Schemata für die einzelnen Datenregister. Nachdem dies abgeschlossen ist, ist es möglich, die Nutzdaten in Abfragen oder für SDI-Ontologien zu nutzen.

4.2 Umsetzung des Datenschemas

Das in Kapitel 3 erstellte Datenschema soll nun in SDI umgesetzt werden. Dafür sollen SDI-Ontologien genutzt werden. Diese erlauben es, die Spalten der einzelnen Datenquellen-Schemata auf Klassen und Eigenschaften abzubilden. Die Herausforderung besteht darin, die im Anchor Model definierten Anchors, Attributes, Knots und Ties in SDI zu modellieren. Um die Vorteile des Anchor Models nutzen zu können, muss das Datenschema so umgesetzt werden, dass es in sechster Normalform vorliegt. Dafür müssen die verschiedenen Komponenten jeweils als eigene Klasse definiert werden, so wie es in einer relationalen Datenbank implementiert werden würde (vgl. [40, S.1239 ff.]). Unter diesen Rahmenbedingungen ist es nicht möglich, das Datenschema 1:1 in SDI umzusetzen. Es gibt verschiedene Aspekte, die das Umsetzen des Datenschemas in SDI erschweren. Diese Probleme werden im Folgenden vorgestellt.

Ein wichtiges Konzept, welches Anchor Modeling nutzt, ist die Verwendung von Schlüsseln. In Datenbanken gibt es Schlüssel, die für die Identifizierung von Datensätzen verwendet werden. Anchor Modeling benötigt diese Schlüssel, um die Daten richtig miteinander zu verknüpfen. Ein Attribute wird dann als eine Tabelle implementiert, welche sich aus der Ausprägung des Attributes und einem Fremdschlüssel des zugehörigen Anchors zusammensetzt. In den Datenquellen gibt es nicht für alle modellierten Anchors einen möglichen Schlüssel, der einen Eintrag eindeutig identifizieren kann. Für einen Auftrag gibt es die Auftragsnummer, die als Schlüssel genutzt werden kann, welcher einen Auftrag eindeutig identifiziert. Es existiert jedoch kein eindeutiger Schlüssel für z. B. die Genealogie-Daten. Die Auftragsnummer kann nicht verwendet werden, da mehrere Genealogie-Daten einem Auftrag zugehörig sein können. Auch für die Anchor `Material` und `Produkt` liegen keine eindeutigen Schlüssel vor. Zwar gibt es die eindeutige Produkt- und Materialnummer, die das dazugehörige Gegenstück identifizieren können, allerdings kann dasselbe Produkt oder Material mehrfach als Eintrag vorhanden sein. Das liegt an der Modellierung der zwei Anchors als ein Logbuch aller Produkte und Materialien. Neben natürlichen Schlüsseln finden sich

auch künstliche Schlüssel, die nur für den Zweck der Identifikation eines Datensatzes erstellt werden. Die einfachste Form eines künstlichen Schlüssels ist ein Zähler, der bei jedem neuen Datensatz um eins erhöht wird. Allerdings ist es in SDI nicht möglich, einen künstlichen Schlüssel für eine Klasse zu generieren. Selbst wenn, müsste dieser so erstellt werden, dass er an allen Stellen auch korrekt referenziert wird.

Ein weiteres Problem ist die Verwendung von Knots. Es gibt keine Möglichkeit, einen Knot in SDI zu definieren. Bei Knots handelt es sich jedoch um ein spezielles Konzept von Anchor Modeling. Mithilfe einer Klasse würden sich Knots modellieren lassen allerdings, jedoch unter der Aufwendung zusätzlicher Arbeit. Bei der Vorbereitung der Nutzdaten oder anschließend zur Erstellung von Abfragen und Erweiterungen des Schemas. Ist die SDI-Ontologie erstellt, kann eine Abfrage genutzt werden, um die unterschiedlichen Werte aus einem Attribute zu extrahieren. Das Ergebnis dieser Abfrage kann als neue Datenquelle im SDI genutzt und für die Implementierung des Knots verwendet werden. Das bereits beschriebene Problem der Schlüssel bleibt bestehen, da diese für Knots ebenfalls benötigt werden.

Ein anderes Problem tritt bei der Abbildung der Klasseneigenschaften auf Spalten der Datenquellen auf. Eine Abbildung von einer Eigenschaft auf eine einzelne Klasse ist ohne Einschränkung möglich. Soll eine Eigenschaft auf mehrere Spalten abgebildet werden, gibt es einiges zu beachten. Wird eine Eigenschaft auf mehrere Spalten abgebildet, müssen die Spalten verknüpft werden. Diese Verknüpfung wird in SDI mittels eines LEFT JOIN realisiert. Die Datenquellen kommen dabei in alphabetischer Reihenfolge vor. Das bedeutet, dass die alphabetisch erste Datenquelle die linke Seite des Joins abbildet. Dadurch wird die Reihenfolge der Datenquellen festgelegt und die Möglichkeiten der Verknüpfung eingeschränkt. Um alle Genealogie-Daten aus den zwei Datenquellen aufzunehmen, müsste ein FULL OUTER JOIN für die Verknüpfung genutzt werden. Durch den LEFT JOIN ist es nur möglich, die Genealogie-Daten aus der alphabetisch ersten Datenquelle zu verknüpfen. Sollte in einem anderen Anwendungsfall für ein LEFT JOIN die korrekte Verknüpfung vorliegen, gibt es keine Möglichkeit, die Seiten des Joins zu tauschen.

Probleme treten nicht nur bei der Erstellung der SDI-Ontologien auf. Um die Nutzbarkeit von Anchor Models in SDI weiter zu analysieren, wurde ein kleineres, einfacheres Testschema erstellt, welches in SDI umgesetzt werden kann. Das Datenschema und die Schemata der Testdaten sind in Anhang B und Anhang C zu finden. Das Datenschema ist ein kleines Anchor Model mit drei Anchors (Product, Order, Customer). Ein Kunde steht in Verbindung mit einem oder mehreren Aufträgen und ein Auftrag besteht aus mehreren Produkten. Die Testdaten wurden so aufgebaut, dass sie mit einer SDI-Ontologie umgesetzt werden können. Dieser Aufbau wird nochmal genauer in Abschnitt 4.2.1 beschrieben. Dadurch konnte erkannt werden, dass auch im weiteren Verlauf Probleme auftreten.

Durch die hohe Normalisierung wird in Datenbanken, die Anchor Modeling nutzen, mit Views gearbeitet. Mithilfe von Views kann diese Normalisierung wieder rückgängig gemacht werden. Wird auf Views verzichtet, werden die Abfragen sehr komplex, da sie viele Joins enthalten, um die Nutzdaten wieder zusammenzuführen (vgl. [40, S. 1241]).

In SDI gibt es keine Möglichkeit, Views zu erstellen. Es besteht die Möglichkeit, die Abfragen sehr komplex zu gestalten oder Views auf eine andere Art und Weise zu simulieren. Views zu simulieren kann auf eine ähnliche Weise wie bei Knots geschehen. Es wird eine Abfrage erstellt, welche die Nutzdaten aus den Datenquellen zusammenführt und das Ergebnis als neue Datenquelle registriert. Diese Datenquelle kann dann als Klasse in der SDI-Ontologie hinzugefügt und für weitere Abfragen genutzt werden. Ein Nachteil bei dieser Vorgehensweise ist, dass der Prozess schwer zu automatisieren ist. Werden Ergebnisse einer Abfrage in IDL gespeichert, werden diese in einem Ordner mit dem Zeitstempel der Ausführung abgelegt. Ändern sich die zugrundeliegenden Nutzdaten, werden die Abfragen zwar automatisch neu ausgeführt, allerdings die Ergebnisse in einem neuen Ordner abgelegt. Diese müssten manuell in den für die View gedachten Ordner im IDL kopiert werden.

Es ist auch möglich, auf die Views zu verzichten und die Abfragen direkt auf den normalisierten Klassen auszuführen. Dieses Vorgehen erhöht die Komplexität der Abfragen, welche sorgfältig erstellt werden müssen, um den Überblick zu wahren. Dieses Vorgehen würde dem Ziel, eine universelle, einfache Schnittstelle für die Daten anzubieten, widersprechen. Es ist zudem möglich, dass SDI nicht in der Lage ist die Abfrage auszuführen, wenn größere Datenquellen und Abfragen mit mehreren Join Operationen verwendet werden. Da ein JOIN eine aufwendige Operation ist und die Ausführungszeit der Abfragen in SDI auf 60 Sekunden begrenzt ist, besteht die Möglichkeit, dass die Abfrage nicht ausgeführt werden kann.

Anstatt Views zu nutzen, können auch verschachtelte Abfragen erstellt werden. Das sind Abfragen, die in ihren FROM oder JOIN Klauseln keinen Tabellennamen, sondern eine andere Abfrage enthalten. Um die Ergebnisse dieser Unterabfragen zu nutzen, müssen diese mit einem Alias versehen werden. Danach kann auf die Ergebnisse der Unterabfragen zugegriffen werden. In SDI ist es allerdings nicht möglich, Unterabfragen auf diese Weise zu nutzen. Wird einer Unterabfrage ein Alias zugeordnet und versucht, auf dieses zuzugreifen, liefert SDI einen Fehler. Dieser Fehler besagt, dass dieses Mapping nicht existiert. SDI erwartet in seinen Abfragen also nur Namen von Klassen, die auch in der SDI-Ontologie definiert wurden.

Es ist zusammengefasst nicht möglich, ein Anchor Model direkt in SDI umzusetzen. Es gibt aber verschiedene Ansatzpunkte, die verändert werden können, um diese Probleme zu lösen. Im Folgenden werden die Ansätze vorgestellt und diskutiert. Darunter fallen die Veränderung der Datenquelle, sodass das Schema besser umgesetzt werden kann und die Veränderung des Datenmodells und -schemas, sodass es mit den Funktionen die SDI bietet umgesetzt werden kann. Das Datenmodell wird dabei auf Kosten der Richtlinien und Normalisierung verändert. Abschließend werden mögliche Verbesserungen bzw. Funktionalitäten vorgestellt, die SDI implementieren könnte, um die Umsetzung von Anchor Modeling zu erleichtern.

4.2.1 Veränderung der Datenquellen

Eine mögliche Stellschraube, um die Umsetzung von Anchor Modeling in SDI zu erleichtern, ist die Veränderung der Datenquellen. Dabei gibt es verschiedene Aspekte,

die untersucht werden können. Würde die Datenquelle so umgeformt werden, dass sie der Form des Schemas entspricht, also bereits in Form eines Anchor Models vorliegt, ist es möglich, das Schema umzusetzen. Dann könnte das Schema umgesetzt werden, indem die Datenquelle 1:1 als Klasse in die SDI-Ontologie übersetzt wird. Die SDI-Ontologie bietet dann keine weitere Abstraktionsebene, bis auf die Umbenennung der Spalten und Klassennamen. Dadurch entstehen viele Dateien und Datenregister, die in SDI registriert werden müssen. Außerdem werden dadurch nicht die Probleme gelöst, die bei der Ausführung der Abfragen auftreten. Dieser triviale Ansatz bildet die Grundlage für die folgenden Überlegungen. Es werden Möglichkeiten gesucht, um die Struktur der Datenquellen zu verändern, ohne dass sie dabei bereits als Anchor Model vorliegen müssen.

Anstatt die Nutzdaten komplett in sechster Normalform vorliegen zu haben, reicht es aus, die Nutzdaten in dritter Normalform vorliegen zu haben. Dies entspricht dem Aufbau der von Anchor Modeling verwendeten Views von Anchors. Die Attributes der Anchors liegen je nach Dateiformat als Spalte oder Feld in der Datenquelle vor. Knots müssen als einzelne Datenquelle vorliegen, da in SDI keine einfache Möglichkeit besteht, Knots zu simulieren. Damit die Nutzdaten in SDI, gemäß Anchor Model, aufgespalten werden können, wird ein Schlüssel benötigt. Dies kann mit einem künstlichen Schlüssel oder aber mit einem natürlichen Schlüssel erreicht werden. Auch wenn im Anchor Model natürliche Schlüssel nicht vorgesehen sind, können diese für die Umsetzung in SDI genutzt werden. Es ist auch möglich, einen zusammengesetzten (aus mehreren Spalten bestehenden) Schlüssel zu nutzen. Bei der Umsetzung als SDI-Ontologie müssen dann immer alle Spalten des Schlüssels in den Klassen der Attributes der Anchors vorhanden sein. Damit die Beziehungen zwischen den Anchors korrekt umgesetzt werden können, müssen die jeweiligen Schlüssel in den richtigen Beziehungsklassen vorhanden sein. Handelt es sich um eine one-to-one Beziehung zwischen zwei Anchors, muss in einer der beiden Datenquellen der eigenen Schlüssel sowie der Schlüssel des anderen Anchors vorhanden sein. Welche der beiden Datenquellen den Schlüssel des anderen Anchors enthält, ist dabei nicht von Relevanz. Bei einer one-to-many Beziehung muss der Schlüssel des Anchors, welcher die one-Seite der Beziehung darstellt, in der Datenquelle vorhanden sein, die den many-Anchor enthält. Das sorgt dafür, dass es keine verschachtelten Strukturen gibt, die beim Hochladen der Nutzdaten in SDI aufgelöst werden müssen. Wären die Beziehung andersrum aufgebaut, würde der Eintrag des one-Anchors seine Eindeutigkeit verlieren. Bei einer many-to-many Beziehung muss eine Datei erstellt werden, die die Beziehung zwischen den Anchors darstellt. In diesem Fall ist es nicht möglich, die Beziehung zwischen den Anchors ausschließlich mit der dazugehörigen Datenquellen darzustellen. Diese Vorgaben sind, ähnlich wie bei der Erstellung von Datenbanken, über eine eigene Tabelle dargestellt. Dadurch minimiert sich die Anzahl der benötigten Datenquellen auf die Anzahl der Anchors, Knots und many-to-many Beziehungen.

Soll die Anzahl der Datenquellen weiter minimiert werden, ist es möglich, mehrere Komponenten in einer Datenquelle zusammenzufassen. Die Datenquellen werden dadurch unübersichtlicher und zusätzliche Nullwerte werden bei unterschiedlichen Längen hinzugefügt. Dieses Vorgehen ist nur in bestimmten Situationen sinnvoll.

Für das entwickelte Schema bedeutet das, dass sieben Datenquellen benötigt werden. Davon sind vier Datenquellen für die Anchors, zwei für die Knots und eine für die many-to-many Beziehung zwischen dem Anchor `Auftrag` und sich selbst. Sind die Nutzdaten entsprechend umgeformt, können sie in SDI hochgeladen werden und mit einer Anchor-Model-konformen SDI-Ontologie verknüpft werden. Bei Abfragen stellen sich dieselben Probleme, welche bereits beschrieben wurden (siehe 4.2).

4.2.2 Veränderung des Datenmodells

In bestimmten Situationen ist es nicht möglich, die Datenquelle zu verändern. Dies ist der Fall, wenn Daten durch bestimmte Prozesse erstellt oder anderweitig verwendet und angepasst werden müssen. Eine weitere Möglichkeit wäre, das Datenmodell oder -schema anstatt der Daten zu verändern.

Die, umgekehrt zum vorherigen Ansatz, einfachste Möglichkeit ist es, das Datenschema so zu verändern, dass es der Form der Datenquelle entspricht. In diese Richtung ist dies jedoch nur bedingt möglich. In Anchor Modeling gibt es keine Möglichkeit, die Datenquelle so abzubilden, dass die Nutzdaten in dritter Normalform vorliegen. Es ist möglich, die Attributes der Anchors als Spalten anzusehen. Dann liegt das Modell nicht mehr in sechster Normalform vor. Die Vorteile von Anchor Modeling würden verloren gehen und das Schema würde eine reine Visualisierung der Nutzdaten darstellen. Zudem ist die Verwendung einer SDI-Ontologie über diesen Daten, bis auf die Umbenennung der Spalten, nicht sinnvoll, da direkt auf den Schemata der Datenquellen gearbeitet werden kann.

Allerdings ist der Ansatz, die Normalisierung des Datenmodells zu verringern, ein guter Gedanke. Somit könnten die Attributes in SDI nicht als eigenständige Klassen, sondern direkt als Eigenschaften der Anchor Klasse modelliert werden. Dadurch würde sich die Anzahl der Klassen in der SDI-Ontologie sowie die benötigten JOIN Operationen in den Abfragen verringern. Auch können die Ties, anstatt als eigene Klasse, als Eigenschaft der Anchor-Klassen modelliert werden. Dazu müssen die richtigen Fremdschlüssel in den richtigen Klassen vorhanden sein. Eine Beschreibung dieses Verhaltens findet sich im vorherigen Abschnitt. Die one-to-one und one-to-many Beziehung kann durch eine einfache Eigenschaft, die als Fremdschlüssel dient, dargestellt werden. Die many-to-many Beziehung muss wieder als eigene Klasse modelliert werden. Um die Verwendung von künstlichen Schlüsseln, die in den Datenquellen nicht vorhanden sind und in SDI nicht generiert werden können, zu umgehen, kann versucht werden, einen natürlichen Schlüssel zu nutzen. Ob ein natürlicher Schlüssel existiert, hängt von den Datenquellen ab und kann durch die Veränderung des Modells nicht beeinflusst werden. Zusätzlich zu den anderen Denormalisierungen, kann im Anchor Model auf Knots verzichtet werden. Diese können in SDI nicht direkt umgesetzt werden. Da Knots nicht zwingend notwendig sind, um etwas zu modellieren, gehen durch sie keine Informationen oder Anwendungsfälle verloren. Werden Knots nicht verwendet, verzichtet man auf einen höheren Grad an Normalisierung und die Möglichkeit, Knots an verschiedenen Stellen zu verwenden. Dadurch können Anomalien in den Nutzdaten entstehen.

Diese Verringerung der Normalisierung hilft allerdings bei den Problemen, die nach der Erstellung der SDI-Ontologie auftreten. Bei dem Erstellen von Abfragen wird die Komplexität sowie die Anzahl der JOIN Operationen verringert, was die Ausführung der Abfragen vereinfacht. Die Probleme, die bei der Erstellung der SDI-Ontologie auftreten, werden durch diese Veränderung nicht behoben. Es ist trotz dessen nicht in allen Situationen möglich, ein Anchor Model in SDI umzusetzen.

Folgt man dem anderen Ansatz, das Schema an den Aufbau der Datenquellen anzupassen, könnte das Schema so umgebaut werden, dass es in SDI umgesetzt werden kann. Dazu müssen einige Bedingungen erfüllt sein. Zum einen darf ein Anchor mit seinen Attributes im Allgemeinen immer nur in einer Datei vorkommen. Es müssten die Genealogie-Daten der Produkt- und Auftrags-Datenquellen in zwei verschiedene Anchor aufgeteilt werden. In bestimmten Situationen ist es möglich, dass die Daten eines Anchors auf mehrere Datenquellen verteilt sind. Ein Anwendungsfall, bei dem ein Zusammenfassen möglich ist, liegt vor, wenn die Datenquellen mittels eines LEFT JOIN verknüpft werden müssen. Dabei ist zu beachten, dass die Namen der Datenquellen alphabetisch im LEFT JOIN vorkommen. Ties können nur zwischen Anchors modelliert werden, deren Datenquellen durch eine Spalte verknüpft sind. Dabei müssen die Verknüpfungen die Form aufweise, welche bereits (siehe Abschnitt 4.2.1) für Beziehungen beschreiben wurde. Die richtigen Schlüsselspalten müssen in den korrekten Tabellen vorhanden sein. Sind diese Bedingungen erfüllt, kann das Schema in SDI umgesetzt werden. Da die Aufgabe allerdings darin besteht, eine SDI-Ontologie zu erstellen, die die Datenquellen verknüpft, ist es nicht sinnvoll, das Schema so zu verändern. Das Schema verliert dadurch seine Modellierungen und ist lediglich eine Darstellung des Aufbaus der Datenquellen. Da dies dem eigentlichen Vorhaben widerspricht, wird dieses Vorgehen als nicht sinnvoll erachtet.

4.2.3 Verbesserung von SDI

Einige Probleme beruhen auf den Einschränkungen von SDI. Diese Einschränkungen sind in den oberen Abschnitten bereits beschrieben. Im Folgenden sollen diese genauer betrachtet und Alternativen oder neuen Funktionalitäten diskutiert werden.

Eine Herausforderung bei der Umsetzung von Anchor Models ist es, Knots in SDI zu erstellen. Da Knots ein spezielles Konstrukt von Anchor Modeling darstellt, werden diese nicht in einer allgemeinen Applikation wie SDI unterstützt. Wie bereits beschrieben, besteht die Möglichkeit, Knots mithilfe von Abfragen zu erstellen. In der Abfrage wird das DISTINCT Keyword genutzt, welches alle unterschiedlichen Werte einer Spalte extrahiert. Das Ergebnis kann als neue Datenquelle registriert und als Knot genutzt werden. Ähnlich dieser Vorgehensweise wäre es möglich, in SDI ein Distinct Abbildung einzuführen. Dies erlaubt es, alle unterschiedlichen Werte einer Spalte auf die Eigenschaft einer Klasse abzubilden. Dadurch können Knots direkt in der Erstellung der SDI-Ontologie umgesetzt werden. Knots sind ein geeignetes Mittel, um die Normalisierung zu erhöhen und um die Ausprägungen eines Typs zu vereinheitlichen. Allerdings sind Knots nicht essenziell notwendig, um ein Datenmodell zu erstellen. Verzichtet man auf Knot, können weiterhin dieselben Sachverhalte modelliert

werden. Anstatt die verschiedenen Ausprägungen auszulagern, werden diese direkt als Wert einer Eigenschaft gespeichert. Dies kann allerdings zu Anomalien führen, wenn die Rohdaten nicht sorgfältig gepflegt werden. Trotzdem ist die Umsetzung dieser Funktionalität in SDI nicht notwendig, da es auch ohne Knots möglich ist, ein Datenmodell zu erstellen.

Ein weiteres Problem ist die Verarbeitung von großen Datenmengen. In Abschnitt 4.2.1 wurde gezeigt, wie eine Datenquelle aufgebaut sein kann, um mittels eines Anchor-Model-basierten Schemas in SDI bereitgestellt werden zu können. Wird für diesen Ansatz das Testschema mit den kleinen Datenquellen genutzt, funktioniert dies einwandfrei. Die einzelnen Anchor können mit Abfragen denormalisiert werden und komplexere Abfragen, die mehrere Anchor verknüpfen und Daten gruppieren, können ausgeführt werden. Würden die Daten des Anwendungsfalles angepasst und das Schema in SDI umgesetzt werden, sind die Tabellen größer und die Ausführungszeit kann sich erhöhen. Das sorgt dafür, dass Abfragen eventuell nicht ausgeführt werden können, da die Bearbeitungszeit durch SDI begrenzt ist. Um dieses Problem zu lösen, könnte eine variable Begrenzung der Ausführungszeit eingeführt werden. Dadurch können komplexeren Abfragen mehr Zeit gegeben werden, um ausgeführt zu werden. Trotzdem muss die Ausführungszeit begrenzt werden, um eine Überlastung des Systems zu verhindern. Bessere Möglichkeiten zur Behebung dieses Problems stellen Optimierungen im Backend von SDI dar. Zu diesem Thema gibt es genügend Literatur wie u. a. [25], [24].

Ein anderes Problem entsteht beim Erstellen von Abfragen. Sollen in einer Abfrage die Attributes eines Anchors benutzt werden, muss der Anchor denormalisiert werden. Die Denormalisierung entsteht durch das Verknüpfen der Anchor-Tabelle mit den Tabellen der Attributes. Damit diese Ansicht wiederverwendet werden kann, wird sie in Anchor Model Implementierungen direkt als View bereitgestellt. Auf diesen Views lassen sich dann weitere Abfragen ausführen. Das erhöht die Übersichtlichkeit bei dem Zugriff auf die Attributes und hilft beim Erstellen von Verknüpfungen zwischen Anchors. In SDI ist es allerdings nicht möglich, Views zu erstellen. Es kann, ähnlich wie bei Knots, eine Abfrage erstellt werden, die die Nutzdaten denormalisiert und das Ergebnis dieser Abfrage wieder als Datenquelle registriert. Die Vor- und Nachteile dieser Methode wurden bereits diskutiert. Views sind ein mächtiges Werkzeug, dass es ermöglicht, Abfragen als virtuelle Tabellen zu nutzen und an verschiedenen Stellen wiederzuverwenden. Diese Funktion in SDI umzusetzen würde die Möglichkeiten von SDI erweitern. Zudem hilft es bei der Nutzung von SDI-Ontologien, die auf Anchor Models basieren. Eine denkbare Realisierung wäre, die Views als wiederverwendbare Abfragen zu implementieren. Dann wäre es möglich, innerhalb einer Abfrage auf die Ergebnisse einer anderen Abfrage zuzugreifen.

Ein anderes Konstrukt, welches den Views ähnlich ist, sind verschachtelte Abfragen. Das sind Abfragen, die in ihren Klauseln weitere Abfragen enthalten. Anstelle eines View könnte die Abfrage eingesetzt werden, die diese View erstellt. Solch eine Abfrage zu erstellen ist in SDI grundsätzlich möglich. Bei dem Zugriff auf die Spalten der Ergebnisse der Unterabfrage zeigen sich einige Probleme. In Standard SQL müssen Unterabfragen mittels des AS Keywords einen Namen zugewiesen bekommen. Mit

diesem Namen kann auf die Spalten der Unterabfrage zugegriffen werden. Wird versucht diese Methode auch in SDI anzuwenden, scheitert dies daran, dass SDI die Spalten der Unterabfrage nicht kennt. Das liegt daran, dass SDI in seiner SELECT Klausel nur Eigenschaften und Klassen aus der angegebenen SDI-Ontologie erwartet. Da die Ergebnisse der Unterabfrage nicht in der SDI-Ontologie definiert sind, kommt es zu einem Fehler in SDI. Somit ist es nicht möglich, die Ergebnisse einer Unterabfrage in SDI zu nutzen. Ein anderer Ansatz wäre diese Implementierung zu überarbeiten und die Ergebnisse von Unterabfragen in der SELECT Klausel zu erlauben.

Möchte man die Datenquellen auf die SDI-Ontologie übertragen, so kann es vorkommen, dass eine Eigenschaft einer Klasse auf mehrere Spalten in den Datenquellen verknüpft werden muss. Es gibt verschiedene Möglichkeiten, wie so ein Fall gehandhabt werden kann. In SDI werden die Datenquellen mit einem LEFT JOIN verknüpft. Die Reihenfolge der Tabellen im Join ist dabei lexikografisch nach dem Namen der Datenquelle. Werden zwei Spalten aus den Datenquellen `trace_data_orders` und `trace_data_products` verknüpft, wird die `trace_data_orders` immer als linke Tabelle genutzt. Die zwei Spalten werden in der ON Klausel genutzt, um eine Verbindung zu definieren. Gibt es mehrere solcher Eigenschaften, müssen diese alle zutreffen, damit die Zeile in der Ergebnismenge aufgenommen wird. Alle anderen, einfachen Verknüpfungen, werden ebenfalls in die Ergebnisse aufgenommen. Haben diese keine übereinstimmenden Werte durch die Verbindungsbedingung, werden für den Eintrag alle anderen Werte mit NULL aufgefüllt. Da der Nutzer keine Kontrolle über die Art des Joins, die Reihenfolge der Tabellen und die Verknüpfung der ON Bedingungen hat, werden die Möglichkeiten stark eingeschränkt. Bestünde bei der Konfiguration dieser Verknüpfungen mehr Freiheiten, wäre es mögliche zwei Datenquellen so zu verknüpfen, dass sie eine Union bilden. Dann müssten die Genealogie-Daten nicht aus den zwei Datenquellen Auftrag und Produkt extrahiert werden, sondern könnten über die Verknüpfung in der SDI-Ontologie definiert werden. Durch diese Funktion müssten jedoch zusätzlichen Bedingungen Beachtung finden und der Join muss immer korrekt gewählt werden.

Wesentlich für Anchor Models ist auch die Verwendung von Schlüsseln. In SDI ist es nicht möglich, Schlüssel zu erstellen oder Eigenschaften als Schlüssel zu definieren. Würde eine Möglichkeit zur Schlüsselgenerierung hinzugefügt werden, müssten diese immer noch korrekt verknüpft werden. Da in der SDI-Ontologie auf Spalten und Felder der zugrundeliegenden Datenquellen abgebildet wird, ist es nicht unmittelbar möglich, eine Referenz auf eine in der SDI-Ontologie definierte Eigenschaft zu erstellen. Es wäre aber möglich, beim Hochladen der Datenquelle einen Schlüssel zu generieren, der jeden Eintrag eindeutig identifiziert. Dieser kann anschließend in den SDI-Ontologien genutzt werden. Die Möglichkeit, Eigenschaften als Schlüssel zu definieren, könnte bei der Wartung und Pflege der Nutzdaten helfen. So kann SDI dafür sorgen, dass keine Daten hochgeladen werden, die einen falschen (doppelten) Schlüssel haben. Zudem könnte durch die Definition von Fremdschlüsseln dafür gesorgt werden, dass auch die Beziehungen zwischen den Datenquellen korrekt sind. Dadurch wird überprüft, ob der Eintrag wirklich auf einen gültigen Eintrag in der anderen Klasse verweist. Für die tatsächliche Umsetzung eines Schemas ist dies nicht zwangsweise nötig, da diese

Gegebenheiten durch die Datenquellen selbst sichergestellt werden können.

Es gibt noch weitere Funktionalitäten, die nicht direkt benötigt werden, aber zum allgemeinen Ziel, eine veränderbare, stabile Datenschnittstelle bereitzustellen, beitragen. Dazu gehört z. B. die Möglichkeit, SDI-Ontologien und Abbildungen im Nachhinein zu bearbeiten. Es gibt bereits die Möglichkeit eine SDI-Ontologie zu verändern, indem ihre JSON Repräsentation heruntergeladen, verändert und wieder hochgeladen wird. Dieses Vorgehen ist nicht sehr benutzerfreundlich und könnte über eine Benutzeroberfläche, ähnlich wie die zur Erstellung der SDI-Ontologie, verbessert werden. Das würde ermöglichen, unkompliziert Datenquellen einer SDI-Ontologie hinzuzufügen oder zu entfernen.

Zusammengefasst bedarf es noch einiger Verbesserungen, die das Arbeiten mit SDI erleichtern würden. Im Grundkonzept funktioniert die Applikation und ist in der Lage, eine stabile Schnittstelle zu erstellen. Zusätzlich zu den benötigten Funktionen in SDI, müssen auch die Datenquellen sowie das Datenmodell und -schema so gestaltet sein, dass sich die Nutzdaten einfach und sauber nutzen lassen können. Ein guter Ansatz um ein funktionierendes Anchor Model in SDI zu erhalten bedarf einer sinnvollen Kombination aus den oben genannten Punkten.

Kapitel 5

Evaluation

In dieser Arbeit wurden verschiedene Aspekte der Datenintegration betrachtet. Es wurde ein Datenschema erstellt, welches der Verknüpfung von Daten aus den verschiedenen Quellen dient. Dieses Schema stellt eine Schnittstelle dar, welche von Anwendern genutzt werden kann, um auf die Nutzdaten zuzugreifen. Für die Implementierung des Datenmodells wurde SDI, eine Applikation zur Datenintegration, verwendet. In SDI wurde, mithilfe von Ontologien, das Datenschema abgebildet und mit den Nutzdaten verknüpft.

In diesem Kapitel geht es um die Evaluation der zwei Aspekte Datenmodellierung und Umsetzung des Datenschemas. Dazu wird das Datenschema mit verschiedenen Metriken ausgewertet. Für SDI wird eine Fitness for Use Analyse durchgeführt, welche die Qualität der Anwendung beschreibt.

In [18] wurde eine Systematic Literature Review (SLR) durchgeführt, um Metriken für Datenmodelle zu finden. Dafür wurden 22 verschiedene Studien ausgewertet. In diesen Studien wurden verschiedene Metriken vorgestellt, die die Qualität eines Datenmodells bewerten. Verschiedene Metriken betrachten dabei verschiedene Aspekte von Datendimensionen. Welche Metriken für die Evaluation des Datenmodells verwendet werden, hängt vom Anwendungsfall und der Qualitätsmerkmale ab (vgl. [18]). Das erstellte Datenschema stellt eine Schnittstelle für die Anwender dar, weshalb Faktoren wie Verständlichkeit und Erweiterbarkeit wichtig sind. Zudem ist die Universalität des Datenschemas wichtig, da es für verschiedene Anwendungsfälle verwendet werden soll. Die Erweiterbarkeit des Datenschemas ist bereits durch die Modellierung mit Anchor Model gegeben, weshalb die Faktoren Verständlichkeit und Universalität für die Evaluation betrachtet werden.

Die einfachste Metrik ist die Anzahl an Tabellen, Attributen und Beziehungen im Schema (vgl. [16]). Im erstellten Schema sind insgesamt 24 Tabellen und 51 Attribute vorhanden. Die Anzahl der Beziehungen begrenzt sich dabei auf vier, wovon es sich bei nur einer Beziehung um eine many-to-many Beziehung handelt. Andere Metriken betrachten die Anzahl der Attribute (NA(T) [7]) und die Anzahl der Fremdschüssel in einer Tabelle (NFK(T) [7]). Diese Metriken werden je Tabelle ausgewertet und geben einen Hinweis auf die Komplexität der Tabelle. Die meisten Tabellen im Datenschema

sind ähnlich aufgebaut und besitzen einen NA Wert von 2 und einen NFK Wert von 1. Diese Werte kommen durch den hohen Grad an Normalisierung von Anchor Models zustande. Bei Ties und Knots erhöhen sich beide Werte um 1, da ein Schlüssel für die Tie selbst sowie die zwei Fremdschlüssel, die die Beziehung zu den beiden Tabellen herstellen, benötigt werden. Es gibt weitere Metriken, die aber für multidimensionale Datenmodell ausgelegt sind (vgl. [18]). Metriken speziell zur Bewertung eines Anchor Models, sind in der Literatur nicht zu finden.

Fitness for Use ist ein Begriff der von J. M.m Juran geprägt wurde und die Qualität eines Produktes beschreibt. Dazu werden fünf Fragen bearbeitet, mit deren Beantwortung die Qualität eines Produkt umfänglich bewertet werden kann (vgl. [32, S. 53]). Die Aspekte sind „*The users of the product or service*“, „*How the users will actually put the product or service to use*“, „*The possibility and probability of any danger to human safety*“, „*The economic resources of both the producer and the user*“, sowie „*The user's specific determinants of a product or service that is fit for their use*“ [32]. Diese Aspekte beinhalten die wichtigsten Faktoren, wie Nutzer, Nutzen, Sicherheit, Kosten und Anforderungen.

SDI hat verschiedene Nutzer, die die Applikation verwenden. Zum einen müssen Data Engineers die Datenquellen registrieren und eventuell in eine bestehende SDI-Ontologie einpflegen. Des Weiteren müssen Data Engineers, in Zusammenarbeit mit Modellierungs-Experten und Domänenexperten, ein Datenschema erstellen und dies in SDI umsetzen. Dafür müssen Daten im IDL hochgeladen und mit einem Register in SDI verknüpft werden. Das Erstellen der Datenregister und Verknüpfen mit den Datenquellen kann über die Benutzeroberfläche von SDI geschehen. Bei der erstmaligen Erstellung von SDI-Ontologien, wird eine Benutzeroberfläche bereitgestellt, die bei der Erstellung hilft. Mithilfe der Benutzeroberfläche können Klassen erstellt und ihnen Eigenschaften hinzugefügt werden. Diese Oberfläche bietet jedoch keine Möglichkeit, Ontologie zu visualisieren, wodurch es schwer ist, bei großen SDI-Ontologien, die Übersicht zu behalten.

Um das Datenschema zu erweitern, muss eine bestehende SDI-Ontologie heruntergeladen werden. Die heruntergeladene Datei enthält eine JSON-Repräsentation des Ontologie-Schemas. Dieses kann mit einem Texteditor bearbeitet und neue Quelldaten-Schemata, Eigenschaften, Klassen und Abbildungen hinzugefügt werden. Diese Trennung von Quelldaten-Schemata und Klassen der SDI-Ontologie ermöglicht eine unabhängige Veränderung der beiden Datenmodelle. Ändert sich die Struktur der Quelldaten, muss das Quelldaten-Schema angepasst und mit den Ontologie-Klasseneigenschaften verknüpft werden. Dabei ändert sich der Aufbau der Klassen nicht und Abfragen müssen nicht angepasst werden. Muss die SDIOntologie erweitert werden, bleibt die bisherige Schnittstelle erhalten, da die SDI-Ontologie auf Anchor Modeling basiert. Es müssen lediglich die neuen Klassen mit den Quelldaten-Schemata verknüpft werden. Alte Abfragen können weiterhin verwendet werden und müssen nicht angepasst werden. Da bei der Verwendung von Anchor Models viele Klassen erstellt werden müssen, werden die SDI-Ontologien schnell groß und unübersichtlich. Dies kann das Bearbeiten der JSON-Repräsentation der SDI-Ontologie erschweren. Die Namensgebung, welche für Anchor Models vorgeschlagen wird, bringt Struktur mit sich und hilft dabei Zusammenhänge zu verdeutlichen. Alleine reicht die Namensgebung jedoch nicht aus, um die

Übersichtlichkeit zu gewährleisten. Zusammengefasst bietet SDI die Grundfunktionen, welche für die Datenintegration benötigt werden. Aufgrund mangelnder qualitativer Aspekte, wie eine übersichtliche Erstellung von SDI-Ontologien, wird das Erstellen von qualitativ hochwertigen Datenmodellen erschwert.

Beim Verknüpfen der Datenquellen-Schemata mit der SDI-Ontologie gibt ebenfalls Schwierigkeiten. Das Verknüpfen einer Klasseneigenschaft mit mehreren Spalten eines Quelldaten-Schemas ist nur bedingt möglich und die Art der Verknüpfung ist nicht konfigurierbar. Dadurch werden bestimmte Verknüpfungen ausgeschlossen und müssen auf anderweitig gelöst werden. Entweder durch Transformation der Rohdaten oder durch Veränderungen im Datenmodell und/oder -schema.

Des Weiteren gibt es Nutzer von SDI, die auf die Daten zugreifen wollen. Dazu gehören Data Experten, wie Data Scientists, die die Daten für Analysezwecke verwenden wollen. Zudem existieren auch Domänenexperten, welche die Nutzdaten für ihre Arbeit benötigen. Aufgrund ihres Expertenwissens sind Data Scientists in der Lage, Abfragen auf dem Datenmodell zu erstellen, um die benötigten Daten zu erhalten. Domänenexperten, welchen das benötigte Wissen über Datenmodelle und SQL nicht aufweisen, können diesen Aufgaben nicht, ohne weitere Unterstützung, nachgehen. Um Abfragen zu erstellen, muss das Modell und dessen Aufbau verstanden werden. Möchte man in den SDI-Ontologien Anchor Modeling mit der dafür vorgesehenen Namensgebung (vgl. [40, S. 1233 f.]) umsetzten, entstehen dabei sehr lange und komplizierte Namen. Da die Ontologie nicht von SDI visualisiert wird, ist es schwer für Domänenexperten, das Schema zu verstehen. Dadurch ist die Wahrscheinlichkeit höher, dass Fehler beim Erstellen von Abfragen auftreten. Dies kann zu weiteren Problemen führen, wenn diese Fehler rein semantischer Natur sind und nicht von einer Syntaxprüfung durch SDI erkannt werden können. Datenexperten können eventuell besser mit dem Datenschema umgehen und ohne Visualisierung die Ontologie verstehen. In Zusammenarbeit mit den Domänenexperten können sie eine Abfrage erstellen, die die benötigten Daten in einer angemessenen Form zurückgibt.

Die Attributes und Anchors müssen über Verknüpfungen wieder denormalisiert werden. Da SDI keine Möglichkeit bietet Abfragen zu verschachteln, Views zu erstellen oder Abfragen in anderen Abfragen zu verwenden, können die Abfragen lang und komplex werden. Das kann die Lesbarkeit der Abfragen erschweren und die Wahrscheinlichkeit für Fehler erhöhen. Zudem müssen viele Verknüpfungen verwendet werden, wodurch die Bearbeitungszeit der Abfragen erhöht wird. Dies kann dazu führen, dass Abfragen nicht ausgeführt werden können, da die Bearbeitungszeit zu lange dauert und SDI die Ausführung der Abfrage abbricht.

Das Verwenden von Anchor Modeling für Ontologien in SDI hat Vor- und Nachteile. Ein Vorteil ist die Trennung zwischen den Quelldaten-Schemata und den Klassen der SDI-Ontologie, welche eine unabhängige Veränderung der beiden Datenmodelle ermöglicht. Durch die Verwendung von Anchor Modeling wird die SDI-Ontologie als eine stabile Schnittstelle entworfen, die leicht erweiterbar ist. Dadurch können Abfragen auf der Ontologie auch nach Veränderungen der Datenmodelle weiterhin verwendet werden. Diesen Vorteilen stehen jedoch auch Nachteile gegenüber. Die

Erstellung von SDI-Ontologien wird durch die fehlende Visualisierung erschwert. Dadurch wird es für Domänenexperten schwerer, die SDI-Ontologie zu verstehen und Abfragen zu erstellen. Durch die hohe Anzahl an Klassen, die benötigt werden, um eine SDI-Ontologie mit Anchor Modeling zu erstellen, wird die Ontologie schnell unübersichtlich. Die für Anchor Models vorgeschlagene Namensgebung bringt auf der einen Seite zwar etwas Struktur in das Modell, auf der anderen Seite werden die Namen jedoch sehr lang und kompliziert. Dadurch können sie zwar bei der Erstellung von SDI-Ontologien unterstützen, erschweren aber das Erstellen von Abfragen durch Domänenexperten. Auch ist die Verknüpfung von Klasseneigenschaften mit mehreren Spalten eines Quelldaten-Schemas nur bedingt möglich und die Art der Verknüpfung ist nicht konfigurierbar. Aufgrund dieser Nachteile ist die Verwendung von Anchor Modeling für Ontologien in SDI nicht zu empfehlen, da viel Aufwand betrieben werden muss, um die Nachteile zu kompensieren.

Kapitel 6

Diskussion und Ausblick

In den Kapiteln 3 und 4 wurde die Erstellung eines Datenschemas mit Anchor Modeling sowie dessen Umsetzung und Übertragung in SDI beschrieben. Es wurde aufgezeigt, dass die Umsetzung eines Anchor-Model-basierten Schemas in SDI zwar möglich ist, aber einigen Einschränkungen unterliegt. Dabei wurde die Umsetzung beispielhaft an einem Anwendungsfall durchgeführt, welcher die Verknüpfung von Datenquellen für die Rückverfolgbarkeit von Produkten, betrachtet.

Das Schema, welches durch diesen Anwendungsfall entstanden ist, enthält alle Komponenten, die für ein Anchor Model benötigt werden. Dadurch konnte die Umsetzung aller Komponenten in SDI untersucht werden. Das Beispiel wies auch unterschiedliche Datenquellen auf, die integriert werden mussten und über eine Schnittstelle bereitgestellt werden sollen. Diese Anforderungen gleichen sich mit den Anforderungen an die Datenintegration. Daher ist der beispielhafte Anwendungsfall gut geeignet, um die Umsetzung zu untersuchen. Jedoch gab es, während der Umsetzung, keine Evolution des Datenschemas, wodurch die Erweiterung des Datenschemas nicht direkt untersucht werden konnte.

Die SDI-Ontologien wurden während der Bearbeitung dieser Arbeit als logische Datenmodelle betrachtet. Diese Datenmodelle nutzen Klassen und Eigenschaften, ähnlich wie in der Objekt-Orientierung, um Daten zu modellieren. Sie wurden als Datenmodell über den Quelldaten-Schemata betrachtet. Dadurch entsprechen die SDI-Ontologien nicht den Ontologien, wie sie sonst in der Informatik verwendet werden (siehe Kapitel 2). Eine andere Betrachtungsweise für die SDI-Ontologien ist es, diese als eine Art Beschreibung der Nutzdaten zu betrachten. Diese Beschreibung kann dann genutzt werden, um die Abfragen zu übersetzen und auf den Quelldaten-Schemata auszuführen. Dadurch werden die SDI-Ontologien zu einer Art Metadaten, die die Nutzdaten beschreiben. Dies ähnelt der Verwendung von Ontologien in der Informatik und kann, bei weiteren Untersuchungen, neue Möglichkeiten und Erkenntnisse für die Umsetzung und Verwendung von SDI-Ontologien bieten.

Des Weiteren stellt sich die Frage, ob sich der Aufwand lohnt, ein solches Datenmodell in SDI umzusetzen. Die Motivation ein einheitliches Datenmodell zu verwenden, besteht darin, dass unterschiedliche Anwender und Nutzer die Daten verwenden können.

Durch die Definition einheitlicher Begriffe, die in unterschiedlichen Anwendungen genutzt werden, können Kommunikationsprobleme vermieden werden (vgl. [17]). In Kapitel 4 wurde erläutert, wie Rohdaten aufgebaut sein können, um mit einem Anchor-Model-basierten Schema in SDI umgesetzt zu werden. Dafür sollen die Rohdaten bereits in Form des Anchor Models vorliegen, können aber auch weniger normalisiert sein. Ist das der Fall, kann das Datenmodell in SDI umgesetzt werden. Die Rohdaten werden dafür weiter normalisiert nur um sie im nächsten Schritt, bei den Abfragen, wieder zu denormalisieren. Zusätzlich wird dafür Vorwissen im Umgang mit JOIN Operationen und der Funktionsweise von Anchor Modeling benötigt. Dieses Vorgehen erscheint nicht sinnvoll, da die Rohdaten nur denormalisiert werden, um Anchor Modeling zu nutzen.

Eventuell ist es sinnvoll, die Rohdaten in einem Datenmodell vorliegen zu haben, welches sich eher an die dritte Normalform anlehnt. Diese ist einfach zu verstehen und bietet Vorteile um Anomalien zu vermeiden. Aufgrund dieser Vorteile werden Datenbanken oft in der dritten Normalform erstellt (vgl. [41, S.52]). Neben dem besseren Verständnis werden ebenfalls mögliche Probleme bei der Ausführung von Abfragen (siehe Kapitel 4.2.3) verringert, da weniger komplexe Abfragen benötigt werden. Zudem werden, zumindest nicht für die allgemeine Nutzung, weniger Views benötigt, die in SDI nicht direkt unterstützt werden. Zusammengefasst können mit einem Modell in dritter Normalform dieselben Sachverhalte bei leichterem Verständnis dargestellt werden.

Ein weiterer Ansatz stellt die Strukturierung der Rohdaten, mithilfe von Anchor Models, dar. Dadurch können bei der Erstellung und Erweiterung der Datenquellen mögliche Anomalien vermieden werden. Die Datenquellen können, je nach Anforderung, leicht erweitert und verändert und mit anderen Datenquellen verknüpft werden. Dafür kann die Datengewinnung umstrukturiert werden, um die Rohdaten direkt in einer Form zu erhalten, die sich für Anchor Modeling eignet. Andererseits ist es auch möglich die Rohdaten, bei Verwendung, durch eine Pipeline in die gewünschte Form zu überführen, bevor sie in SDI geladen werden. Da die Umstrukturierung der Datengewinnung oft nicht möglich ist, bietet sich eher der Einsatz einer Pipeline an. Diese kann dann, je nach Anwendungsfall, implementiert und angepasst werden, sodass die Rohdaten in der gewünschten Form vorliegen. Die transformierten Rohdaten müssen dann in einem Datenspeicher abgelegt werden. Hierzu findet sich bereits hinreichend Literatur (u. a. [3], [42]), bietet aber auch die Möglichkeit für eine weiterführende Forschung, insbesondere mit Rohdaten in der Form von Anchor Models. Diese normalisierten Datenquellen können in SDI mithilfe der Ontologien denormalisiert werden, um sie für Anwender verständlich anzubieten. Weitere mögliche Forschungsgebiete bieten sich in der Herstellung, Verarbeitung und Speicherung von Rohdaten an.

Kapitel 7

Ergebnis

In dieser Arbeit wurde das Thema der Datenintegration mittels SDI und Anchor Modeling behandelt. Hierzu wurde überprüft, ob mithilfe von Anchor Modeling ein Datenschema erstellt werden kann, welches für die Integration heterogener Datenquellen mit SDI geeignet ist. Um diese Frage zu beantworten wurde, anhand eines Anwendungsfalles, ein Datenschema mit Anchor Modeling erstellt, welches verschiedene Datenquellen integriert. Bei den Datenquellen handelt es sich um Genealogie-Daten. Das sind Daten, die genutzt werden, um Produkte rückverfolgen zu können. Diese Genealogie-Daten wurden mit Produktions- und Auftragsdaten verknüpft, sodass sie für weiter Analysen und Anwendungen genutzt werden können. Dieses Datenschema wurde so aufgebaut, dass es verständlich, erweiterbar und stabil ist.

Es wurde aufgezeigt, dass die Umsetzung von Anchor Modeling in SDI nicht uneingeschränkt möglich ist. Es gibt viele Funktionalitäten und Konzepte, die Anchor Model Implementierungen voraussetzen, jedoch nicht von SDI unterstützt werden. Darunter fällt die Verwendung und Generierung von Schlüsseln, die Nutzung von Views zur Denormalisierung der Anchor und Attributes und das Erstellen von Knots, die nicht direkt mit SDI erstellt werden können. Nicht nur speziell bei der Verwendung von Anchor Modeling, sondern auch allgemein sind manche Funktionen von SDI nicht so weit ausgereift oder vorhanden, dass es für eine benutzerfreundliche Datenintegration und -bereitstellung genutzt werden kann. Die Einschränkung der Verknüpfung zwischen Datenquellen sowie die aufwendige Bearbeitung von Ontologien sind zwei Beispiele für nicht ausgereifte Funktionen.

Es gibt Möglichkeiten SDI für die Datenintegration zu nutzen und eine SDI-Ontologie basierend auf Anchor Modeling zu erstellen. Liegen die Nutzdaten bereits in einer Form vor, die einem Anchor Model entspricht oder ähnelt, ist es möglich das Modell in SDI umzusetzen. Dafür müssen die Daten nicht in der sechsten Normalform vorliegen, sollten aber die Struktur des Modells bereits abbilden. Indem man verschiedene Anchor mit ihren Attributen in einer Datenquelle vereint werden, kann die Normalisierung und Anzahl der benötigten Datenquellen reduziert werden.

Weiterhin besteht die Frage, ob eine solch starke Normalisierung auf einer hohen logischen Ebene, überhaupt sinnvoll ist. Bei der Diskussion in Kapitel 6 wurde darauf

hingewiesen, dass eine andere Herangehensweisen, möglicherweise bessere Ergebnisse liefern könnte. Hierfür müssen zusätzliche Vorkehrungen bei der Datenbeschaffung und -vorbereitung getroffen werden, um die Datenqualität und -struktur zu gewährleisten. Dies kann als Grundlage für weitere Forschung genutzt werden.

Anchor Modeling ist ein interessanter Ansatz für die Datenmodellierung, ist aber als übersichtliche Schnittstelle von Nutzdaten ungeeignet. Durch die hohe Normalisierung entstehen viele Tabellen, die für den allgemeinen Nutzer unübersichtlich sein können. Aufgrund seiner positiven Eigenschaften, wie die Stabilität bei Erweiterungen des Datenmodells, sowie die Möglichkeit der Historisierung von Daten, bietet dieser Ansatz vielversprechende Möglichkeiten für die Datenmodellierung.

Das Nutzen von Anchor Modeling im Zusammenhang mit SDI für die Datenintegration ist, zum Zeitpunkt dieser Arbeit, möglich. Aufgrund der Einschränkungen von SDI und der Komplexität von Anchor Models ist es allerdings nicht empfehlenswert. SDI ermöglicht nur bedingt die Verknüpfung von Datenquellen, die Bearbeitung von Ontologien ist aufwendig und bietet auch keine Möglichkeiten für eine Visualisierung. Anchor Models erzeugen, durch ihre hohe Normalisierung, viele Tabellen, die für den Nutzer unübersichtlich sein können. Zudem ist die Namensgebung nicht für Anwender der Nutzdaten geeignet, da diese eher das Modell beschreiben, als die Daten selbst. Das Zusammenfügen der Tabellen erfordert viel Wissen über die Daten und das Modell, was die Nutzung weiter erschwert. Liegen die Rohdaten nicht in einer Anchor-Model-basierten Form vor, erschwert das die Verknüpfung der Datenquellen-Schemata mit dem SDI-Ontologie-Schema. Werden die Nutzdaten in eine solche Anchor-Model-basierte Form transformiert, wird die Nutzung und Umsetzung in SDI erleichtert. Ob die Verwendung den Aufwand für die Datenvorbereitung und -integration rechtfertigt, muss von jedem Nutzer selbst entschieden werden. Sollte SDI seine Funktionalitäten erweitern und verbessern, kann es allerdings als ein mögliches Werkzeug für die Datenintegration genutzt werden.

Literatur

[1] Hani Abdeen, Osama Shata und Abdelkarim Erradi. „Software interfaces: On the impact of interface design anomalies“. In: *International Conference on Computer Science and Information Technology*. 5. IEEE, 2013, S. 184–193.

[2] *About Anchor Modeling*. URL: https://www.anchormodeling.com/about/.

[3] Andreas Bauer und Holger Günzel. *Data-Warehouse-Systeme: Architektur, Entwicklung, Anwendung*. dpunkt. verlag, 2013.

[4] Alex Bekker. *The 'Scary' Seven: big data challenges and ways to solve them*. https://www.scnsoft.com/blog/big-data-challenges-and-their-solutions. März 2018.

[5] T. Berners-Lee u. a. *Uniform Resource Identifier (URI): Generic Syntax*. https://www.rfc-editor.org/rfc/rfc3986. Jan. 2005.

[6] Mario Bunge und Martin Mahner. *Über die Dinge der Natur - Materialismus und Wissenschaft*. 2. Aufl. S. Hirzel Verlag, 2021.

[7] Coral Calero u. a. „Towards Data Warehouse Quality Metrics.“ In: *Design and Management of Data Warehouses*. Bd. 39. Juni 2001, S. 2.1–2.8.

[8] C.J. Date, H. Darwen und N. Lorentzos. *Temporal data & the relational model*. Morgan Kaufmann Publishers, 2002.

[9] AnHai Doan, Alon Halevy und Zachary Ives. *Principles of Data Integration*. Elsevier, Juli 2012.

[10] Stephan Dreistel. *Mathematik für Software Engineering*. Springer Vieweg, 2018.

[11] Dremio. *Introduction to Data Engineering*. https://www.dremio.com/resources/guides/intro-data-engineering/.

[12] Duden. *Genealogie*. https://www.duden.de/rechtschreibung/Genealogie.

[13] Duden. *Semantik*. https://www.duden.de/rechtschreibung/Semantik.

[14] Thomas Filbry u. a. *Datenintegration - Integrationsansätze, Beispielszenarien, Problemlösungen, Talend Open Studio*. Hanser Verlag, 2013.

[15] John Gantz und David Reinsel. „The digital universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East“. In: *IDC iView: IDC Analyze the future* (Dez. 2012), S. 1–16.

[16] Marcela Genero, Geert Poels und Mario Piattini. „Defining and validating metrics for assessing the understandability of entity–relationship diagrams“. In: *Data and Knowledge Engineering* 64.3 (2008), S. 534–557. ISSN: 0169-023X. DOI: https://doi.org/10.1016/j.datak.2007.09.011. URL: https://www.sciencedirect.com/science/article/pii/S0169023X07001796.

[17] Dale L. Goodhue, Micheal D. Wybo und Laurie J. Kirsch. „The Impact of Data Integration on the Cost and Benfits of Information Systems“. In: *MIS Quarterly* 16.3 (Sep. 1992), S. 293–311.

[18] Anjana Gosain und Heena. „Literature Review of Data Model Quality Metrics of Data Warehouse“. In: *Procedia Computer Science* 48 (2015). International Conference on Computer, Communication and Convergence (ICCC 2015), S. 236–243. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2015.04.176. URL: https://www.sciencedirect.com/science/article/pii/S1877050915006857.

[19] Thomas R. Gruber. „A translation approach to portable ontology specifications“. In: *Knowledge Acquistion* 5 (1993), S. 199–220.

[20] Do Hong Hai. „Schema Matching and Mapping-Based Data Integration“. Diss. University of Leipzig, Aug. 2005.

[21] Pascal Hitzler u. a. *OWL 2 web ontology language primer*. https://www.w3.org/TR/owl2-primer/. Dez. 2012.

[22] *Insights Hub*. https://plm.sw.siemens.com/en-US/insights-hub/.

[23] *Introducing JSON*. https://www.json.org/json-en.html.

[24] Matthias Jarke und Jürgen Koch. „Query Optimization in Database Systems“. In: *Computing Surveys* 16.2 (Juni 1984), S. 111–144.

[25] Sadhana J. Kamatkar u. a. „Database Performance Tuning and Query Optimization“. In: *Data Mining and Big Data*. International Conference on Data Mining and Big Data. Juni 2018, S. 3–11.

[26] Frank Manola und Eric Miller. *RDF Primer*. https://www.w3.org/TR/2004/REC-rdf-primer-20040210/. Feb. 2004.

[27] Deborah L. McGuinness und Frank van Hermelen. *OWL Web Ontology Language Overview*. https://www.w3.org/TR/2004/REC-owl-features-20040210/. Feb. 2004.

[28] Irene Mikhailouskaya. *Your Guide to Data Quality Management*. Dez. 2018. URL: https://www.scnsoft.com/blog/guide-to-data-quality-management.

[29] Natalia Miloslavskaya und Alexander Tolstoy. „Big Data, Fast Data and Data Lake Concepts“. In: *International Conference on Biologically Inspired Cognitive Architectures* 88.7 (Juli 2016), S. 300–305.

[30] Harry Mucksch. *Das Data Warehouse Konzept - Architektur - Datenmodelle - Anwendungen*. Bd. 4. Springer, 2000.

[31] Dan Myers. *List of Conformed Dimensions of Data Quality*. https://dimensionsofdataquality.com/alldimensions. 2019.

[32] M Pamela Neely. „The Product approach to data quality and fitness for use: a Framework for analysis“. In: *Proceedings of the 2005 International Conference on Information Quality (MIT ICIQ Conference)*. Proceedings of the Tenth International Conference on Information Quality (ICIQ-05). Rochester Institute of Technology. 2005.

[33] Natalya F. Noy und Deborah L. McGuinness. „Ontology Development 101: AGuide to Creating Your First Ontology“. Magisterarb. Stanford University, 2001.

[34] OpenJS Foundation & Contributors. *About Node-RED*. https://nodered.org/about/.

[35] *Parquet Overview*. https://parquet.apache.org/docs/overview/. Macrh 2022.
[36] *Polars*. https://www.pola.rs/.
[37] Ghadafi M. Razak, Linda C. Hendry und Mark Stevenson. „Supply chain traceability: a review of the benefits and its relationship with supply chain resilience“. In: *Production Planning & Control* 34.11 (Okt. 2023), S. 1114–1134. DOI: 10.1080/09537287.2021.1983661. eprint: https://doi.org/10.1080/09537287.2021.1983661. URL: https://doi.org/10.1080/09537287.2021.1983661.
[38] Olle Regardt u. a. „Anchor Modeling.“ In: *International Conference on Conceptual Modeling*. Nov. 2009, S. 234–250. ISBN: 978-3-642-04839-5. DOI: 10.1007/978-3-642-04840-1_19.
[39] Malte Rehbein. „Ontologien“. In: *Digital Humanities: Eine Einführung*. Hrsg. von Fotis Jannidis, Hubertus Kohle und Malte Rehbein. Stuttgart: J.B. Metzler, 2017, S. 162–176. ISBN: 978-3-476-05446-3. DOI: 10.1007/978-3-476-05446-3_11. URL: https://doi.org/10.1007/978-3-476-05446-3_11.
[40] Lars Rönnbäck u. a. „Anchor modeling - Agile information modeling in evolving data envoirements“. In: *Data & Knowledge Engineering*. Hrsg. von Alberto H.F Laender, Silvana Castano und Umeshwar Dayal. Bd. 69. 12. Dez. 2010, S. 1230–1253.
[41] Edwin Schicker. *Datenbanken und SQL*. 5. Aufl. Informatik & Praxis. Springer Vieweg, 2017.
[42] Prachi Shah u. a. „IoT-Based Big Data Storage Systems in Cloud Computing“. In: *Proceedings of Second International Conference on Smart Energy and Communication*. 2. Springer, Jan. 2021, S. 323–333.
[43] John Miles Smith u. a. „Multibase: integrating heterogeneous distributed database systems“. In: *FIPS '81: Proceedings of the May 4-7, 1981, national computer conference*. Bd. 50. Chicago, Illinois: Association for Computing Machinery, Mai 1981, S. 487–499.
[44] H. Stuckenschmidt. *Ontologien - Konzpte, Technologien und Anwendungen*. Springer, 2009.
[45] Tutanch und Nico Litzel. *Was ist ein Data Scientist?* https://www.bigdata-insider.de/was-ist-ein-data-scientist-a-600907/. Apr. 2017.
[46] W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. https://www.w3.org/TR/2012/REC-owl2-overview-20121211/. Dez. 2012.
[47] *What is Tableau*. https://www.tableau.com/why-tableau/what-is-tableau.
[48] Jihong Yan u. a. „Industrial Big Data in an Industry 4.0 Environment: Challenges, Schemes, and Applications for Predictive Maintenance“. In: *IEEE Access* 5 (Okt. 2017), S. 23484–23491.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang A

Schema der SAP-Daten

SAP	
id	*str*
afko_aufnr	*str*
afko_gltri	*int*
afko_plnbez	*str*
afpo_kdauf	*str*
afpo_kdpos	*str*
resb_posnr	*str*
resb_potx1	*str*
resb_matnr	*str*
resb_bdmng	*str*
resb_meins	*str*
makt_maktx	*str*
afpo_dwerk	*str*
afko_fevor	*str*
vbap_yybcvfgrp	*str*
vbap_yybcezndr	*str*
vbap_yykurzanga	*str*
afko_getri_timestamp	*str*
afko_lead_aufnr	*str*
afko_zzfabnr	*str*

Tabelle A.1: Schema der SAP-Daten

Anhang B

Testschema als Anchor Model

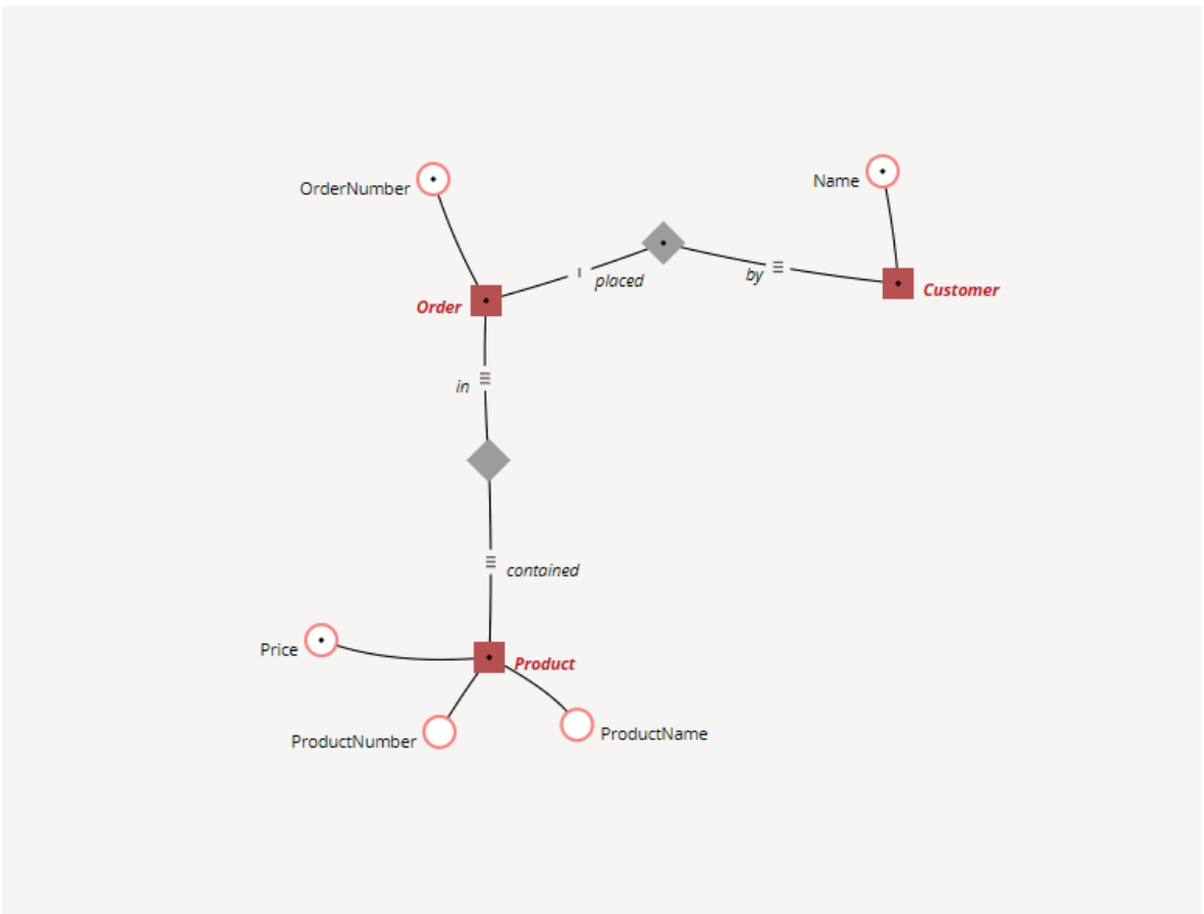

Abbildung B.1: Anchor Model Testschema

Anhang C

Schema der Testdaten

Products	
ProductName	*str*
ProductNumber	*str*
Price	*float*
PrimaryKey	*int*

Tabelle C.1: Schema der Produkt-Testdaten

Orders	
OrderNumber	*str*
CustomerNumber	*int*
PrimaryKey	*int*

Tabelle C.2: Schema der Auftrags-Testdaten

Customer	
Name	*str*
PrimaryKey	*int*

Tabelle C.3: Schema der Kunden-Testdaten

Order-Product	
order_id	*int*
product_id	*int*

Tabelle C.4: Schema der Order-Product-Testdaten (Beziehungstabelle)