Evaluierung einer Datenmigrationssoftware hinsichtlich des TPC-DI-Benchmarks

Bachelorarbeit

Christian Tessnow (212204598)

Universität Rostock

Fakultät für Informatik und Elektrotechnik

Institut für Informatik

Betreuer und 1. Gutachter: Prof. Dr. rer. nat. habil. Andreas Heuer

2. Gutachter: Dr.-Ing. Holger Meyer

Abgabe: 21. März 2016

Inhaltsverzeichnis

Abstract

Die Datenintegration ist ein zeit- und rechenaufwändiger Prozess. Um diesen zu vereinfachen, wurden diverse Programme entwickelt. Da aber alle Tools unterschiedliche Funktionen bieten, kann schwer eingeschätzt werden, wie gut diese für die eigenen Anforderungen geeignet sind. Um einen Vergleich zwischen den Programmen zu ermöglichen, wurde unter anderem der TPC-DI-Benchmark entwickelt. Mit diesem werden in dieser Arbeit zwei Datenintegrationslösungen untersucht und miteinander verglichen. Zusätzlich wird auch ein Blick auf die Anforderungen der Forschung geworfen.
Ziel der Arbeit ist es beim Datenmigrationstool Tensei-Data Schwächen ausfindig zu machen und eine davon im Anschluss zu beheben.

1 Einleitung

Diese Arbeit untersucht die Datenmigrationssoftware Tensei-Data auf Korrektheit, Vollständigkeit und Leistungsfähigkeit. Um diese Kriterien beurteilen zu können, wird die Analyse mit Hilfe des TPC-DI-Benchmarks vorgenommen. Zudem werden auch die Anforderungen, die die Forschung an Datenintegrations- und Datenmigrationslösungen stellt, nicht außer Acht gelassen. Aus der Menge der nicht erfüllten Anforderungen wird dann eine Anforderung ausgewählt und Tensei-Data dahingehend erweitert, so dass diese erfüllt wird. Zusätzlich wird die Software mit einer weiteren, lang bewährten Migrationssoftware verglichen.
Zunächst wird in Abschnitt 2 das Thema motiviert, ehe im darauffolgenden Abschnitt die Grundbegriffe geklärt werden. Der vierte Abschnitt gibt einen Überblick über die Kernkomponenten von Datenintegrationswerkzeugen und stellt einige Benchmarks und Integrationstools vor. Abschnitt 5 definiert anschließend die Anforderungen aus der Forschung an Datenintegrationssoftware. In Abschnitt 6 wird kurz Tensei-Data vorgestellt. Abschnitt 7 beschäftigt sich dann mit dem Aufbau und den Anforderungen des TPC-DI-Benchmarks. Anschließend wird im Abschnitt 8 die Analyse von Tensei-Data vorgenommen. In Abschnitt 9 erfolgt dann der Vergleich mit Talend Open Studio. Abschließend wird in Abschnitt 10 die implementierte Erweiterung vorgestellt und in Abschnitt 11 ein Ausblick auf zukünftige Arbeiten gegeben. Abschnitt 12 resümiert abschließend diese Arbeit.

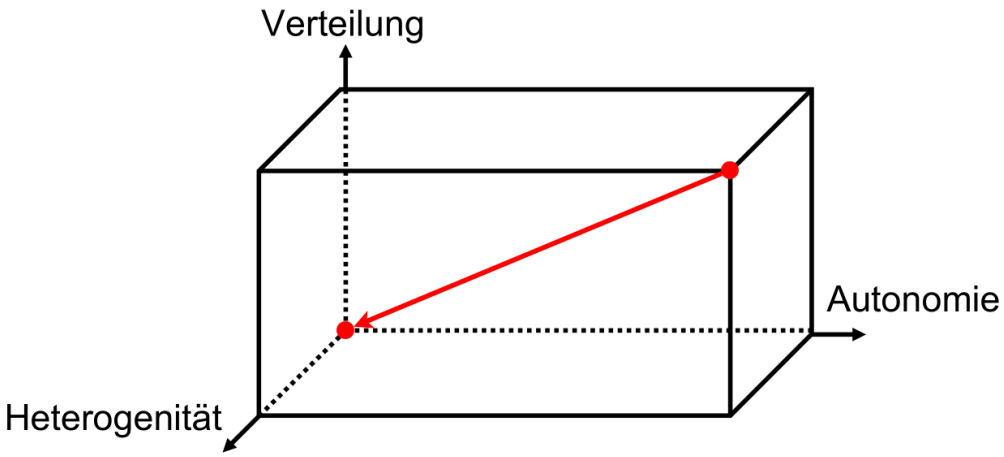

Abbildung 1: Darstellung der Datenmigration (in Anlehnung an [OV99])

2 Motivation

Als Teilgebiet der Datenintegration dient die Datenmigration dazu, bestehende Altsysteme durch Neue zu ersetzen. Klassische Anwendungsfälle sind dabei die Migration von Daten aus Textdateien in eine Datenbank oder die Migration aus mehreren Datenbanken in eine Datenbank. Ebenso wird die Datenintegration beim Aufbau von Data Warehouses genutzt, mit dem Unterschied, dass hier keine Altsysteme abgelöst werden sollen.
Um eine Migration durchführen zu können, ist es notwendig, die wahrscheinlich heterogenen, autonomen und vielleicht verteilten Daten in ein gemeinsames Schema zu transformieren und dann in das Zielsystem zu schreiben. Das Zielsystem zeichnet sich dabei durch Homogenität und Heteronomie (Gegenteil von Autonomie) aus. Zudem ist das System dann nicht mehr logisch verteilt. Eine Visualisierung dieser Transformation ist in Abbildung 1 zu sehen.
Zur Realisierung dieser Transformation wurden verschiedene Datenmigrationslösungen entwickelt, die aber alle unterschiedlich funktionieren und ihre eigenen Stärken und Schwächen haben. Eines dieser Werkzeuge, mit dem diese Aufgabe umgesetzt werden soll, ist Tensei-Data. Die Software soll dem Nutzer eine schnelle und einfache Migration erlauben und durch teilautomatisierte Prozesse viel Arbeit abnehmen.

Um die verschiedenen Integrationslösungen vergleichen zu können, wurden auf Grundlage der Kernanforderungen an diese Werkzeuge Benchmarks entwickelt, mit dem die Leistungsfähigkeit, Korrektheit und Vollständigkeit der Programme geprüft werden können.
Die große Bedeutung der Datenintegration zeigt sich auch in der über 20-jährigen Forschung zu diesem Thema. Während einige Anforderungen an die Datenmigration schon zu Beginn der Forschung gestellt worden sind, gibt es auch Anforderungen, wie die automatische Fremdschlüsselerkennung, die erst im letzten Jahr erforscht wurden.
Auch in der Wirtschaft wird das Thema immer wichtiger, da im Zuge der Globalisierung immer mehr Firmen fusionieren oder übernommen werden und dabei auch die IT-Systeme zusammengeführt werden.

3 Begriffsabgrenzung

Der Begriff Datenintegration ist in der Literatur sehr unterschiedlich definiert. In dieser Arbeit wird daher die Definition von [LN07] genutzt.
Hier wird die Datenintegration als das „korrekte, vollständige und effiziente Zusammenführen von Daten und Inhalt verschiedener, heterogener Quellen zu einer einheitlichen und strukturierten Informationsmenge zur effektiven Interpretation durch Nutzer und Anwendungen“ definiert. Ziel der Datenintegration soll es sein, aus den kombinierten Daten einen Mehrwert zu schöpfen. Dabei werden verschiedene Techniken unterschieden.

3.1 Materialisierte Datenintegration

Mit der materialisierten Datenintegration wird das „Kopieren“ der Daten von mehreren Quellensystemen in ein Zielsystem beschrieben. Das Ziel ist es dabei, die alten Quellsysteme abzulösen und das Zielsystem weiterzuführen. Bei der Realisierung sind dabei insbesondere das Schema Matching und das Schema Mapping wichtig.

3.2 Datenmigration

Die Datenmigration beschreibt grundlegend den gleichen Prozess wie die materialisierte Datenintegration. Einziger Unterschied ist, dass hier üblicherweise nur ein Quellsystem migriert wird und damit das Schema Matching entfällt. Die Begriffe Datenmigration und materialisierte Datenintegration werden in dieser Arbeit daher synonym verwendet.

3.3 Datenföderation

Bei der Datenföderation oder auch virtuellen Datenintegration bleiben alle Daten im Quellsystem. Diese werden dann vom Zielsystem angefragt. Der Anwender stellt somit seine Anfragen gegen das Zielsystem, welches die Anfragen in kleine Anfragen an das jeweilige Quellsystem zerlegt und die Ergebnisse danach zusammenführt. Das Quellsystem wird zusätzlich noch durch alte Anwendungsprogramme und für operative Prozesse genutzt.

3.4 Datenfusion

Ein Spezialfall der materialisierten Datenintegration ist die Datenfusion. Hierbei werden die Quelldaten nicht direkt übernommen, sondern zu einem neuen Wert zusammengefasst und dann in das Zielsystem geschrieben. Das ist insbesondere wichtig, wenn die Daten dieselben Objekte repräsentieren, aber unterschiedliche Werte für sie besitzen. Der neue Wert muss daher nicht zwangsweise in den Quelldaten auftauchen, stattdessen kann dieser auch mit einer Funktion berechnet werden.

3.5 Schema-Evolution

Als letztes Konzept sei hier die Schema-Evolution genannt. Dabei ist nur das Quellsystem bekannt. Dieses wird schrittweise durch Evolutionsoperationen verändert. Bei jeder Operation wird dann die Datenbank automatisch in die neue Struktur überführt. Dadurch ist das Schema Mapping nicht mehr nötig.

3.6 Fazit

Es gibt verschiedene Verfahren um die Datenintegration durchzuführen. Da Tensei-Data das Zielsystem mit Daten befüllt und das Altsystem nach der Integration abgelöst werden soll, werden im Folgenden nur noch die materialisierte Datenintegration und die Datenmigration betrachtet.

4 Stand der Technik

Da alle Datenintegrationswerkzeuge eine ähnliche Aufgabe haben, zeichnen sie sich auch alle durch ähnliche Komponenten aus. Diese werden im Folgenden vorgestellt.
Des Weiteren werden andere Benchmarks und Datenintegrationsprogramme benannt.

4.1 Komponenten von Integrationswerkzeugen

Schema Matching

Der Schema Matcher soll in heterogenen Quellsystemen nach zueinander passenden Informationseinheiten suchen. Die Suche kann dabei anhand der Attributnamen, der Datenstruktur oder auf Grundlage der Daten geschehen.

Schema Mapping

Nach dem Schema Matching folgt das Schema Mapping. Dabei werden die Datenelemente aus den Quellsystemen auf die Zielelemente gemappt, um die Beziehungen zwischen den beiden System zu beschreiben. Das Schema Mapping kann auch automatisiert auf Basis der Attributnamen oder der Datenstruktur erfolgen. Unter Umständen muss daraus auch die nötige Datentransformation abgeleitet werden. Bei relationalen Datenbanken geschieht das in der Regel mit SQL-Ausdrücken.

Transformation

Da die Strukturen von Quell- und Zielsystem zumeist unterschiedlich sind, müssen die Daten während der Migration angepasst werden. Die Daten können dabei zusammengefasst oder auch zerteilt werden.

Abbildung 2: Bewertung von Datenintegrationswerkzeugen nach Gartner (Juli 2015)

Anfrageplanung

Im Bereich der Datenföderation wird auch ein Modul zur Anfrageplanung benötigt. Dieses soll die Anfrage an das globale Schema in Anfragen an die jeweiligen Quellsysteme zerteilen und nach deren Bearbeitung, die Ergebnisse zusammenführen.

4.2 Integrationstools

All diese Komponenten werden sich in einer bestimmten Form in fast jeder Integrationssoftware wiederfinden. Welche Lösungen es konkret gibt, wird nun betrachtet.
Da das Problem der Datenintegration schon lange bekannt ist, gibt es auch dementsprechend viele Lösungen dafür. So listet der jährliche Bericht von Gartner ([TR15]) zum Thema Datenintegration im Jahr 2015 13 Integrationslösungen auf (Abbildung 2). [Rev15] benennt sogar 24 Tools.
Das beste Tool laut Gartner ist dabei die Informatica Data Integration Suite, gefolgt von IBMs Lösung InfoSphere Information Server. Weitere Softwarelösungen sind die Microsoft SQL Server Integration Services (SSIS) und der Oracle Data Integrator.
Auch deutsche Hersteller haben Datenintegrationswerkzeuge entwickelt. So hat SAP eine ganze Reihe von Integrationslösungen entwickelt, die miteinander kombiniert werden können. Ebenso bietet die Software AG eine Suite für Datenintegration an.
Alle genannten Programme richten sich vor allem an größere Unternehmen. Als einziges kostenfreies Tool ist Talend Open Studio auf dem Markt, wodurch es sich wie Tensei-Data auch für kleine bis mittlere Unternehmen eignet. Beide Programme sind zudem nicht auf einige wenige Informationssysteme spezialisiert, sondern können bei einer Vielzahl verschiedener Quell- und Zielsysteme eingesetzt werden.

4.3 Benchmarks

Damit diese Tools verglichen werden können, wurden verschiedene Benchmarks entwickelt. Neben dem TPC-DI-Benchmark, der in Abschnitt 7 genauer vorgestellt wird, gibt es noch einige andere Benchmarks, die nun betrachtet werden.

iBench

Im Jahr 2012 wurde der iBench-Benchmark an der Universität Toronto entwickelt. Mit diesem Benchmark können verschiedenste Teilgebiete der Datenintegration nachgebildet und getestet werden. Dazu gehören die virtuelle Datenintegration, Datenaustausch, Schema-Evolution, Schema Matching und Mappingoperationen wie Komposition und Inversion [AGCM15].
Die Komposition beschreibt das Zusammenfassen mehrerer hintereinander ausgeführter Evolutionsoperationen bei der Schema-Evolution. Die Inversion hingegen beschreibt das Zurücksetzen des Systems auf den ursprünglichen Stand durch Umkehren der Evolutionsoperation, mit dem Ziel das Anfangsschema dann ändern zu können. Nach der Änderung kann dann durch Komposition wieder das Zielschema hergestellt werden [FKPT11].
Dieser Benchmark ist sehr vielseitig und umfasst mehrere Gebiete der Datenintegration. Für diese Arbeit ist er jedoch ungeeignet, da es hauptsächlich um Datenföderation und Schema-Evolution geht.

Amalgam

Ein weiterer Benchmark ist der Amalgam-Benchmark. Dieser wurde schon im Jahr 2001 ebenfalls an der Universität von Toronto entwickelt. Bei diesem Benchmark geht es darum, bibliografische Daten aus einer Internetquelle in eine lokale, relationale Datenbank zu schreiben [MFH+01].
Dieser Benchmark ist für diese Arbeit auch nicht geeignet, da er neben der Migration auf die Erstellung des Zielschemas umfasst. Zudem können die Daten aus der Internetquelle sehr unstrukturiert sein. Ebenso können auch mehrere Quellen genutzt werden, wodurch der Einsatz eines Schema Matchers unerlässlich wäre.

Linked Open Data Integration Benchmark

Ebenfalls im Jahr 2012 hat ein spanisch-deutsches Forscherteam der FU Berlin und der Universität in Sevilla den Linked Open Data Integration Benchmark (LODIB) entwickelt. Bei diesem Benchmark geht es darum, Daten aus einem bestimmten Anwendungsbereich mithilfe von Semantik, Ontologien und einem überschaubaren Vokabular zusammenzuführen [RSB12].
Das Anwendungsgebiet dieses Benchmarks ist sehr speziell und kommt daher auch nicht in Frage.

Data Intensive Integration Process Benchmark

Als letzter Benchmark ist der Data Intensive Integration Process Benchmark (DIPBench) zu nennen. Dieser wurde 2008 an den TU Dresden und FH Dresden entwickelt. Dieser befasst sich, wie der Name schon sagt, mit datenintensiven Integrationsprozessen. Datenintensiv heißt in diesem Fall, dass viele Daten migriert und dabei auch transformiert werden sollen. Bei der Entwicklung wurde im Speziellen auf die praktische Relevanz geachtet, womit dieser Benchmark dem TPC-DI-Benchmark sehr nahe kommt [BHLW08].
Der DIPBench wäre auch sehr gut für diese Arbeit geeignet, da er sehr praxisnahe ist und viele Aspekte der Migration umfasst. Da der TPC-DI-Benchmark aber de facto Industriestandard ist und die Ergebnisse von der TPC publiziert werden können, ist die Wahl des Benchmarks nicht auf den DIPBench gefallen.

5 Stand der Forschung

In diesem Abschnitt geht es um die Anforderungen, die die Forschung an Datenmigrations- oder Datenintegrationssoftware stellt. Zum Teil werden kurz Lösungen für diese Probleme vorgestellt.

5.1 Komplementierung

Die Komplementierung ist die Kernaufgabe jedes Datenintegrationswerkzeugs. Dadurch soll sichergestellt werden, dass alle zusammengehörige Daten, die über die verschiedenen Quellsysteme verteilt sind, im Zielsystem vereint werden [LN07].

5.2 Aktualität

Damit das Altsystem nach der erfolgreichen Migration abgelöst werden kann, müssen alle Daten in das Zielsystem übertragen werden, so dass es dann auf dem aktuellen Stand. Zusätzlicher müssen die Anwendungsprogramme, die mit den Daten arbeiten, an das neue Informationssystem angepasst werden. Wenn das nicht beachtet werden, können zwei divergierende Datenbestände enstehen, die auch widersprüchliche Daten enthalten können.
Beim Befüllen eines Data Warehouses ist es wichtig regelmäßig den veränderten Datenbestand zu migrieren, andernfalls kann hier nicht der aktuelle Datenbestand analysiert werden [LN07].

5.3 Reduzierung der Anfragekomplexität

Das Ziel der Datenmigration soll es sein, die Komplexität von Anfragen an das System zu reduzieren. Durch das Zusammenführen von Daten in einem System müssen die Anfragen nicht mehr in „kleine“ Unterfragen für jede einzelne Quelle unterteilt werden, sondern können direkt an das System gestellt werden. Dadurch ist es auch möglich komplexere Anfragen zu stellen, die vorher nicht möglich waren [LN07].

5.4 Antwortzeit

Auch die Antwortzeit soll sich nach der Migration verringern. Mit der reduzierten Anfragekomplexität (durch eventuell vereinfachter Anfragen) und das Zusammenführen der Daten in ein System wird sich auch die Zeit für die Berechnung der Ergebnisse verringern. Wenn das Zielsystem eine Datenbank ist, kann durch die Indexierung eine weitere Leistungsverbesserung vorgenommen werden [LN07].

5.5 Flexibilität

Die Flexibilität soll gewährleisten, dass ein Migrationswerkzeug nicht nur für wenige Anwendungsfälle geeignet ist, sondern bei vielen verschiedenen Szenarien eingesetzt werden kann. Daraus folgt auch, dass die Software nicht nur auf bestimmte Systeme beschränkt sein soll, sondern so vielfältig wie möglich sein soll. Gleiches gilt für den Speicherort der Daten [LN07].

5.6 Autonomie der Datenquellen

Eine weitere Anforderung an Datenintegrationslösungen (insbesondere bei der Datenföderation) ist, dass sie die Autonomie der Datenquellen nicht verletzen soll (Designautonomie). Das heißt, dass die Software weder das Datenformat der Quelldatei verändern, noch Datensätze hinzufügen, ändern oder löschen darf. Auch soll dem Quellsystem bei der Datenföderation der Zeitpunkt und die Art und Weise der Anfrageausführung überlassen werden. Weiterhin muss es möglich sein die Quelldateien auch nach der Migration weiterverwenden zu können. Bei der Datenmigration ist die Anforderung jedoch nicht so wichtig, da das Quellsystem in der Regel nicht weiterverwendet werden soll [LN07].

5.7 Datenqualität

Ebenso soll es möglich sein, die Datenqualität im Zielsystem zu erhöhen. Umsetzbar ist das durch die Erkennung und Eliminierung von Duplikaten, dem Ersetzen von fehlenden Daten (zum Beispiel durch Standard- oder Durchschnittswerte) und dem Bereinigen von falschen Daten. Zudem soll die Anreicherung von Daten aus externen Quellen möglich sein [KSS14].

5.8 Transparenz

Der Prozess der Datenmigration soll für den Endanwender der Quell- und Zielsysteme transparent erfolgen. Das bedeutet, dass sich für den Nutzer der Arbeitsablauf nicht ändern soll (Schnittstellentransparenz) und er nicht merken soll, ob er auf den Quell- oder den Zieldaten arbeitet (Quellentransparenz). Daraus ergibt sich dann auch die Ortstransparenz. Zudem sollen sich die Anfragen des Nutzer immer an ein globales, heterogenes Schema richten (Schematransparenz) [LN07].

5.9 Konsistenz

Ebenso sollen die Daten zu jedem Zeitpunkt der Migration konsistent sein. Das bedeutet, dass eventuelle Datenkonflikte, die bei der Zusammenführung der Quelldaten entstanden sind, gelöst werden sollen. Das muss nicht zwangsweise automatisch erfolgen, sondern kann auch bei der Migration als Meldung an den Anwender ausgegeben werden, so dass dieser dann die Entscheidung treffen kann [KSS14].
Genauso sollen bei der Aktualisierung von historischen Daten die alten Datensätze nicht überschrieben, sondern mittels Flag als veraltet gekennzeichnet werden. Die aktuellen Daten sollen dann wie ein neuer Datensatz in das Ziel geschrieben werden. Dabei ist es auch notwendig, den dann neuen Primärschlüssel an die abhängigen Tabellen weiterzugeben und die entsprechenden Datensätze zu aktualisieren (Aktualisierungsweitergabe) [TPC14].

5.10 Automatisches Schema Matching

Um das Mapping erstellen zu können, muss zunächst das Schema Matching durchgeführt werden. Dabei werden alle Quellsysteme auf zusammenpassende Informationseinheiten durchsucht. Zusammenpassende Einheiten können anhand von ähnlichen Attributnamen (beschriftungsbasiertes Matching) oder ähnlichen Daten (instanzbasiertes Matching) gefunden werden. In [KPN15] und [PKQN15] werden zwei Verfahren, die ein instanzbasiertes Matching ermöglichen, vorgestellt. Weiterhin kann auch anhand ähnlicher Struktureinheiten (schemabasiertes Matching) ein Schema Matching gefunden werden. Ebenso sind Mischformen der genannten Methoden möglich [LN07].

5.11 Automatisches Schema Mapping

Im Anschluss an das Schema Matching erfolgt das Schema Mapping. Da die Erstellung des Mappings bei großen Datenquellen eine zeitintensive und repetitive Tätigkeit ist, soll der Anwender möglichst gut dabei unterstützt werden. Wie beim Schema Matching kann die Zuordnung der Informationseinheiten anhand der Attributnamen oder der Struktureinheiten erfolgen. Auch hier sind Mischformen möglich. Neben der Zuordnung der Elemente soll auch der Einsatz von Transformatoren vorgeschlagen werden [HRO06, KSS14].

Abbildung 3: DFASDL von Quell- (links oben) und Zielsystem (rechts oben) und das daraus vorgeschlagenen Mapping

6 Tensei-Data

6.1 Allgemeines

Die Datenmigrationssoftware Tensei-Data wird seit 2014 von der Firma Wegtam aus Bentwisch bei Rostock entwickelt. Das Tool richtet sich vor allem an kleine und mittlere Unternehmen. Ziel ist es, die materialisiert Datenintegration schnell und einfach durchführen zu können. Dabei können (semi-)strukturierte Textdateien wie CSV, XML und JSON, als auch die Datenbanksysteme Derby, H2, HyperSQL, MariaDB, Microsoft SQL Server, MySQL, Oracle, PostgreSQL und SQLite verwendet werden.
Im Juni 2015 wurde die erste Version veröffentlicht. Zu Beginn dieser Arbeit war Version 1.4 der aktuellste Release. Alle später genannten Benchmarkergebnisse beziehen sich daher auf diese Version.
Das Programm wird innerhalb einer virtuellen Maschine ausgeliefert und kann über eine Weboberfläche bedient werden.

6.2 Vorgehensweise

Um eine Migration durchführen zu können, müssen zunächst alle Verbindungsinformationen der Quell- und Zielsysteme angegeben werden. Neben dem Dateipfad sind das auch Nutzername und Passwort für Datenbanken.
Anschließend muss zu jeder Verbindungsinformation die Datenstruktur bestimmt werden. Dies geschieht mit der XML-basierten Datenbeschreibungssprache DFASDL (Data Format and Semantics Description Language). Um die Datenstruktur zu beschreiben, stehen fast alle gängigen Datentypen wie zum Beispiel String, Integer oder Date zur Verfügung. Zusätzlich kann zu jedem Attribut die Semantik der Daten angegeben werden. Eine beispielhafte DFASDL ist in Abbildung 3 zu sehen.
Das Element *seq* definiert dabei eine sich wiederholende Struktur und das *elem*-Element kapselt alle zusammengehörenden Attribute eines Datensatzes. Innerhalb eines *elem*-Elements sind dann die einzelnen Attribute beschrieben. Mit *num*, *str* und anderen Tags wird der Datentyp des Attributes bestimmt. Das Attribut *id* gibt jedem Element eine eindeutige ID. Mit dem *stop-sign* wird in Textdateien der Begrenzer des Attributwertes bestimmt. Zusätzlich kann mit *s* die Semantik des Attributes angegeben werden. Bei Datenbanken wird das Attribut *stop-sign* nicht benötigt, stattdessen muss ein *db-column-name* angegeben werden.
Für CSV-Dateien und relationale Datenbanken können die Datenstrukturen ausgelesen und die DFASDL automatisch generiert werden.
Als nächster Schritt kann dann das Mapping definiert werden. Dazu wird ein Kochbuch, das mindestens ein Rezept enthält, erstellt. Darin wird genau beschrieben, welche Quelldaten wie transformiert und an welchen Zielort geschrieben werden sollen. Mit Hilfe der Element-ID und der Semantik kann das Programm versuchen ein einfaches Mapping ohne Transformatoren zu erstellen. Das Mapping der Quell- auf die Zielattribute erfolgt dabei anhand der gleichen Werte des Attributes *id* beziehungsweise *s* in den Datenbeschreibungen der Quell- und Zielsysteme (Abbildung 3). Sollte auf der Quell- oder Zielseite eine Semantikbeschreibung mehrfach vorkommen, dann wird nur das letzte Vorkommen gemappt. Ein Matching findet also nicht statt.

Abschließend wird mit dem Kochbuch die Transformationskonfiguration erstellt und die Migration gestartet.
Der Start der Migration kann manuell über die Weboberfläche, mithilfe eines Cronjobs oder via Trigger (zum Beispiel durch Hinzufügen einer Datei in einem Ordner) erfolgen.
Die Migration läuft dabei in zwei Phasen ab. Zunächst werden alle benötigten Daten aus dem Quellsystem gelesen und im Arbeitsspeicher abgelegt (Parsing-Phase), danach werden die Daten mit den ausgewählten Transformatoren verändert und abschließend in das Zielsystem geschrieben (Processing-Phase).

7 TPC-DI-Benchmark

7.1 Allgemeines

Der TPC-DI-Benchmark ist ein im Oktober 2013 erstmals veröffentlichter Benchmark zum Testen von Datenmigrationswerkzeugen. Die aktuelle Version stammt aus dem November 2014. Entwickelt wurde dieser von den Mitgliedern des Transaction Processing Performance Council (TPC). Mitglieder dieser Vereinigung sind unter anderem IBM, Microsoft, Oracle und SAP. Da es bis zum Entwicklungszeitpunkt keinen Industriestandard zur Messung und Vergleich der Leistung von Migrationssystemen gab, wurde dieser Benchmark entwickelt. Vorher nutzte noch jeder Softwarehersteller sein eigenes Messwerkzeug, das auf die Stärken des eigenen Tools ausgerichtet ist. Einziger Vergleichsparameter war somit nur die pro Zeiteinheit verarbeitete Datenmenge, in der Regel Datensätze pro Sekunde. Ziel des Benchmarks soll es sein, die Softwarelösung hinsichtlich Korrektheit, Vollständigkeit und Leistungsfähigkeit zu untersuchen. Wenn alle Anforderungen vom getesteten Integrationswerkzeug erfüllt werden, wird auf Grundlage der Datenmenge, der dafür benötigten Zeit und dem Preis ein Wert definiert, der den Vergleich verschiedener Migrationslösungen ermöglichen soll [PRJC14, TPC14].

7.2 Aufbau

Der Benchmark besteht aus fünf Phasen, die Initialization Phase, die Historical Load Phase, zwei Incremental Update Phasen und zum Schluss die Automated Audit Phase.
In der Initialization Phase soll das „System Under Test“ von alten Test- und Hilfsdaten bereinigt und die (leere) Zieldatenbank erstellt werden.
Während der Historical Load Phase sollen alle historischen Daten aus den Quelldateien in das Zielsystem geschrieben werden.
In den beiden Incremental Update Phasen werden Veränderungen am Datenbestand vorgenommen. In diesem Benchmark sind das nur das Einfügen von neuen Datensätzen und das Verändern von bestehenden Datensätzen; auf Löschoperation wurde verzichtet.

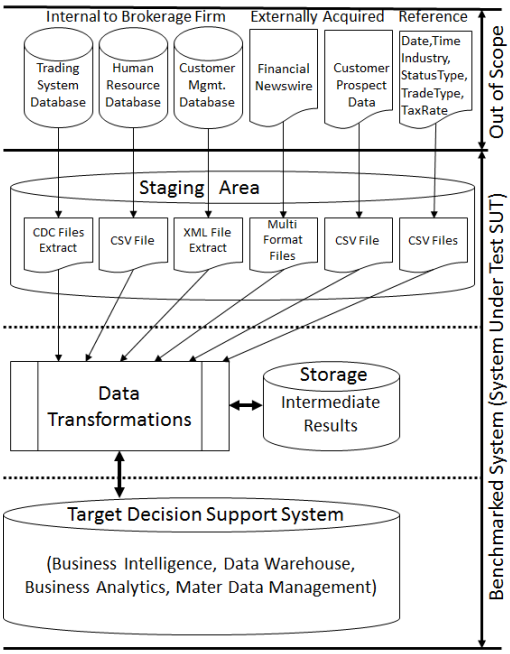

Abbildung 4: Übersicht Migration [PRJC14]

Zum Schluss des Benchmarks soll die Automated Audit Phase durchgeführt werden. Dabei wird im Zielsystem geprüft, ob alle Daten vollständig und korrekt ins Zielsystem migriert wurden.
Die zu migrierenden Daten des fiktionalen Onlinebrokers kommen aus unterschiedlichen Quellen. Alle Daten die mit dem Handel, den Mitarbeitern und den Kunden zu tun haben, kommen aus der eigenen Datenbank. Weitere Kundendaten zur Anreicherung und Aktienkurse von Unternehmen kommen aus externen Quellen. Zusätzlich existieren Referenzdatensätze zu Datum, Zeit, Steuersätzen und anderen Daten, die ausschließlich für die Migration gebraucht werden, aber trotzdem nach der Migration im Zielsystem erhalten bleiben sollen.
Alle Daten stehen in verschiedenen Textdateiformaten (TXT, CSV, CDC, XML) zur Verfügung. Datenbanken werden in diesem Benchmark nicht als Quellsystem verwendet.
Die Quelldateien sollen in der Initialization Phase in die Staging Area geladen und in den darauffolgenden drei Phasen in das Zielsystem migriert

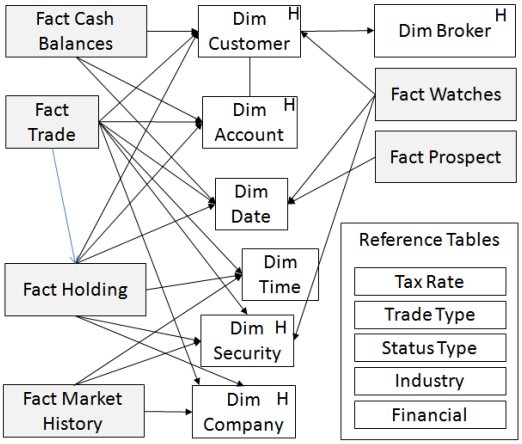

Abbildung 5: Zielsystem [PRJC14]

werden (Abbildung 4). Innerhalb des Zielsystem wird eine Datenbank im Schneesturmschema angelegt (Abbildung 5).
Für jede Migrationssoftware muss der Test zweimal durchlaufen werden, einmal als Performancetest und einmal als Qualifikationstest. Die beiden Testdurchläufe unterscheiden sich dabei nur durch die Menge der zu migrierenden Daten [PRJC14, TPC14].

7.3 Anforderungen

Aus der beschriebenen Migration ergeben sich für die Datenmigrationssoftware einige Anforderungen, die nun genauer betrachtet werden sollen.

7.3.1 Daten aus Quellsystem auslesen

Für alle Migrationslösungen unerlässlich ist das Lesen von Daten aus dem Quellsystem. Dabei ist es unerheblich, ob es sich dabei um Textdateien oder Datenbanken handelt oder die Daten auf dem lokalen System oder im Netzwerk gespeichert sind. Innerhalb einer Migration soll es dabei auch möglich sein, mehrere Quellen auf einmal zu bearbeiten. Die Quellen müssen nicht zwangsweise das gleiche Format haben.
Für den Benchmark sind alle Daten als Textdateien auf dem lokalen System zur Verfügung gestellt [PRJC14, TPC14].

7.3.2 Daten transformieren

Ebenso wichtig ist die Transformation von Daten. Das können einfache Transformationen wie das Zusammenfügen mehrerer Zeichenketten, aber auch komplexe Transformationen, die sich aus mehreren kleinen Transformationen zusammensetzen, sein. Auch hier können verschiedene Datenformate vorliegen. Im Benchmark sind sehr wenige Transformationen gefordert. In der Regel sollen die Daten eins-zu-eins vom Quell- in das Zielsystem übernommen werden. Die einzige komplexere Transformation ist die Bestimmung eines neuen Attributwertes auf Grundlage mehrerer anderer Attributwerte aus dem Quellsystem [PRJC14, TPC14].

7.3.3 Daten in Zielsystem schreiben

Nachdem die Daten eingelesen und verarbeitet wurden, müssen sie ins Zielsystem geschrieben werden. Beim Schreiben können neue Datensätze eingefügt, aktualisiert oder gelöscht werden. Auch hier muss es möglich sein mehrere Systeme oder Dateien als Ziel zu definieren.
Für den Benchmark ist es nur nötig, neue Datensätze einzufügen und veraltete Datensätze zu aktualisieren. Das Löschen alter Daten ist nicht notwendig. Zudem soll bei einem Update nicht der entsprechende Datensatz komplett geändert werden, sondern dieser nur als veraltet gekennzeichnet und der aktualisierte Datensatz wie ein neuer Datensatz eingefügt werden. Das hat zur Folge, dass bei allen Datensätzen, die sich auf den veralteten Datensatz beziehen, das Fremdschlüsselattribut ebenso aktualisiert werden muss (Aktualisierungsweitergabe) [PRJC14, TPC14].

7.3.4 Referentielle Integrität beachten

Um die eben beschriebenen Updates richtig durchführen zu können, muss bei der Migration die referentielle Integrität beachtet werden. Dazu ist es notwendig, dass aus dem Quellsystem bestimmt wird, welche Attribute als Fremdschlüssel verwendet werden und von welchen Tabellen und welchen Attributen sie abhängig sind. Anhand dessen kann dann die Schreibreihenfolge der Tabelle automatisch bestimmt werden. Die Reihenfolge gibt dann an, dass zunächst alle Daten migriert werden müssen, die keine Fremdschlüssel enthalten und erst danach die Datensätze, die auf andere Relationen verweisen, eingefügt werden. Diese Anforderung wird besonders wichtig, wenn auf die Daten im Zielsystem schon während der Migration zugegriffen wird. Wenn beim Zugriff auf eine Faktentabelle die benötigten Dimensionstabellen noch nicht migriert worden sind, dann entstehen im Falle eines natürlichen Verbundes Dangling Tuples und die Daten werden dem Nutzer im schlimmsten Fall nicht angezeigt.
Wie schon in Abschnitt 7.3.3 beschrieben, ist diese Anforderung für den Benchmark essenziell [PRJC14, TPC14].

7.3.5 Joins

Wenn die Daten einer Zieltabelle aus mehreren Quelltabellen (zum Beispiel für Datenanreicherung) kommen, muss die Migrationssoftware auch Joins zwischen Datensätzen ermöglichen. Dafür ist je nach Migration nicht nur der natürliche Verbund erforderlich, sondern auch der linke, rechte oder volle äußere Verbund. [PRJC14, TPC14].

7.3.6 Filter

Eine weitere Anforderung ist die Filterung von Daten. Dadurch ist es möglich, nur eine Teilmenge der Daten zu migrieren, um zum Beispiel irrelevante Daten auszusortieren. Die Filterung kann anhand von einem oder mehreren Attributwerten erfolgen [PRJC14, TPC14].

7.3.7 Aggregation

Ein weiterer Punkt, den eine Datenmigrationssoftware erfüllen soll, ist die Aggregation von Daten. Dadurch sollen mehrere Datensätze zu einem Datensatz zusammengefasst werden. Mögliche Aggregationen sind die Bildung von Summen oder Durchschnitten oder die Bestimmung von Minimal- oder Maximalwerten. Dazu muss mindestens ein Attribut angegeben werden, auf dessen Basis die Daten gruppiert werden. Für die Bestimmung von Durchschnittswerten ist zudem korrektes mathematisches Runden notwendig. Aggregationen auf nicht-numerischen Daten werden vom Benchmark nicht gefordert [PRJC14, TPC14].

7.3.8 IDs vergeben

Wenn der Nutzer es wünscht, dann sollen die Daten im Zielsystem neue, fortlaufende IDs erhalten. Dazu ist es notwendig, bei der Migration zu dokumentieren, welche IDs aus welcher Tabelle wie ersetzt wurden, um die IDs auch in den abhängigen Relationen korrekt zu ergänzen. Dafür ist es zwingend erforderlich, die in Abschnitt 7.3.4 beschriebene Anforderung zu erfüllen [PRJC14, TPC14].

7.3.9 Automatische Typkonvertierung

Um dem Nutzer die Erstellung der Migration zu erleichtern, sollen die Datentypen zwischen Quell- und Zielsystem automatisch konvertiert werden, sofern dies erforderlich und möglich ist. Insbesondere geht es hierbei um die Konvertierung von Strings und Integer, die boolsche Werte repräsentieren, zum „richtigen“ boolschen Datentyp. So soll zum Beispiel der Integerwert *1* zu *true* und *0* zu *false* konvertiert werden [PRJC14, TPC14].

7.3.10 Erkennung und Eliminierung von Duplikaten

Um eine Datenbereinigung zu ermöglichen, sollen Duplikate von der Software erkannt und eliminiert werden. Damit nicht ungewollt falsche Daten gelöscht werden, genügt es auch, vermeintliche Duplikate dem Nutzer zu melden und dem Nutzer die Entscheidung zu überlassen [PRJC14, TPC14].

7.3.11 „Ereignisorientierte Migration“

Weiterhin erwartet der Benchmark, dass die Migrationssoftware auf bestimmte Ereignisse oder Attributwerte reagieren und unterschiedliche Migrationen durchführen kann. So wird in einer Teilmigration erwartet, dass beim erfolgreichen Join der dazugehörige Datensatz aktualisiert wird und beim Entstehen eines Dangling Tuples ein neuer Datensatz eingefügt wird. Ähnliches gilt für die Migration von CDC-Dateien. Diese Dateien speichern alle Veränderungen auf einem Datenbestand. Die Struktur einer CDC-Datei entspricht der einer CSV-Datei. Das erste Attribut des Datensatzes gibt an, ob die nachfolgenden Daten eingefügt („I“), aktualisiert („U“) oder gelöscht („D“) werden sollen [PRJC14, TPC14].

7.3.12 Verhalten bei fehlerhaften Daten

Ein wichtiger Punkt bei der Datenmigration ist der Umgang mit fehlerhaften Daten. Im Quellsystem können das zum Beispiel Datensätze, die nicht der Datenbeschreibung genügen, sein. Auf der Zielseite sind das vor allem null-Werte für Attribute, die nicht null sein dürfen.
Der Benchmark erwartet, dass diese fehlerhaften Daten bei der Migration übersprungen werden und dem Nutzer diese Fehler mitgeteilt werden, so dass dieser die Fehler anschließend manuell beheben kann [PRJC14, TPC14].

7.3.13 Verhalten beim Löschen der Quelldateien

Ein weiterer Fehler, der entstehen kann, ist das Löschen von Quelldateien während der Migration. Die Migration soll dabei nicht abbrechen oder anderweitig beeinflusst werden [PRJC14, TPC14].

7.3.14 Verhalten beim vorzeitigen Beenden der Migration

Als letzter Fehlerfall ist das vorzeitige Beenden der Migration, zum Beispiel durch einen Stromausfall, zu nennen. Der aktuelle Stand der Migration soll dabei gespeichert werden, Nach dem Neustarten der Software soll die Migration an dieser Stelle automatisch fortgesetzt werden [PRJC14, TPC14].

7.4 Kritik

Der Benchmark deckt alle wichtigen Anforderungen, die eine Datenmigrationssoftware erfüllen soll, ab. Jedoch sind einige Anforderungen zum Teil sehr speziell. So werden als Quelle nur Textdateien und keine Datenbanken verwendet. Ebenso müssen alle Dateien lokal auf dem Rechner, auf dem die Migration abläuft, gespeichert sein. Eine Beurteilung, ob die Software auch Dateien auf anderen Rechnern oder Datenbanken im Allgemeinen verwenden kann, ist somit nicht möglich.
Ein weiterer Kritikpunkt ist die Bestimmung des Endergebnisses, bei dem auch der Preis eine Rolle spielt. So können Migrationslösungen, die weniger Funktionen bieten, über einen geringeren Preis ein besseres Ergebnis erzielen, als eine bessere Lösung, die dementsprechend mehr kostet. Aus diesem Grund werden kostenlos verfügbare Datenmigrationstools hier immer am besten abschneiden, unabhängig vom Funktionsumfang. Alternativ könnte ein Zwischenergebnis, das nicht vom Preis abhängig ist, zur Bewertung herangezogen werden. Jedoch wird damit nur das Verhältnis von migrierter Datenmenge zur dafür verbrauchten Zeit wiedergegeben. Damit würde dann auch das Ziel des Benchmarks, sich von einer Datenmenge-pro-Zeit-Bewertung zu verabschieden, verfehlt.
Das Benchmarkergebnis kann somit nur einen ersten Eindruck über die Qualität der Software vermitteln und sollte nicht zu einer abschließenden Beurteilung herangezogen werden. Auf andere wichtige Punkte, die zur Bewertung von Software nötig sind, wird im Benchmark nicht eingegangen.

8 Analyse

Nachdem in den Abschnitten 5 und 7 die Anforderungen an Datenmigrationssoftware definiert worden sind, soll nun analysiert werden, inwiefern Tensei-Data diese erfüllt. Dazu mussten einige Anpassungen vorgenommen werden, die zunächst erläutert werden. Danach werden die Anforderungen der Forschung betrachtet, bevor dann auf die Anforderungen aus dem Benchmark eingegangen wird. Zuletzt wird ein kurzes Zwischenfazit gezogen.

8.1 Anpassungen durch den Autor

Vor der Durchführung des Benchmarks mussten alle Textdateien mit einem Pipe-Symbol („|“) als Trennzeichen geändert werden. Das Symbol wurde durch ein Komma ersetzt. Andernfalls kann Tensei-Data die Datei nicht parsen. Dadurch ist es dann auch möglich, die Datenbeschreibung der Dateien automatisch zu generieren. Zudem wurden alle null-Werte am Ende einer Zeile durch valide Attributwerte ersetzt (betrifft nur einige Datensätze in einer Datei), da Tensei-Data beim Parsen dieser Textzeilen sonst die Migration abgebrochen hätte.
Streng genommen führen diese Anpassungen schon zum Nicht-Bestehen des Benchmarks.

8.2 Anforderungen aus der Forschung

Die Komplementierung ist die Kernforderung an Datenmigrationssoftware und wird somit auch von Tensei-Data weitestgehend unterstützt. Einziger Kritikpunkt an dieser Stelle ist, dass Joins von Textdateien in der verwendeten Version nicht möglich sind und dadurch das Zusammenführen von mehreren Textdateien in eine Tabelle unmöglich ist.
Der nächste Punkt ist die Reduzierung der Anfragekomplexität. Alleine aus der Definition der Datenmigration ergibt sich, dass diese Anforderungen erfüllt werden, da die Anfrage an das System nicht mehr in kleine Unteranfragen für jedes einzelne System beziehungsweise jede einzelne Datei zerteilt werden muss (wie es bei der Datenföderation gemacht wird). Stattdessen

kann nach der Migration die Anfrage direkt an die Daten im Zielsystem gestellt werden. Dadurch ist es für den Anwender auch möglich, komplexere Anfragen zu stellen, die vorher nicht möglich waren.
Ebenso erfüllt Tensei-Data die Anforderung der Flexibilität. Auch wenn nur Textdateien und die bekanntesten Datenbanksysteme als Quell- und Zielsystem genutzt werden können, sollte das ein Großteil der Anwendungsfälle abdecken. Zudem ist Tensei-Data nicht auf bestimmte Anwendungsfälle oder Szenarien fixiert, sondern kann überall verwendet und an die Migrationswünsche angepasst werden.
Auch die Autonomie der Quelldaten wird von Tensei-Data nicht verletzt. Die Daten werden gemäß der Datenbeschreibung eingelesen und dann intern weiterverarbeitet. Die Quelldaten bleiben dabei unverändert, da nur lesend zugegriffen wird.
In Bezug auf die Datenqualität weist Tensei-Data allerdings Schwächen auf. Wie schon oben angesprochen, hat Tensei-Data Probleme beim Join von Textdateien, womit auch die Anreicherung mit externen Daten in diesem Fall schwierig ist. Ein größeres Problem ist jedoch die Datenbereinigung. So ist es nicht möglich, Duplikate in den Daten zu erkennen und diese im Anschluss zu eliminieren. Ebenso können auch fehlende Daten nicht durch Default- oder Durchschnittswerte ersetzt werden.
Ebenso hat Tensei-Data Schwächen bei der Überprüfung der Konsistenz von Daten, da diese nur aus der Quelle gelesen und danach ins Ziel geschrieben werden. Eine Analyse der Daten findet zwischendurch nicht statt. Ebenfalls werden beim Aktualisieren von historischen Daten die alten Datensätze nicht als veraltet gekennzeichnet und der neue Primärschlüsselwert wird auch nicht an die abhängigen Tabellen weitergegeben.
Der Forderung nach einem automatischen Schema Matching kommt Tensei-Data auch nicht nach. Wenn in der Datenbeschreibung der Quellsysteme mehrere Elemente die gleiche Semantik haben, dann kann das Programm keine Korrespondenzen feststellen.
Als letzte Anforderung der Forschung bleibt noch die automatische Mappingerstellung zu betrachten. Tensei-Data bietet die Möglichkeit anhand der Elementnamen und der Semantik das Mapping vorzuschlagen. Dafür müssen

die Bezeichner auf der Quell- und Zielseite exakt gleich sein. Diese Mappingerstellung kann jedoch keine Transformatoren vorschlagen. Zudem ist der Aufwand für den Nutzer durch das Einfügen der Semantik in die DFASDL oder das Anpassen der Bezeichner nicht viel kleiner, als das hier von einer automatischen Mappingerstellung gesprochen werden kann, sie ist aber in Ansätzen vorhanden.
Da sich die Forderungen nach Aktualität (5.2) und möglichst kurzer Antwortzeit (5.4) primär auf die Datenföderation beziehen und für die Datenmigration nicht relevant sind, werden diese hier nicht weiter betrachtet.
Analog bezieht sich die Forderung nach Transparenz (5.8) eher auf die Software, die die Daten nutzen soll, und wird somit auch nicht weiter betrachtet.

8.3 Anforderungen des Benchmarks

Im Folgenden wird Tensei-Data auf die Anforderungen aus dem Benchmark untersucht. Die Reihenfolge der Anforderungen ist dabei dieselbe wie in Abschnitt 7.3. Anforderungen, die ähnliche Ziel verfolgen oder aus ähnlichem Grund nicht erfüllt werden, wurden dabei zusammengefasst.
Die drei Kernanforderungen Daten aus dem Quellsystem zu lesen, diese zu transformieren und dann in das Zielsystem zu schreiben, werden von Tensei-Data ganz gut erfüllt. Einzig das Auslesen von Attributen aus einer XML-Datei ist nicht möglich, wenn gleichzeitig auch der Elementinhalt gelesen werden soll. Es können also nur die Attribute oder der Inhalt gelesen werden, beides ist nicht möglich. Das Lesen von Daten aus Multiformatdateien, wie sie im Benchmark verwendet werden, können theoretisch gelesen werden, bedürfen aber sehr großen manuellen Aufwands, da in der Datenbeschreibung genau angegeben werden muss, wann sich das Format ändert. Zudem können die Attribute bei dieser Art von Dateien nur mit regulären Ausdrücken aus dem Datensatz separiert werden.
Wenn bei der Migration eine Datenbank ein Data Warehouse oder ähnliches verwendet wird, dann ist es auch wichtig, dass die referentielle Integrität zwischen den Tabellen beachtet wird. Dies macht Tensei-Data jedoch nicht. Wenn diese Beziehungen trotzdem beachtet werden sollen, muss der Anwen-

der, der die Migration durchführt, bei der Erstellung der DFASDL auf die Reihenfolge der Tabellen achten. Zunächst müssen die Tabellen beschrieben werden, die von keiner anderen Tabelle abhängig sind und anschließend können sukzessive die abhängigen Tabellen ergänzt werden.
Um nur die benötigten Daten in der gewünschten Granularität zu migrieren, sind auch die Forderung nach Joins, Filter und Aggregation wichtig. Alle drei Anforderungen werden von Tensei-Data jedoch nur im Datenbankbereich erfüllt. Dazu ist es notwendig, in der DFASDL eine generische Tabelle mit Hilfe einer SQL-Anweisung zu erstellen (die Struktur muss trotzdem beschrieben werden). Tensei-Data reicht im Migrationsprozess das SQL-Statement dann an die Datenbank weiter, sodass diese dann für das Filtern, Zusammenführen und Aggregieren der Daten zuständig ist. Im Bereich der Textdateien ist die Anforderung des Joins vorgesehen, jedoch werden die Daten in der getesteten Version nicht korrekt zusammengeführt. Stattdessen werden die Daten in der Reihenfolge wie sie in der Quelle stehen, unabhängig vom Verbundattribut, verknüpft.
Wenn durch die eben genannten Operationen neue Datensätze erzeugt werden, ist es auch sinnvoll diesen Daten neue IDs zuzuweisen. Diese Anforderung wurde in Tensei-Data bedacht, jedoch funktioniert auch hier die Umsetzung nicht korrekt, was dazu führt, dass die Migration abgebrochen wird.
Die automatische Typkonvertierung funktioniert bei Tensei-Data nur sofern die Datentypen ineinander umwandelbar sind. Sollten die Datentypen nicht konvertierbar sein, dann bricht die Migration auch hier ab. Einzig bei der Konvertierung von oder zu Boolean muss mit Hilfe eines Transformators nachgeholfen werden, weil dieser Datentyp nicht unterstützt wird. Da boolsche Daten in der Regel durch Integer (1 oder 0) oder Strings („true“ oder „false“) dargestellt werden, müssen sie auch mit diesem Datentyp in der DFASDL beschrieben werden.
Wie schon im vorherigen Abschnitt beschrieben, kann Tensei-Data keine Duplikate erkennen. Ebenso können auch keine fehlenden Werte durch Durchschnittswerte ersetzt werden. Ein Ersetzung mit Defaultwerten ist aber mit Hilfe eines Transformators möglich. Ursache für die Nicht-Erfüllung der Anforderung ist die fehlende Datenanalyse während der Migration.

Aus dem gleichen Grund kann auch keine „ereignisorientierte Migration“, wie sie in Abschnitt 7.3.11 beschrieben wurde, stattfinden. Daher konnten auch die beiden Incremental Update Phasen nicht durchgeführt werden.
Das Verhalten bei (unerwarteten) Fehler ist bei Tensei-Data auch sehr interessant. Wenn zum Beispiel beim Parsen einer Quelldatei ein Datensatz dabei ist, der nicht der Datenbeschreibung entspricht, dann führt auch das zum Abbruch der Migration. Auch beim Schreiben von Daten in eine Datenbank kann dieser Fehler auftreten. Als Beispiel kann hier das Schreiben eines null-Wertes in eine Spalte, die keine null-Werte zulässt, genannt werden. In diesem Fall wird auf Datenbankebene ein Rollback durchgeführt und alle zuvor migrierten Daten wieder aus dem Zielsystem gelöscht. Problematisch ist das nur beim Schreiben in eine Datei. Da diese keine Rollbackfunktion haben, bleiben die bis zum Abbruch geschriebenen Daten in der Datei und der Nutzer müsste vor dem nächsten Migrationsversuch diese erst wieder löschen. Im Gegensatz dazu ist das Löschen von Quelldateien während einer Migration kein Problem. Beim Parsen der Datei ist das Löschen nicht möglich, da die Software exklusiven Zugriff auf die Datei hat und das Löschen somit vom Betriebssystem verboten wird. Beim Übergang in die Processing-Phase wird der exklusive Zugriff auf die Datei aufgehoben und das Löschen ermöglicht. Da sich zu diesem Zeitpunkt alle Daten bereits im Arbeitsspeicher befinden, ist das Löschen unproblematisch.
Beim unerwarteten vorzeitigen Beenden der Migration erfüllt die Software auch nicht die Benchmarkanforderung. Nach dem Neustart des Systems und der Migrationssoftware muss die Migration manuell neugestartet werden. Die Migration beginnt dabei wieder von vorne. Dabei ist darauf zu achten, dass die bereits in die Datenbank geschriebenen Daten vorher noch manuell aus der Datenbank gelöscht werden müssen, da in der Datenbank kein Rollback ausgelöst wurde. Gleiches gilt auch für das Schreiben in Textdateien. Es entsteht somit immer ein inkonsistenter Zustand.

8.4 Fazit

Nach dem Betrachten der Anforderungen und der Analyse von Tensei-Data ist zu erkennen, dass die Software im Bereich der Datenbanken die grundlegenden und sehr wichtigen Anforderungen erfüllt. Im Bereich der Textdateien sind jedoch einige Anforderungen wie zum Beispiel das Auslesen der Attribute aus XML-Dateien oder Joins, Filter und Aggregation noch nicht erfüllt. Auch beim Umgang mit Fehlern kann noch nachgebessert werden, insbesondere da diese bei Tensei-Data noch recht häufig auftreten und die Fehlermeldungen manchmal wenig aussagekräftig sind.
Zudem ist Tensei-Data bei der Migration aufgrund der virtuellen Maschine mit 40 bis 50 Zeilen pro Sekunde sehr langsam. Zum Vergleich sind bis zu 300 Zeilen pro Sekunde möglich, wenn die Anwendung direkt aus dem Quelltext gestartet wird. Dem „normalen“ Anwender ist dieser in der Regel aber nicht zugänglich.

9 Vergleich mit Talend Open Studio

Um die Bewertung von Tensei-Data besser einordnen zu können, wird die Software nun mit Open Talend Studio verglichen. Die Wahl fiel auf diese Software, da sie eine ähnliche Zielgruppe wie Tensei-Data anspricht.

9.1 Vergleich

Um den Vergleich durchführen zu können, wurden aus dem Benchmark Teilaufgaben ausgewählt, so dass zu allen Anforderungen eine Aussage getroffen werden kann. Der komplette Benchmark wurde somit nicht durchgeführt. Im Gegensatz zu Tensei-Data wurden hier jedoch keine Anpassungen der Quelldaten durchgeführt.
Auch hier wird zunächst ein Blick auf die Anforderungen aus der Forschung geworfen, bevor im Anschluss die Anforderungen aus dem Benchmark betrachtet werden.

9.1.1 Vergleich bezüglich Forschungsanforderungen

Ebenso wie Tensei-Data erfüllt Talend Open Studio die Forderungen nach Komplementierung, Reduzierung der Anfragekomplexität und Autonomie der Datenquellen. Die Anforderung Flexibilität wird ebenfalls von Talend Open Studio erfüllt, dabei ist die Software sogar besser als Tensei-Data, da sie über deutlich mehr Konnektoren verfügt und somit auch in deutlich mehr Anwendungsfällen einsetzbar ist. Grund für diesen großen Unterschied ist die 10 Jahre längere Entwicklungszeit von Talend Open Studio.
Einzig bei den Anforderungen Datenqualität und Konsistenz weist Talend Open Studio Schwächen auf. Zum Punkt Datenqualität ist dabei die fehlende Erkennung und somit auch fehlende Eliminierung von Duplikaten in den Daten zu nennen. Ebenso können fehlende oder falsche Daten nicht automatisch ersetzt werden. Die Anreicherung mit externen Daten funktioniert jedoch problemlos.
Auch die Konsistenz der Daten kann nicht sichergestellt werden, da keine Analyse der Daten stattfindet. Somit können auch keine widersprüchlichen

Daten erkannt werden. Ebenso ist die Aktualisierungsweitergabe beim Verändern von historischen Daten nicht möglich.
Ähnlich zu Tensei-Data kann auch Talend Open Studio das Mapping für die Migration nicht automatisch erstellen. Selbst das Schema Matching kann nicht durchgeführt werden.
Wie bei der Analyse von Tensei-Data werden die Forderungen nach Aktualität, Antwortzeit und Transparenz hier nicht weiter betrachtet.

9.1.2 Vergleich bezüglich Benchmarkanforderungen

Das Auslesen, Transformieren und Schreiben von Daten ist bei Talend Open Studio problemlos möglich. Aufgrund der Vielzahl an Konnektoren können jedoch deutlich mehr Systeme angebunden werden. Die Transformation von Daten ist bei Talend Open Studio aber etwas komplizierter, da es (fast) keine vorgefertigten Transformatoren gibt. Stattdessen müssen diese in Java selbst programmiert werden. Dadurch kann zwar jede gewünschte Transformation entwickelt werden, jedoch ist der Aufwand ungleich größer und das System nur für erfahrene Nutzer verwendbar.
Beim Auslesen der Daten wird von Talend Open Studio nicht auf die Semantik der Attribute geachtet. Somit weiß das Programm auch nicht, welche Attribute Fremdschlüssel sind und kann daher auch nicht die referentielle Integrität beachten. Wenn zumindest die Schreibreihenfolge beachten werden soll, dann muss wie bei Tensei-Data die Reihenfolge der Teilmigration vom Anwender selber beachtet werden.
Die Forderungen nach Joins, Filter und Aggregationen werden sowohl im Datenbankbereich, als auch auf Ebene von Textdateien erfüllt. Bei den Joins sind sowohl der innere als auch der (linke) äußere Verbund möglich, einen vollen äußeren Verbund gibt es jedoch nicht. Sowohl die Filter als auch die Aggregationen erlauben es, die Daten nicht nur anhand eines Attributes zu filtern beziehungsweise zu aggregieren, sondern an beliebig vielen, sodass die Datenmenge und Granularität frei gewählt werden können.
Die Neuvergabe von IDs ist bei Talend Open Studio eigentlich nicht vorgesehen. Da bei der Erstellung eines Migrationsjobs im Hintergrund der da-

zugehörige Java-Code generiert wird, kann dieses Problem aber mit einem Trick gelöst werden. Dazu muss nur im Mapping-Editor der „Attributwert“ *Numeric.sequence(„name“,1,1)* auf das gewählte Attribut gemappt werden. So wird dann im Zielsystem für jeden Datensatz eine neue ID erstellt.
Die automatische Typkonvertierung ist bei Talend Open Studio nur begrenzt möglich. So können alle Datentypen in Strings und die Integerwerte *0* und *1* in den boolschen Datentyp umgewandelt werden. Da das Problem aber relevant ist, gibt es dafür einen Transformator. Bei diesem kann angegeben werden, welches Attribut von Datentyp A nach Datentyp B umgewandelt werden soll. Wie bei Tensei-Data gilt dabei, dass nicht alle Konvertierungen möglich sind und zum vorzeitigen Abbruch der Migration führen können. Das Programm weist jedoch im Vorfeld mit einer Warnung daraufhin, dass die Konvertierung nicht möglich ist.
Wie schon in Abschnitt 9.1.1 geschrieben, findet während der Migration keine Analyse der Daten statt und daher können auch keine Duplikate erkannt oder eliminiert werden.
Bei der „ereignisorientierten Migration“ ist Talend Open Studio jedoch deutlich besser als Tensei-Data. Eine unterschiedliche Bearbeitung der Daten anhand von Attributwerten ist ohne Weiteres möglich. Die Daten können sowohl unterschiedlich geparst werden, als auch an unterschiedliche Orte in das Zielsystem geschrieben werden. Auch die unterschiedliche Bearbeitung auf Grundlage eines gelungenen oder erfolglosen Joins ist möglich.
Bei den Fehlerfällen, die bei einer Migration auftreten können, sind beide Programme hingegen wieder sehr ähnlich. Einziger Unterschied unter diesem Aspekt ist das Verhalten bei fehlerhaften Daten. Während Tensei-Data in solchen Fällen immer die Migration abbricht, kann bei Talend Open Studio gewählt werden, ob es genauso handeln oder den fehlerhaften Datensatz überspringen soll.
Beim Löschen von Quelldaten während einer Migration arbeiten beide Softwarelösungen benchmarkkonform und beim vorzeitigen Beenden der Migration können beide die Anforderung nicht erfüllen.

9.2 Fazit

Die beiden Migrationssysteme Talend Open Studio und Tensei-Data liegen bezüglich der in den Abschnitten 5 und 7.3 genannten Anforderung nicht so weit auseinander.
Mit Blick auf die Anforderungsliste der Forschung sind beide Lösungen in etwa auf Augenhöhe. Talend Open Studio hat hier den Vorteil, dass es deutlich mehr Konnektoren besitzt und somit flexibler einsetzbar ist. Im Gegensatz dazu unterstützt Tensei den Anwender besser bei der Erstellung des Mappings und schlägt ihm auf Grundlage der Datenbeschreibung vor, wie Teile der Daten gemappt werden könnten.
In Hinsicht auf die Anforderungen des Benchmarks sind die Unterschiede jedoch größer.
Hier zeichnet sich Tensei-Data insbesondere durch die Vielzahl an mitgelieferten Transformatoren und der automatischen Typkonvertierung aus. Die Vorteile von Talend Open Studio liegen hier in der größeren Anzahl an Konnektoren und den besseren Möglichkeiten der Datenweiterverarbeitung. Auch bei den möglichen Fehlerfällen, die während der Migration entstehen können, ist Talend Open Studio besser gerüstet. Für die klassischen Anwendungsfälle (Lesen von XML, CSV, DB und Schreiben in DB) sind beide System gut geeignet.
Die Unterschiede in der Leistungsfähigkeit sind jedoch beträchtlich. Während Tensei-Data 40 bis 50 Datensätze pro Sekunde verarbeiten kann, schafft Talend Open Studio durchschnittlich 8000 Datensätze.

10 Erweiterung Tensei-Data

Auf Basis der Benchmarkergebnisse soll nun eine Erweiterung für Tensei-Data entwickelt werden, so dass die Software diese Anforderung erfüllt.

10.1 Auswahl

Aus den nicht erfüllten Anforderungen sind insbesondere die Fremdschlüsselerkennung sowie die Duplikaterkennung als Teilgebiet der Datenqualität interessant. Für beide Themengebiete gibt es bereits Lösungsansätze in der Forschung, die einfach umgesetzt werden können.
In [NH10] werden verschiedene Ansätze zur Erkennung von Duplikaten aus einer oder mehreren Quellen diskutiert. Die Lösungsansätze reichen dabei von der Levenshtein- und Jaro-Winkler-Distanz zur Bestimmung der Ähnlichkeit zweier Zeichenketten bis hin zu Clusteringalgorithmen bei Daten mit komplexen Beziehungen.
Die Erkennung der Fremdschlüssel ist etwas einfacher und kann schemabasiert erfolgen. Es gibt aber auch datenbasierte Ansätze, diese werden in [KPN15] und [PKQN15] vorgestellt. Diese Ansätze basieren auf der Eigenschaft, dass die Werte des abhängigen Attributes immer eine Teilmenge der referenzierten Attributwerte sind.
In Absprache mit der Firma Wegtam wurde vereinbart, die Fremdschlüsselabhängigkeiten schemabasiert aus dem Quellsystem auszulesen. Da die Arbeitsabläufe ähnlich sind, werden zusätzlich die Attribute mit Primärschlüsseleigenschaft und Autoinkrementwerten bestimmt.

10.2 Anforderungen

Damit die Ergebnisse dieser Analyse an den nächsten Teilprozess weitergegeben werden können, muss die Datenbeschreibung DFASDL des Quellsystems dahingehend erweitert werden. Dabei soll dem *seq*-Element das Attribut *db-primary-key* zugewiesen werden. Als Attributwert soll der Spaltenname des Primärschlüsselattributes genutzt werden. Hat eine Tabelle einen zusammengesetzten Primärschlüssel, sollen alle Spaltennamen kommasepa-

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<dfasdl xmlns="http://www.dfasdl.org/DFASDL" semantic="custom">
  <seq id="child">
    <elem id="child_row">
      <num db-column-name="id" id="child_row_id" max-digits="11"/>
      <num db-column-name="parent_id" id="child_row_parent_id" max-digits="11"/>
    </elem>
  </seq>
  <seq id="parent">
    <elem id="parent_row">
      <num db-column-name="id" id="parent_row_id" max-digits="11"/>
    </elem>
  </seq>
</dfasdl>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<dfasdl xmlns="http://www.dfasdl.org/DFASDL" semantic="custom">
  <seq id="child">
    <elem id="child_row">
      <num db-column-name="id" id="child_row_id" max-digits="11"/>
      <num db-column-name="parent_id" db-foreign-key="parent_row_id" id="child_row_parent_id" max-digits="11"/>
    </elem>
  </seq>
  <seq db-primary-key="id" id="parent">
    <elem id="parent_row">
      <num db-auto-inc="true" db-column-name="id" id="parent_row_id" max-digits="11"/>
    </elem>
  </seq>
</dfasdl>
```

Abbildung 6: Vergleich der DFASDL vorher (oben) und nachher (unten)

riert als Wert verwendet werden. Wenn die Tabelle keinen ausgewiesenen Primärschlüssel hat, dann soll das Attribut weggelassen werden.
Tabellenattribute, die Fremdschlüssel sind, sollen in der Datenbeschreibung mit dem Attribut *db-foreign-key* versehen werden. Im Gegensatz zu den Primärschlüsseln soll hier die eindeutige ID des referenzierten Elementes als Wert genutzt werden. Dadurch kann beim Schreibprozess korrekt ermittelt werden, auf welches Attribut in welcher Tabelle der Fremdschlüssel verweist.
Zuletzt sollen Elemente, die eine Autoinkrementeigenschaft haben, gekennzeichnet werden. Dazu soll das Attribut *db-auto-inc* verwendet werden. Bei Elementen, die diese Eigenschaft besitzen, wird der Wert des Attributes auf *„true“* (als String) gesetzt, andernfalls wird das Attribut weggelassen.
Wie sich die DFASDL durch die Implementierung verändert, ist in Abbildung 6 zu sehen.
Diese Erweiterung soll später bei allen unterstützten Datenbanksystemen nutzbar sein. Da das Vorgehen bei allen Systemen aber ähnlich ist und sich nur die SQL-Anweisungen unterscheiden, soll die Erweiterung im Rahmen

dieser Arbeit erst einmal nur für zwei bis vier selbstgewählte Systeme umgesetzt werden. Die Erstellung der DFASDL bei den anderen Systemen darf dabei aber nicht fehlerhaft werden. Konkret wurden die Datenbanksysteme MySQL, PostgreSQL und H2 bearbeitet.
Das Bestimmen der Reihenfolge der Tabellen, das Schreiben der Daten in das Zielsystem und die Erweiterung der DFASDL-API gehören nicht zum Aufgabenumfang und werden von den Wegtam-Mitarbeitern übernommen und parallel zu dieser Erweiterung implementiert.

10.3 Konzept

Alle Informationen, die für die obengenannten Anforderungen gebraucht werden, können aus der *Information_Schema*-Datenbank entnommen werden. Hier speichern alle Datenbanksysteme den Aufbau und die Struktur der Datenbanken. Somit lassen sich auch hier die Informationen zu Primär- und Fremdschlüsseln sowie Autoinkrementen finden. Um den Programmieranteil in Scala möglichst gering zu halten, werden die SQL-Anfragen so gestaltet, dass die notwendigen Informationen direkt aus dem Ergebnis abgelesen werden können. Insbesondere ist darauf zu achten, dass der Primärschlüssel einer Tabelle aus mehreren Attributen zusammengesetzt sein kann. Das heißt, dass in solch einem Fall eine Attributliste im Ergebnis zurückgegeben wird.
Um bereits vorhandene Strukturen zu nutzen, kann der DatabaseSchema-Extractor-Trait (vergleichbar mit einem Interface in Java) für die Erweiterung verwendet werden. Hier werden bereits zwei SQL-Anfragen an die *Information_Schema*-Datenbank gestellt, um die notwendigen Informationen zusammenzusuchen und daraus die DFASDL zu erstellen.
Während es bei der ersten Anfrage darum geht, alle Tabellennamen der Datenbank zu bestimmen, geht es bei der Zweiten darum, alle Attribute einer Relation inklusiver der Metadaten wie den Datentyp oder die maximale Länge der Werte zu ermitteln.

10.4 Umsetzung

Damit das Programm durch weitere Leseprozesse nicht unnötig langsamer gemacht wird, wurden die bestehenden SQL-Anfragen erweitert. Da der Primärschlüssel an das *seq*-Tag gebunden werden soll, wurde die erste SQL-Anfrage so ergänzt, dass sie neben dem Tabellennamen (Attribut *Table_Name*) auch die Liste der dazugehörigen Primärschlüssel (Attribut *Column_Name*) zurückgibt. Dazu war es notwendig, die bereits angefragte *Tables*-Tabelle mittels Verbund mit der entsprechenden Tabelle, die Information über die Primärschlüssel enthält, zu erweitern. Je nach Datenbanksystem können das unterschiedliche Tabellen sein. So wird bei MySQL dafür die *Columns*-Tabelle, bei PostgreSQL die *Table_Constraints*- sowie die *Key_Column_Usage*-Tabelle und bei H2 die *Constraints*-Tabelle genutzt. Besondere Herausforderung dabei war das korrekte Verbinden der Tabellen, da sie keinen ausgewiesenen Primärschlüssel haben. Stattdessen musste der Join über mehrere Attribute, die den Namen der Datenbank und den Namen der Tabelle beinhalten, ausgeführt werden.
Da eine Tabelle keinen, einen oder einen zusammengesetzten Primärschlüssel haben kann, ist es zwingend notwendig, einen Left Join (beziehungsweise Right Join, wenn man die Reihenfolge der Tabellen invertiert) zu verwenden. Um die Spaltennamen bei einem zusammengesetzten Primärschlüssel in einem Datensatz kommasepariert zu gruppieren, wurde die Group By-Klausel auf den Tabellennamen angewendet. Zusätzlich musste in der Select-Klausel die Funktion *Group_Concat()* (bei PostgreSQL *String_Agg()*) auf die Spalte *Column_Name* angewendet werden.
Damit beim Auslesen der Ergebnistabelle keine Exception kommt, muss zusätzlich darauf geachtet werden, dass der Spaltenname beim Ergebnis bei allen Datenbanksystem der Gleiche ist. Das kann mit einer Umbenennung realisiert werden. Damit bei den im Rahmen dieser Arbeit nicht beachteten Datenbanksystemen die Datenstruktur auch weiterhin bestimmt werden kann, war es nötig, bei diesen SQL-Anfragen eine generische Spalte in das Ergebnis einzubauen, die immer den Wert *null* hat (*SELECT table_name, null AS column_name FROM ...*).

Für die Ermittlung der Fremdschlüssel und Autoinkrementspalten konnte dann die zweite SQL-Anweisung, die bisher nur die *Columns*-Tabelle angefragt hat, genutzt werden.
Zur Bestimmung der Autoinkrementattribute wäre keine Anpassung des ursprünglichen SQL-Statements nötig gewesen, da sich diese Information in der *Columns*-Tabelle befindet (Attribut *Column_Default* bei H2 und PostgreSQL, *Extra* bei MySQL). Einzig die unterschiedlichen Attributwerte mussten beachtet werden. Während bei MySQL „auto_increment“ als Attributwert angegeben werden, wird bei den anderen beiden Datenbanksystemen ein Verweis auf den internen Index zurückgeliefert.
Für die Ermittlung der Fremdschlüssel waren jedoch einige Änderungen notwendig. So mussten auch hier wieder mehrere Tabellen verbunden werden. Je nach Datenbanksystem unterschieden sich dabei nicht nur die Namen der Tabellen sondern auch die Anzahl der Tabellen, die benötigt wurden, um an alle Informationen zu kommen. So wurden bei PostgreSQL zusätzlich die Tabellen *Key_Column_Usage*, *Table_Constraints* und *Constraint_Column_Usage* benötigt. Im Vergleich dazu wird bei MySQL die Tabelle *Key_Column_Usage* hinzugenommen. Auch hier musste wieder der Left Join verwendet werden, da ein Attribut entweder Fremdschlüssel ist oder eben nicht. Wenn das Attribut ein Fremdschlüssel ist, dann ist es wichtig zu wissen, auf welches Attribut sich dieses bezieht. Diese Informationen befinden sich ebenfalls in der Ergebnismenge. Bei der Erstellung der Datenbeschreibung ist dann darauf zu achten, dass die ID, die im Attribut *db-foreign-key* angegeben wird, exakt der ID des referenzierten Elementes entspricht, weil sonst beim Erstellen der Datenbank im Zielsystem Fehler auftreten würden.
Da die SQL-Anweisung die Ergebnisse genauso zurückliefert, wie sie im Programmcode gebraucht werden, musste im Code nur eine neue Klasse eingeführt werden. Diese dient dazu, die Daten der ersten SQL-Anfrage zu kapseln. Daher musste bei einigen Funktionen noch der Datentyp des Rückgabewertes beziehungsweise der übergebenen Parameter angepasst werden. Für die Weitergabe der Ergebnisse der zweiten SQL-Anfrage musste eine Funktion um zwei Parameter für Fremdschlüssel und Autoinkremente erweitert werden.

Zusätzlich wird nach der Erstellung der DFASDL noch die Richtigkeit der Daten geprüft. Dazu wird für jeden Wert von *db-foreign-key* geprüft, ob ein anderes Datenelement diesen Wert als ID besitzt. Andernfalls wird dem Nutzer eine Fehlermeldung ausgegeben und die Migration abgebrochen.

10.5 Erweiterungsmöglichkeiten

Die Primärschlüssel, Fremdschlüsselabhängigkeiten und Autoinkrementeigenschaft können auch datenbasiert bestimmt werden. Jedoch verfolgt die Firma Wegtam das Kredo, dass das System „dumm“ sein soll. Das heißt, dass in das Zielsystem nicht mehr Informationen geschrieben werden sollen, als das Quellsystem (explizit) bereitstellt.
Zudem ist der in [KPN15] vorgestellt Algorithmus zur Entdeckung von Fremdschlüsseln nicht geeignet, da er nur unäre Beziehungen findet. Im Gegensatz dazu erfüllt die Weiterentwicklung aus [PKQN15] die Anforderung, auch zusammengesetzte Schlüssel zu erkennen. Allerdings ist hier die exponentielle Komplexität sehr hoch. Eventuell könnte dieser Algorithmus in Zukunft als optionales Feature eingebaut werden.

11 Ausblick

Durch die Benchmarkergebnisse ergeben sich für die Entwickler von Tensei-Data einige Aufgaben. So sollten weitere Konnektoren eingebaut werden, um die Flexibilität weiter zu steigern. Darüber hinaus könnte das automatische Mapping weiter ausgebaut werden, und auch eine datenbasierte Fremdschlüsselerkennung sowie die Duplikaterkennung sollten nicht aus den Augen verloren werden. Zudem ist es wichtig, weiter innovative Ideen zu entwickeln, um sich von Talend und anderen Migrationssystemen abzusetzen.
Zur weiteren Beurteilung von Tensei-Data ist es sinnvoll, das Tool mit einem der Marktführer im Bereich der Datenmigration zu vergleichen. Ideale Kandidaten könnten dabei die Tools von IBM, SAP und Oracle sein.
Ebenso sollte der Benchmark noch einmal wiederholt werden, wenn alle Anforderungen erfüllt werden, um ein abschließendes Ergebnis zu berechnen. Alternativ könnte auch ein anderer Benchmarktest verwendet werden, der vielleicht andere Anforderungen stellt. Vom Umfang könnte der DIPBench dafür gut geeignet sein.
Zudem sollte das Verhalten bei größer werdenen Datenmengen bei Tensei-Data noch einmal genau betrachtet werden. Schon bei geringen Datenmenge von 50.000 Datensätzen wurde deutlich, dass das Verarbeiten der Daten immer langsamer wurde.

12 Fazit

Mit Blick auf die kurze Entwicklungszeit ist Tensei-Data bereits sehr weit fortgeschritten. So können fast alle relevanten Anforderungen der Forschung erfüllt werden. Die Anforderungen des Benchmarks können im Gegensatz dazu nicht vollständig erfüllt werden. So weist die Software bei der Migration von Textdateien noch einige Lücken, wie das Filtern oder Aggregieren von Daten, auf. Im Datenbankbereich können aber auch diese Forderungen erfüllt werden. Einzig bei der Leistungsfähigkeit und der fehlenden Datenanalyse zeigt die Software deutliche Schwächen. Zudem ist das System noch sehr fehleranfällig, was zu häufigen Migrationsabbrüchen führt.
Auch im Vergleich mit Talend Open Studio schlägt sich Tensei-Data sehr gut. Bei vielen Anforderungen sind beide Tools ähnlich gut; in einigen Fällen ist Tensei-Data sogar besser. Jedoch gibt es im Speziellen bei der Migration aus Textdateien noch deutliches Verbesserungspotenzial gegenüber Talend Open Studio. Ebenso weist Talend Open Studio eine bedeutend größere Leistungsfähigkeit und Flexibilität auf.

Literatur

[AGCM15] Patricia Arocena, Boris Glavic, Radu Ciucanu, and Renée J. Miller. The iBench Project. `http://dblab.cs.toronto.edu/project/iBench/`, 2015. Zuletzt besucht: 6. Januar 2016, 10:25.

[BHLW08] Matthias Böhm, Dirk Habich, Wolfgang Lehner, and Uwe Wloka. DIPBench: An independent benchmark for Data-Intensive Integration Processes. In *Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference on*, pages 214–221, April 2008.

[FKPT11] Ronald Fagin, Phokion G. Kolaitis, Lucian Popa, and Wang-Chiew Tan. *Schema Matching and Mapping*, chapter Schema Mapping Evolution Through Composition and Inversion, pages 191–222. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[HRO06] Alon Halevy, Anand Rajaraman, and Joann Ordille. Data Integration: The Teenage Years. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, VLDB '06, pages 9–16. VLDB Endowment, 2006.

[KPN15] Sebastian Kruse, Thorsten Papenbrock, and Felix Naumann. Scaling out the discovery of inclusion dependencies. In Thomas Seidl, Norbert Ritter, Harald Schöning, Kai-Uwe Sattler, Theo Härder, Steffen Friedrich, and Wolfram Wingerath, editors, *Datenbanksysteme für Business, Technologie und Web (BTW), 16. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme"(DBIS), 4.-6.3.2015 in Hamburg, Germany. Proceedings*, volume 241 of *LNI*, pages 445–454. GI, 2015.

[KSS14] Veit Köppen, Gunter Saake, and Kai-Uwe Sattler. *Data Warehouse Technologien, 2. Auflage.* MITP, 2014.

[LN07] Ulf Leser and Felix Naumann. *Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen*. dpunkt-Verlag, 2007.

[MFH+01] Renée J. Miller, Daniel Fisla, Mary Huang, David Kymlicka, Fei Ku, and Vivian Lee. The Amalgam Schema and Data Integration Test Suite. `http://www.cs.toronto.edu/~miller/amalgam`, 2001. Zuletzt besucht: 3. März 2016, 13:30.

[NH10] Felix Naumann and Melanie Herschel. *An Introduction to Duplicate Detection*. Morgan and Claypool Publishers, 2010.

[OV99] M. Tamer Özsu and Patrick Valduriez. *Principles of Distributed Database Systems (2nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1999.

[PKQN15] Thorsten Papenbrock, Sebastian Kruse, Jorge-Arnulfo Quiané-Ruiz, and Felix Naumann. Divide & conquer-based inclusion dependency discovery. *PVLDB*, 8(7):774–785, 2015.

[PRJC14] Meikel Poess, Tilmann Rabl, Hans-Arno Jacobsen, and Brian Caufield. TPC-DI: The First Industry Benchmark for Data Integration. *Proc. VLDB Endow.*, 7(13):1367–1378, August 2014.

[Rev15] Solutions Review. Data Integration and Application Integration Solutions Directory. `http://solutions-review.com/data-integration/data-integration-solutions-directory/`, 2015. Zuletzt besucht: 3. März 2016, 13:30.

[RSB12] Carlos R. Rivero, Andreas Schultz, and Christian Bizer. Linked Open Data Integration Benchmark (LODIB). `http://lodib.wbsg.de/`, 2012. Zuletzt besucht: 3. März 2016, 13:30.

[TPC14] TPC. TPC Benchmark DI. `http://www.tpc.org/TPC_Documents_Current_Versions/pdf/TPCDI-v1.1.0.pdf`, 2014. Zuletzt besucht: 7. Januar 2016, 11:05.

[TR15] Eric Thoo and Lakshmi Randall. Magic Quadrant for Data Integration Tools. http://now.informatica.com/en_data-integration-magic-quadrant_tools_analyst-report_2942.html, 2015. Zuletzt besucht: 3. März 2016, 15:05; Dokument wird nach Registrierung per Mail zugesandt.

Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich diese Bachelorarbeit selbstständig ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe. Alle den benutzten Quellen wörtlich oder sinngemäß entnommenen Stellen sind als solche einzeln kenntlich gemacht.
Diese Arbeit ist bislang keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht worden.
Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Rostock, den 21. März 2016