Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

# Interaktive kontextsensitive Integration und Aufbereitung heterogener Datenquellen unter Verwendung von Data Mashups

Christoph Trybek

| | |
|---|---|
| **Studiengang:** | Informatik |
| **Prüfer/in:** | Prof. Dr.-Ing. habil. Bernhard Mitschang |
| **Betreuer/in:** | Dipl.-Inf. Michael Behringer,<br>Dipl.-Inf. Pascal Hirmer |
| **Beginn am:** | 24. Oktober 2016 |
| **Beendet am:** | 25. April 2017 |
| **CR-Nummer:** | D.1.7, D.2.12, H.1.2, H.5.2 |

# Kurzfassung

Durch eine stetig anwachsende Menge an Daten wird es zunehmend schwieriger, diese aufzubereiten und zu integrieren. Da bisherige Lösungsansätze eine hohe technische Versiertheit erfordern ist es notwendig, die Aufgaben an IT-Experten zu delegieren, die keinerlei domänenspezifische Kenntnisse über die vorliegenden Daten besitzen. Dies kann zu Fehlern oder einem erhöhten Kommunikationsaufwand führen. Aus diesem Grund sollten Domänenexperten, ohne tiefgründiges Programmierverständnis, in der Lage sein, umfangreiche Datensätze selbstständig aufzubereiten und zu integrieren. Data Mashups stellen einen leicht zu bedienenden Ansatz für die Verarbeitung von Daten dar und eignen sich deshalb besonders zur Einbindung von Anwendern ohne technischen Hintergrund.

Im Rahmen der nachfolgenden Arbeit werden zunächst die notwendigen Schritte einer Datenintegration definiert und anschließend ein Konzept entwickelt, das den Anwender dabei unterstützt. Hierbei werden wiederkehrende Aufgaben vom System übernommen, deren Ausführungen gezielt durch den Anwender beeinflusst werden können. Dazu wurde eine Auswahl essentieller Operationen zur Datenaufbereitung definiert, sowie ein Klassifikationsverfahren entwickelt, welches den Anwender dabei unterstützt, eine interaktive Schemaintegration auszuführen. Der Anwender soll die Daten in einem gewohnten Tabellenformat aufbereiten und integrieren können. Um die Funktionalität des Konzepts zu verdeutlichen, wird ein Prototyp entwickelt, der die Grundlage für eine Implementierung in *FlexMash*, ein an der Universität Stuttgart entwickeltes Data Mashup-Werkzeug, bildet.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 1 Einleitung und Motivation

Mit der kontinuierlich steigenden Menge verfügbarer Daten gewinnen Datenverarbeitung und Datenintegration zunehmend an Bedeutung [MCB+11]. Trotz des Fortschritts bei der technischen Verarbeitung und Analyse von Daten, ist es nach wie vor sehr zeitintensiv Daten in ein gleichförmiges Schema für nachfolgende Analysen zu überführen. Bei der Datenintegration besteht eine große Herausforderung darin, dass die meisten Datenquellen unabhängig voneinander sind und dadurch heterogene Datenstrukturen produzieren. Zur Verbesserung der Qualität dieser Daten sind Analysten dazu angehalten, regelmäßig Daten neu zu strukturieren und Fehlerauswertungen vorzunehmen [KPHH11; LWH14]. Dabei sind manuelle Datenoptimierungen sehr zeitaufwendig, fehleranfällig und kostenintensiv. Im Zusammenhang mit immer umfangreicher werdenden Datenmengen ist dieser Ansatz nicht praktikabel. Durch die gesteigerte Komplexität von Datenstrukturen wird die Arbeit von Analysten zusätzlich erschwert. Eine weitere Herausforderung ist eine schnelle Reaktion auf sprunghafte Anforderungen an die Geschäftslogik der Anwender [LWH14; RB01]. Aus diesen Gründen benötigt man ein zeitsparendes, günstiges und interaktives Verfahren zur Datenintegration.

Zur Lösung, der oben genannten Herausforderungen, beschäftigt sich diese Arbeit mit Konzepten zur Datenaufbereitung und Datenintegration im Kontext von Data Mashups. Data Mashups sind ein anwenderzentrierter Ansatz um eine Datenintegration durchzuführen. Hierbei sollen vor allem Anwender mit wenig Programiererfahrung angesprochen werden, wie Analysten und Berater aber auch professionelle Entwickler und Domänenexperten. Sie können durch eine visuelle Programmierumgebung ohne großen Aufwand Daten integrieren und verarbeiten [DM14; LWH14]. Der vorgestellte Lösungsansatz setzt maßgeblich auf eine interaktive Einbindung des Domänenwissens des Anwenders. Zur Darstellung der Daten wird eine intuitive Benutzeroberfläche verwendet. Durch das Bereitstellen definierter Operationen wird der Anwender bei der Aufbereitung und Integration von Daten bei wiederkehrenden manuellen Aufgaben unterstützt.

## 1.1 Motivationsszenario

Im Zuge dieses Abschnitts wird ein Anwendungsfall beschrieben, welcher zur Verdeutlichung der Problemstellung dient, die durch das entwickelte Konzept gelöst werden soll.

Nach der Übernahme einer Firma durch eine Andere ist es wünschenswert, die unterschiedlichen Datenbanken mit Kundeninformationen in ein uniformes Datenhaltungssystem zu integrieren. Hierzu müssen die Datensätze in eine homogene Struktur gebracht, sowie semantische Äquivalenzen ermittelt werden. Im Regelfall würde eine solche Aufgabe an IT-Experten delegiert werden, der keinerlei Kenntnis über die Domäne der Daten besitzt. Dadurch kann es zu Fehlern, Ungenauigkeiten oder einem Mehraufwand, durch mehrfache Revisionen kommen. Damit die mit diesem Aufwand verbundenen Kosten eingespart werden, soll ein Domänenexperte beauftragt werden, welcher große Kenntnisse über die Systeme beider zu vereinenden Daten besitzt, jedoch über wenig Programmiererfahrung verfügt. Zur Bewältigung dieses Auftrags wäre die Existenz eines Werkzeugs hilfreich, welches intuitiv und ohne das Programmierkenntnisse vorausgesetzt werden, die wiederkehrenden Aufgaben einer Datenaufbereitung und Datenintegration ausführt. Hierbei soll der Anwender interaktiv in den Vorgang eingebunden werden, um sein Domänenwissen in den Integrationsprozess einfließen zu lassen. Der Prozess umfasst drei Stufen: Vorverarbeitung, Schemaintegration und Nachbearbeitung. Im ersten Schritt findet eine computergestützte Aufbereitung der Daten statt. In der nachfolgenden Phase wird die Integration der Schemata durchgeführt und im finalen Abschnitt werden die vereinigten Datensätze erneut aufbereitet. Nach Abschluss des Integrationsvorgangs soll ein einheitliches Schema und ein Protokoll der angewandten Operationen zur Homogenisierung und Integration der beiden Schemata vorliegen. Dieses Protokoll kann dazu genutzt werden, die Transformation der Daten nachzuvollziehen oder als Anleitung für eine erneute Ausführung einer Datenintegration mit ähnlichen Datensätzen verwendet werden.

## Gliederung

Die vorliegende Arbeit ist in folgender Weise gegliedert:

**Kapitel 2 – Grundlagen** beschreibt die Grundlagen dieser Arbeit.

**Kapitel 3 – Verwandte Arbeiten** stellt verschiedene, mit der Thematik dieser Arbeit verwandte, Arbeiten vor.

**Kapitel 4 – Konzept zur semiautomatischen Aufbereitung und Integration** schildert das Konzept für eine automatisierte Schemainegration für FlexMash.

**Kapitel 5 – Prototypische Implementierung** beschreibt die Implementierung des im vorherigen Kapitel entwickelten Konzepts.

**Kapitel 6 – Zusammenfassung und Ausblick** fasst die Ergebnisse der Arbeit zusammen und stellt Anknüpfungspunkte vor.

# 2 Grundlagen

In diesem Kapitel werden die, für diese Arbeit benötigten Grundlagenkenntnisse, vermittelt. Dazu gehören Fachbegriffe und Konzepte im Kontext der Schemaintegration, Datenqualität, Datenaufbereitung und Datamashups.

## 2.1 Schemaintegration

Als Schemaintegration bezeichnet man ein Verfahren, welches unterschiedliche Datenschemata vereint und in eine einheitliche, globale Struktur überführt. Sie umfasst im Allgemeinen Methoden zur Verarbeitung von Daten, wie das Einpflegen einer Datenbank in eine andere oder das Darstellen von strukturierten Daten durch eine Webanwendung [BH08]. Ein Großteil der Arbeit einer Datenintegration beansprucht die Reduktion oder Eliminierung von Problemen mit der Qualität der Ausgangsdaten. Durch die zunehmende Bedeutung von Informationen bei der Entscheidungsfindung von Organisationen ist es immer wichtiger, eine hohe Qualität der integrierten Daten zu garantieren. Inkonsistente, fehlerhafte, ungenaue und veraltete Daten sind häufig der Grund für schlechte Entscheidungen, welche die Organisationen häufig teuer zu stehen kommen [CTT09]. Die häufigsten Komplikationen, bei der Vereinigung mehrerer Schemata, entstehen durch strukturelle oder semantische Unterschiede. Dies geschieht beispielsweise wenn die Schemata in unterschiedlichen Domänen entworfen wurden. Häufig entstehen Inkompatibilitäten durch die verschiedenen Ansichten, der am Entwurfsprozess beteiligten Personen. Diese Unterschiede sorgen dafür, dass identische Objekte durch unterschiedliche Datenmodelle repräsentiert werden. Ein weiterer Grund für Konflikte sind inkompatible Entwurfsspezifikationen. Dazu gehört die fehlerhafte Wahl von Namen, Typen, etc., welche zu mangelhaften Ergebnissen bei der Schemaintegration führen. Um die Fehler einer Integration zu minimieren, ist es wichtig, die Menge der gemeinsamen Konzepte aufzuspüren. Des Weiteren müssen auch unterschiedliche Strukturen gefunden werden, die semantisch identisch sind [BH08; RB01].

Man kann eine Schemaintegration als eine Operation ansehen, die zwei Schemata als Eingabe erhält und eine Abbildung zwischen semantisch zusammenhängenden Elementen erzeugt. Eine Abbildung ist definiert, als eine Menge von Elementen aus dem Schema $S_1$, die auf eine Menge von Elementen aus Schema $S_2$, projiziert wird. Jedes dieser Elemente kann eine Regel enthalten, welche die Beziehung zwischen ihnen definiert. Diese Beschreibung kann eine Richtung oder eine Relation zwischen den Elementen enthalten. In Tabelle 2.1 werden zwei

| **Schema** $S_1$ | **Schema** $S_2$ |
|---|---|
| LagerNummer | LagerhallenID |
| Adresse | Straße |
| PLZ | Hausnummer |
| Telefonnummer | Stadt |

**Tabelle 2.1:** Beispiel zweier unterschiedlicher Schemata

unterschiedliche Schemata $S_1$ und $S_2$ aufgelistet, welche die Informationen von Lagerhallen darstellen. Eine Schemaintegration könnte Element *LagerNummer* in $S_1$, auf *LagerhallenID* in $S_2$, mit der Beziehung *LagerNummer* = *LagerhallenID*, abbilden. Außerdem könnte man die drei Elemente *Straße, Hausnummer, Stadt* in $S_2$ durch Verknüpfen auf *Adresse* in $S_1$ abbilden. Dazu benutzt man eine Regel der Form *Verknüpfe(Straße, Hausnummer, Stadt) = Adresse*. Da die Eigenschaften, für eine präzise Abbildung der Elemente, auf Heuristiken aufbauen, die sich nicht durch eine mathematische Form beschreiben lassen, ist man bei einer Automatisierung darauf beschränkt, eine approximierte Abbildung zu finden [RB01].

## 2.2 Datenqualität

Im Bereich der Datenqualität definieren Wang und Strong [WS96] vier grundlegende Dimensionen: die intrinsische, die kontextuelle und die repräsentative Qualität, sowie die Zugänglichkeit einer Datenquelle:

**Intrinsische Datenqualität**
Die intrinsische Datenqualität wird hauptsächlich an der Genauigkeit, Sachlichkeit, Glaubwürdigkeit und dem Ansehen der Quelle gemessen.

**Kontextuelle Datenqualität**
Die ausschlaggebenden Faktoren bei der kontextuellen Qualität sind die Relevanz, Vollständigkeit und Aktualität der Daten. Des Weiteren spielen auch der Gewinn von zusätzlichem informationellem Wert und ein angemessener Umfang von Daten eine wichtige Rolle.

**Repräsentative Datenqualität**
Die repräsentative Qualität von Daten bezieht sich insbesondere auf eine konsistente und kurzgefasste Darstellung und eine leicht verständliche und interpretierbare Bedeutung der Daten.

**Zugänglichkeit einer Datenquelle**
Die Zugänglichkeit von Daten ist ein wichtiger Aspekt bei der Evaluation der Qualität

Im Rahmen dieser Arbeit werden nur Teile der kontextuellen und der repräsentativen Datenqualität beachtet, da sich nur diese im Kontext des genutzten Datenformats anwenden lassen. Der Fokus liegt insbesondere auf der Vollständigkeit und der Konsistenz von Daten:

**Vollständigkeit von Daten**
Pipino, Lee und Wang [PLW02] unterscheiden die Vollständigkeit von Daten zwischen Schema-, Spalten- und Bestandsvollständigkeit. Ein Schema ist vollständig genau dann wenn es alle Entitäten und Attribute enthält. Die Spaltenvollständigkeit wird aus der Anzahl inhaltloser Felder einer Spalte berechnet. Die Bestandsvollständigkeit eines Schemas ist erreicht, sobald es alle möglichen Werte eines Attributs mindestens einmal enthält.

**Konsistenz von Daten**
Batini und Scannapieco [BS16] definieren die Dimension der Konsistenz von Daten als die Erfassung von Verletzungen semantischer Regeln, die für eine Menge von Daten definiert wurden (z. B. Datentyp).

## 2.3 Datenbereinigung

Reale Daten enthalten, insbesondere durch eine umfangreiche Größe und Herkunft aus verschiedenen, heterogenen Quellen, häufig qualitative Mängel, wie Unvollständigkeit, Inkonsistenz und Verunreinigung. Techniken zur Datenbereinigung ermitteln und beseitigen die Unzulänglichkeiten der Daten und tragen zur Verbesserung ihrer Qualität bei [HPK11]. Eine Datenbereinigung kann nicht allein von Computern zuverlässig bewerkstelligt werden, da ihnen dafür essentielle Informationen über die Domäne der Daten fehlen. Automatisierte Verfahren können den Domänenexperten bei der Beseitigung von potentiellen Fehlern unterstützen, jedoch ist letztendlich eine Verifikation und Korrektion durch den Menschen entscheidend [KHP+11]. Die nachfolgenden Verfahren lösen die zuvor beschriebenen Probleme, im Hinblick auf die Qualität der Daten.

### 2.3.1 Standardisierung von Datenwerten

Einige Einträge innerhalb eines Datensatzes können bestimmte Zeichen, Begriffe oder Abkürzungen enthalten, die keinerlei Informationsgehalt besitzen. Da sie keinen Nutzen für eine Schemaintegration oder Duplikaterkennung haben, sollten sie aus den Dateneinträgen entfernt werden [Chr12]. Eine Herausforderung bei der Normierung von Daten sind unterschiedliche Maßeinheiten, wie Meter und Yard oder Kilogramm und Pound [HPK11]. Um ein Matching von Attributen praktikabel zu machen, sollten Zeichen, Token oder Werte in eine standardisierte Form umgewandelt werden [RD00]. Beispielsweise sollten verschiedene Arten von Klammern durch eine spezifische Klammer ersetzt und Umlaute in ausgeschriebener Form dargestellt werden. Um dies zu erreichen, können entweder Umsetzungstabellen oder Transformationsregeln

spezifiziert werden. Der Vorteil von Umsetzungstabellen besteht darin, dass sie im Vergleich zu Umwandlungsregeln rasch an wechselnde Anforderungen angepasst werden können. Sie können jedoch nicht so effizient und schnell angewendet werden wie Transformationsregeln. Zusätzlich sollten alle vorkommenden Zeichen in ein einheitliches Format überführt werden, wie z. B. ASCII oder Unicode [Chr12].

### 2.3.2 Detektion und Bereinigung fehlerhafter Worte

Die Prozedur der Detektion und Bereinigung fehlerhafter Worte teilt sich in drei Komponenten auf: Aufspüren des Fehlers, Generierung möglicher Verbesserungsvorschläge sowie Erzeugung einer Rangliste dieser Vorschläge. Die Non-Word Fehlerdetektion ist ein Verfahren zur maschinellen Detektion fehlerhafter Worte. Bei der nachfolgenden automatisierten Beseitigung von Fehlern unterscheidet man zwischen Isolated-Word- und kontextabhängiger Wortkorrektur [Kuk92] Durch die Eigenschaften des, in dieser Arbeit genutzten Datenformats, ist nur die Isolated-Word-Wortkorrektur von Bedeutung.

#### Non-Word Fehlerdetektion

Zunächst müssen fehlerhafte Worte maschinell identifiziert werden. Hierzu bedient man sich einer der folgenden Techniken:

**Wörterbuch**
Bei dieser Technik überprüft man, mit Hilfe eines vordefinierten Wörterbuchs, ob das spezifische Wort existiert. Um niedrige Laufzeiten zu erzielen, nutzt man häufig eine Hashtabelle. Dazu berechnet man den Hashwert des jeweiligen Worts und vergleicht das Wort an der Hashadresse mit der Eingabe. Falls die beiden Worte unterschiedlich sind, oder kein Wort mit diesem Hashwert existiert, befindet sich ein Fehler in diesem Wort [Knu98].

**N-Gram**
Hierbei werden Teilzeichenfolgen eines Ausdrucks der Länge $n$ erzeugt, wobei der Wert von $n$ typischerweise zwischen $1$ und $3$ liegt. Anschließend werden die, aus der Eingabe gewonnenen Substrings, mit bereits erzeugten N-Grams aus einem Wörterbuch, anhand ihrer Existenz oder Anzahl ihres Auftretens, verglichen. Enthält die Eingabe dabei nicht vorhandene oder besonders seltene Teilworte, kann sie mit hoher Wahrscheinlichkeit als fehlerhafter Wert markiert werden [MC86].

### Isolated-Word Fehlerkorrektur

Ein Fehler dieser Klasse ist ein einzelnes, falsch buchstabiertes oder abgetipptes, Wort, welches als inkorrekte Repräsentation eines korrekten Wortes identifiziert wurde [BR99; Kuk92]. Die hauptsächlich auftretenden Fehler laut Kukich [Kuk92] sind:

**Typographische Fehler**
Sie entstehen meistens wenn ein Buchstabe an Stelle eines anderen eingetippt wird (z. B. *dre* anstatt *der*). Dabei wird davon ausgegangen, dass der Schreiber die richtige Buchstabenfolge kennt, sie jedoch aus Versehen falsch abgetippt hat.

**Kognitive Fehler**
Sie basieren auf der Annahme, dass der Schreiber sich bewusst für eine inkorrekte Schreibweise des Wortes entscheidet, weil er die Korrekte nicht kennt (z. B. *Zuch* anstatt *Zug*).

**Phonetische Fehler**
Sie werden dadurch erzeugt, dass der Schreiber eine phonetisch korrekte Buchstabenfolge, an Stelle einer grammatikalisch Richtigen, schreibt. Diese Fehler werden als eine Unterklasse der kognitiven Fehler angesehen (z. B. *Fotonenstrahl* anstatt *Photonenstrahl*).

Da sich die Fehlerkategorien überschneiden, kann es sich als sehr herausfordernd erweisen, gefundene Fehler eindeutig zu charakterisieren. Die nachfolgenden Techniken, zur Bereinigung von Isolierten-Wort Fehlern, benötigen jedoch keine präzise Klassifikation der Verfehlungen [CTT09]:

**Damerau–Levenshtein-Distanz**
Ist ein Maß, mit dem bestimmt werden kann, ob und in wie weit sich zwei gegebene Zeichenketten gleichen. Sie errechnet sich durch die Anzahl von Substitutionen, Additionen, Subtraktionen und Transpositionierungen von Zeichen, um einen String in den Anderen zu überführen (z. B. beträgt die Damerau–Levenshtein-Distanz von $Gras$ und $Glas$ 1, da der Buchstabe $r$ durch ein $l$ substituiert wird) [Dam64; Lev66].

**Soundex**
Dieses Verfahren wird dazu genutzt, phonetische Fehler zu korrigieren. Dabei wird auf die Eingabe einer Zeichenfolge ein Code produziert, der die phonetischen Eigenschaften des gegebenen Strings repräsentiert. Die Darstellung besteht aus Anfangszeichen des Ausdrucks und einer vierstelligen, numerischen Darstellung der Aussprache der verbleibenden Zeichenfolge [Phi90].

**Regelbasiert**
Hierbei wird das korrekte Wort anhand vordefinierter Heuristiken bestimmt, die mit dem Wissen um prominente Rechtschreibfehler, erzeugt wurden [YF83].

### 2.3.3 Einfügen fehlender Werte

Das Ersetzen fehlender Werte sollte, um die Daten nicht zu verunreinigen oder bestehende Muster zu zerstören, mit größtmöglicher Sorgfalt erfolgen [Pyl99]. Han et. al. [HPK11] und Christen [Chr12] beschreiben folgende Techniken, um fehlende Werte im Kontext einer Datenintegration adäquat zu behandeln:

**Eintrag entfernen**
Wenn fehlende Werte enthalten sind, wird der Eintrag aus dem Datensatz entfernt. Dies kann im Zuge des Matchingvorgangs zu Verlusten von Projektionselementen führen. Diese Methode ist dann praktikabel, wenn mehrere essentielle Attribute innerhalb eines Eintrags fehlen.

**Manuelles Eintragen**
Diese Operation ist nur für kleine Datensätze geeignet. Sie ist sehr zeitaufwendig und erfordert domänenspezifisches Wissen, um einen geeigneten Wert zu ermitteln.

**Ersetzen durch eine globale Konstante**
Hierbei werden fehlende Werte durch eine globale Konstante, wie z. B. $Unbekannt$ oder $-\infty$ ersetzt.

**Berechnung des fehlenden Werts durch Approximationsverfahren**
Diese Technik eignet sich besonders für numerische Werte. Bei symmetrischen Datenverteilungen wird generell der Mittelwert verwendet, andernfalls der Median.

**Einsetzen des wahrscheinlichsten Werts**
Durch das Anwenden von definierten Regeln, Umsetzungstabellen oder Klassifikationsverfahren auf den Datensatz wird der wahrscheinlichste Wert ermittelt und eingesetzt. Dieses Verfahren erweist sich, durch die Zusammenhänge spezifischer Attribute, wie z. B. $Stadt$ und $Postleitzahl$, als besonders präzise.

### 2.3.4 Aufspüren von Ausreißern

Unter Ausreißern versteht man Objekte in Datensätzen, die sich substanziell von den übrigen Datenobjekten differenzieren. Sie können die übrigen Daten stören oder unbrauchbar machen und sollten deshalb bearbeitet oder entfernt werden [Pyl99]. Han et al. [HPK11] klassifizieren Ausreißer in drei grundlegende Kategorien:

**Globale Ausreißer**
Sie sind der einfachste Typ eines Ausreißers und beschreiben einen Datenwert, der sich maßgeblich von den anderen Werten unterscheidet. Damit diese Datenwerte ermittelt werden können ist es entscheidend, eine angemessene Dimension zugelassener Werte zu definieren.

**Kontextuelle Ausreißer**
Sie sind abweichende Datenwerte innerhalb eines spezifizierten Bezugsrahmens. Ob diese Objekte Ausreißer sind, hängt von ihren jeweiligen kontextuellen oder ihren Verhaltensattributen ab. Für eine Temperaturmessung könnten dies z. B. geographische Koordinaten oder das Datum sein.

**Kollektive Ausreißer**
Sie sind gruppierte Objekte, die, einzeln betrachtet, möglicherweise keine Ausreißer sind, jedoch als Einheit von der Norm abweichen. Damit kollektive Ausreißer zuverlässig identifiziert werden können, müssen die Relationen zwischen den einzelnen Datenobjekten betrachtet werden. Diese Beziehungen errechnen sich aus Abstand oder Grad der Ähnlichkeit.

### Verfahren zur Ermittlung von Ausreißern

Nach Han et al. lassen sich die Techniken zur Detektion abnormaler Datenwerte in supervised, semisupervised und unsupervised klassifizieren [HPK11]. Da keine Klassifikation der Daten vorhanden ist, werden im Rahmen dieser Arbeit nur die nachfolgenden unsupervised Verfahren betrachtet:

**Nachbarschaftsbasiert**
Dieses Verfahren nutzt ein Distanzmaß, um die Ähnlichkeit zwischen den Objekten zu bestimmen. Ausreißer werden dadurch bestimmt, dass ihre Distanz zu einem Objekt signifikant größer ist, als die Distanz des Objektes zu einer Vielzahl anderer Objekte. Dabei wird zwischen *distanzbasierten* und *dichtheitsbasierten* Techniken unterschieden. *Distanzbasierte* Verfahren untersuchen Elemente innerhalb des Umkreises eines Objekts mit einem vorgegebenen Radius. Ist dabei ein definierter Schwellenwert von Nachbarn unterschritten, wird das Objekt als Ausreißer markiert. Dieses Verfahren eignet sich, um globale Ausreißer zu finden. *Dichtheitsbasierte* Methoden betrachten die Dichte eines Elements und seiner Nachbarn. Ist die Dichte geringer, im Vergleich zu den Nachbarn, ist das Objekt ein Ausreißer. Dieses Verfahren eignet sich besonders um lokale Ausreißer aufzuspüren.

**Clusterbasiert**
Diese Technik nutzt die Zugehörigkeit zu Clustern, um abnormale Objekte zu bestimmen. Dabei gehören Ausreißer entweder zu besonders kleinen, abgelegenen Clustern oder zu gar keinem.

Die vorhergehenden Methoden basieren auf der Annahme, dass korrekte Objekte häufiger einem bestimmten Muster folgen als Ausreißer. Dabei bilden die Objekte nicht unbedingt eine uniforme Gruppe, sondern ergeben meistens multiple Gruppen mit unterschiedlichen Attributen. Dabei wird ein Objekt, mit großem Abstand zu diesen Fraktionen, als abnormales Element klassifiziert. Es ist jedoch möglich, dass die Objekte in einigen Anwendungen keinem Muster folgen und unterschiedlich verteilt sind. Daraus folgt, dass kollektive Ausreißer

und abnorme Elemente, in unterschiedlich verteilten Objekten, von diesen Techniken nicht zuverlässig erkannt werden können [HPK11].

### 2.3.5 Ermitteln von Duplikaten

Duplikate lassen sich in umfassenden Datenmengen nur unter großem Aufwand ermitteln, während sie gleichzeitig für eine erhebliche Minderung der Datenqualität sorgen. Im Wesentlichen unterscheidet man zwischen zwei Typen von Duplikaten [NH10]:

**Intra-Quellen-Duplikate**
Sie beschreiben mehrere Einträge in einer Datenbank, welche die gleiche Entität repräsentieren. Sie entstehen häufig, wenn Daten einer bestimmten Entität in das System eingegeben werden, ohne vorher zu prüfen, ob diese bereits existieren. Des Weiteren können Duplikate, durch die Verwendung unterschiedlicher Datenformate oder fehlerhafter Eingaben, entstehen.

**Inter-Quellen-Duplikate**
Sie gelten als identische Repräsentation einer Entität in mehreren Datenbanken, die bei einer Fusion zu Duplikaten führen. Unterschiedliche Datenquellen verwenden meist unterschiedliche Schemata zur Repräsentation ihrer Daten. Die Aufnahme von Daten zu unterschiedlichen Zeitpunkten in unterschiedliche Datenbanken kann, durch eine Evolution der Daten, doppelte Einträge erzeugen.

Im Allgemeinen werden Duplikate ermittelt, indem man alle Einträge einer Datenbank miteinander vergleicht und ihre Ähnlichkeit berechnet. Basierend auf diesem Wert kann festgestellt werden, ob es sich mit hoher Wahrscheinlichkeit um ein Duplikat handelt oder ob eine menschliche Beurteilung benötigt wird [NH10].

#### Algorithmen zur Bestimmung von Duplikaten

Naumann und Herschel [NH10] unterscheiden zwischen Algorithmen für das paarweise Vergleichen, Algorithmen für Daten mit komplexen Beziehungen und Clustering als Möglichkeiten zur Bestimmung von Duplikaten innerhalb von Datenstrukturen. Durch die vergleichbar einfache Anwendbarkeit, sind im Rahmen dieser Arbeit, nur Techniken für das paarweise Vergleichen von Elementen relevant:

**Blockbildung**
Diese Methode teilt die Datensätze in Blöcke auf und untersucht die Einträge innerhalb dieser Blöcke auf Duplikate. Durch diese Partitionierung muss nicht jeder Eintrag miteinander verglichen werden, was für eine wesentlich bessere Laufzeit sorgt. Für die Generierung dieser Blöcke ist es essentiell, die Datensätze anhand einer oder mehrerer Variablen zu sortieren. Wählt man beispielsweise als Variable die ersten drei Buchstaben des Vornamens, wären innerhalb eines Blocks nur Einträge mit dieser Zeichenfolge als

Präfix [Jar89]. Dabei ist es entscheidend die Parameter so zu wählen, dass potenzielle Duplikate genau einer Partition zugewiesen werden. Außerdem sollten die Blöcke eine adäquate Größe besitzen. Sind sie zu groß ist dieses Verfahren nicht mehr praktikabel [NH10].

**Geordnete Nachbarschaft**
Diese Technik berechnet zuerst für jeden Datenbankeintrag einen Schlüssel, basierend auf relevanten Feldern oder Teilen davon. Im nächsten Schritt wird der Datensatz anhand des Schlüssels geordnet. Die zur Erzeugung des Schlüssels notwendigen Attribute sollten so gewählt werden, dass potentielle Duplikate nah beieinander liegen. Anschließend werden die Einträge schrittweise, innerhalb eines Fensters der Größe $w$, auf Ähnlichkeit überprüft. Danach wird der Eintrag an der ersten Position aus dem Intervall entfernt und der nächste Eintag hinzugefügt. Die Genauigkeit sowie die Effizienz dieser Methode hängt von der Wahl der Fenstergröße $w$ ab. Dabei ist bei einem einfachen Durchlauf ein relativ großes Fenster genauer als ein kleines, besitzt jedoch eine höhere Laufzeit. In Studien hat sich gezeigt, dass multiples Traversieren des Datensatzes mit einem kleinen Fenster und wechselnden Schlüsselkandidaten eine höhere Laufzeit hat, aber Ergebnisse mit der höchsten Präzision liefert [HS98].

## 2.4 Konzepte zur automatisierten Schemaintegration

Im nachfolgenden Abschnitt werden die grundlegenden Konzepte einer automatisierten Schemaintegration beschrieben

### 2.4.1 Schema-Level-Matching

Schema-Level-Matcher arbeiten nur mit den Informationen die ein Schema bereitstellt. Die Daten der einzelnen Instanzen werden dabei vernachlässigt. Allgemein wird zwischen zwei Ebenen der Granularität unterschieden: der Element- und der Struktur-Ebene [RB01]. Auf der Element-Ebene werden die Projektionen der Schemata durch die Analyse von Labels und Konzepten der einzelnen Elemente berechnet [Shv04]. Diese Methode ist üblicherweise darauf beschränkt $1:1$-, $1:n$- und $n:1$-Kardinalitäten zu ermitteln. Um auch $n:m$-Abbildungen aufspüren zu können, müssen, auf Struktur-Ebene, strukturelle Abhängigkeiten zwischen den Schemaelementen betrachtet werden [RB01].

#### Namensbasiertes Matching

Bei diesem Verfahren werden äquivalente Schemaelemente durch die Analyse ihrer Namen ermittelt. Die Elemente stimmen genau dann überein, wenn ihre Bezeichnungen identisch oder äquivalent sind. Die Ähnlichkeit zweier Namen lässt sich durch die Betrachtung ihrer

Synonyme und Oberbegriffe bestimmen. Dazu verwendet man hauptsächlich Wörterbücher oder Thesauri. Manchmal kann es hilfreich sein, auf mehrsprachige Wörterverzeichnisse zurückzugreifen, um, in die Sprache aufgenommenen Worte einer anderen Sprache, richtig deuten zu können (z. B. $Team \equiv Gruppe$). Ergänzend dazu kann ein, vom Anwender bereitgestelltes Lexikon mit domänenspezifischen Informationen, wie häufig benutzte Abkürzungen oder besondere firmeninterne Bezeichnungen, eingebunden werden [RB01].

#### Beschreibungsbasiertes Matching

Diese Methode betrachtet zusätzliche Informationen die von Schemata bereitgestellt werden können, wie Kommentare oder Beschreibungen. Diese Informationen können nach Schlüsselworten durchsucht werden, um so die Ähnlichkeit zwischen Schemaelementen zu berechnen [RB01].

#### Bedingungsbasiertes Matching

Schemata enthalten häufig Bedingungen, welche die Datentypen, Wertebereiche, Beziehungstypen etc., der enthaltenen Elemente beschreiben. Diese Bedingungen sind sehr nützlich um die Semantiken der Daten feststellen zu können [LNE89]. Dadurch, dass mehrere Elemente mit identischen Konditionen in einem Schema existieren können, führt die alleinige Evaluation der Bedingungen häufig zu ungenauen $n : m$ Abbildungen. Man kann diese Methode jedoch verwenden, um Verfahren wie das namensbasierte Matching zu präzisieren. Hierzu schließt man bespielsweise die Elemente aus, die zwar den gleichen Namen haben, sich jedoch in ihrem Datentyp unterscheiden [RB01].

### 2.4.2 Instance-Level-Matching

Dieses Konzept analysiert die Werte, welche für das jeweilige Attribut in der Datenstruktur verfügbar sind. Das Instance-Level-Matching ist wesentlich langsamer als das Schema-Level-Matching, da die Projektionen von den Datenwerten der Instanzen abhängen. Der Hauptvorteil bei diesem Ansatz liegt darin, dass durch die Evaluation der Daten eine präzisere Klassifikation erreicht werden kann, als durch die Analyse der Schemalabels. Das ist vor allem für das Zusammenführen von semistrukturierten Daten hilfreich, die lückenhafte Schemainformationen enthalten können. Außerdem kann man mit diesem Verfahren das Schema-Level-Matching verbessern, indem man bei nicht eindeutigen Projektionen die Abbildungen bevorzugt, die eine größere Anzahl ähnlicher Instanzen teilen [KZS+10; RB01]. Bei textuellen Elementen bietet sich eine linguistische Analyse an. Dazu kann man Information-Retrieval- oder Text-Mining-Techniken verwenden, indem man besipielsweise Schlüsselbegriffe findet, charakterisiert und ihre relative Häufigkeit evaluiert. Numerische oder alphanumerische Werte kann man, mit Hilfe von Bedingungen wie Wertebereichen oder Zeichenmustern, klassifizieren. So lassen

sich beispielsweise Telefonnummern, Postleitzahlen oder geographische Koordinaten ermitteln [RB01].

### 2.4.3 Kombination unterschiedlicher Verfahren

Die, in den vorhergehenden Kapiteln erläuterten Verfahren, benutzen jeweils unterschiedliche Informationen der Schemata, um den besten Projektionskandidaten zu ermitteln. Jedes Schema bietet dadurch eine andere Anwendbarkeit und Nützlichkeit für eine Integrationsaufgabe. Dadurch ist es sehr unwahrscheinlich, dass die Verwendung eines einzigen Verfahrens zu befriedigenden Ergebnissen führt. Individuelle Ansätze können entweder direkt als Hybrid oder durch die Kombination der Ergebnisse mehrerer Verfahren miteinander verbunden werden [RB01; Shv04].

#### Hybrid Matching

Dieses Verfahren besteht typischerweise aus einer fest vorgegebenen Zusammensetzung individueller Matching-Methoden, die entweder zeitgleich oder in einer definierten Reihenfolge ausgeführt werden. Dabei erziehlt der hybride Ansatz eine höhere Effizienz, da er nicht so häufig über die Schemata traversieren muss wie eine Kombination einzelner Verfahren [RB01].

#### Composite Matching

Bei dieser Methode werden die Ergebnisse individuell ausgeführter Verfahren, sowie hybrider Techniken miteinander verknüpft. Die Ausführung der Methoden kann entweder als zeitgleich oder aufeinanderfolgend definiert werden. Bei einer sequenziellen Ausführung wird durch das Anwenden eines weiteren Matchers auf die Ergebnismenge des vorhergehenden Matchers kontinuierlich verbessert. Composite Matcher erlauben es dem Anwender, je nach Applikation, aus einer Sammlung individueller Matcher einen geeigneten Ablauf zu modellieren. Dadurch ist dieses Verfahren weitaus flexibler als das Hybrid Matching [RB01].

## 2.5 Data Mashup

Mashups werden als zusammengesetzte Anwendungen definiert, welche aus wiederverwendbaren Daten, einer Anwendungslogik und eventuell aus einer graphischen Benutzeroberfläche bestehen. Dabei unterscheiden sie sich von herkömmlichen Webanwendungen dahingehend, dass sie in der Regel dynamisch entwickelt werden und nur einem sehr bestimmten und kurzlebigen Zweck dienen. Man bezeichnet dabei alle verwendeten Daten, Anwendungslogiken und Benutzeroberflächen als Mashup-Komponenten. Die benutzten Mashup-Komponenten sind entweder lokal oder remote verfügbar und stammen meistens von Drittanbietern in Form von

Web-Feeds oder APIs [APTD11; DM14]. Die Mashup-Logik definiert die Zusammensetzung der einzelnen Komponenten, den Kontrollfluss, den Datenfluss, die Datentransformationen und die externen Schnittstellen des Mashups. Das bedeutet, dass Mashups eigenständige Anwendungen bilden, die aus verschiedenen Anwendungen bestehen und von Anwendern ohne tiefgründiges Programmierwissen entwickelt werden können [DM14].

Data Mashups sind eine Unterkategorie von Mashups, die zur Verarbeitung unterschiedlichster Datenquellen entwickelt worden sind. Ein Data Mashup ermöglicht die Komposition von heterogenen Daten aus Data-Services oder Web-Ressourcen zu einer neuen, potentiell wertvolleren Informationsquelle. Dabei können die Ausgangsquellen strukturierte, semistrukturierte oder unstrukturierte Daten bereitstellen. Um eine uniforme Ergebnismenge zu produzieren, sind Operationen zur Datenmediation, wie Neuformatierung, Bereinigung, Teilung und Vereinigung der Daten nötig. Die Ausgabe wird, in der Regel als im Web zugängliche Ressource, veröffentlicht [APTD11; DM14].

# 3 Verwandte Arbeiten

Dieses Kapitel beschreibt die, für diese Bachelorarbeit relevanten Arbeiten. Dazu gehören FlexMash, Wrangler und Mashroom+. FlexMash ist dahingehend bedeutend, da es die Grundlage für das, in dieser Arbeit entwickelte, Konzept bildet. Die weiteren Anwendungen sind für diese Arbeit von Bedeutung, da sie einen Domänenexperten interaktiv in den Prozess der Datenaufbereitung, durch *Wrangler*, oder Datenintegration, durch *Mashroom+*, einbinden.

## 3.1 FlexMash

Mit dem Ziel die Verarbeitung und Integration von zunehmend komplexer werdenden Datenstrukturen auch von Personen ohne Programmiererfahrung zu ermöglichen, wird an der Universität Stuttgart von Hirmer et al. [HM16; HRWM15] das Tool FlexMash entwickelt. Die Anwendung besteht dabei aus zwei Stufen. Zunächst modelliert der Anwender einen Ablaufplan, den sogenannten *MashupPlan*, unter Verwendung einer graphischen Benutzeroberfläche. Dabei ist es für dessen Anfertigung nicht erforderlich, Kenntnisse über die genaue technische Umsetzung zu besitzen. Der definierte *Mashup Plan* wird in der nachfolgenden Stufe in ein ausführbares Modell transformiert und gestartet.

Ein *Mashup Plan* besteht aus zwei Arten von Nodes: die *Data Source Descriptions (DSD)* und *Data Processing Descriptions (DPD)* sowie einer gemeinsamen Kante die den Kontroll- und Datenfluss beschreibt. Die Anwendung verwendet zur Kommunikation zwischen den Komponenten des *Mashup Plans* JSON-Objekte[1]. Die *DSD* stellen dabei eine abstrakte Beschreibung einer Datenquelle zur Verfügung und die *DPD* definieren die Art und Weise wie diese Datenquellen verarbeitet werden. Diese Komponenten werden von Anwendern mit technischem Hintergrundwissen erstellt und in einem Repository abgespeichert. Anschließend können sie von Anwendern mit Domänenwissen, durch beliebiges Kombinieren zur Modellierung von Ablaufplänen verwendet werden.

[1] http://json.org

## 3.2 Wrangler

Kandel et al. haben mit Wrangler [KPHH11] ein Werkzeug zur interaktiven Datentransformation entwickelt. Besonderes Augenmerk liegt dabei auf der Unterstützung von Analysten, um die, bei der Bearbeitung von umfangreichen Datensätzen wiederkehrenden manuellen Aufgaben, weitgehend zu minimieren. Dabei benutzt dieses Werkzeug *Programming-by-Demonstration* um den Domänenexperten die Konstruktion komplexer Transformationsregeln zu erleichtern. Im Laufe des Transformationsprozesses erzeugt Wrangler ein Protokoll, das die vorgenommenen Schritte enthält. Dies ermöglicht eine Wiederverwendung bei ähnlichen Datentransformationen oder eine Rückverfolgung der durchgeführten Transformationen. Zusätzlich können die einzelnen Transformationsschritte nachträglich verändert oder entfernt werden. Wrangler berechnet anschließend, auf Grundlage der veränderten Transformationshistorie, das Ergebnis der Datentransformation erneut.

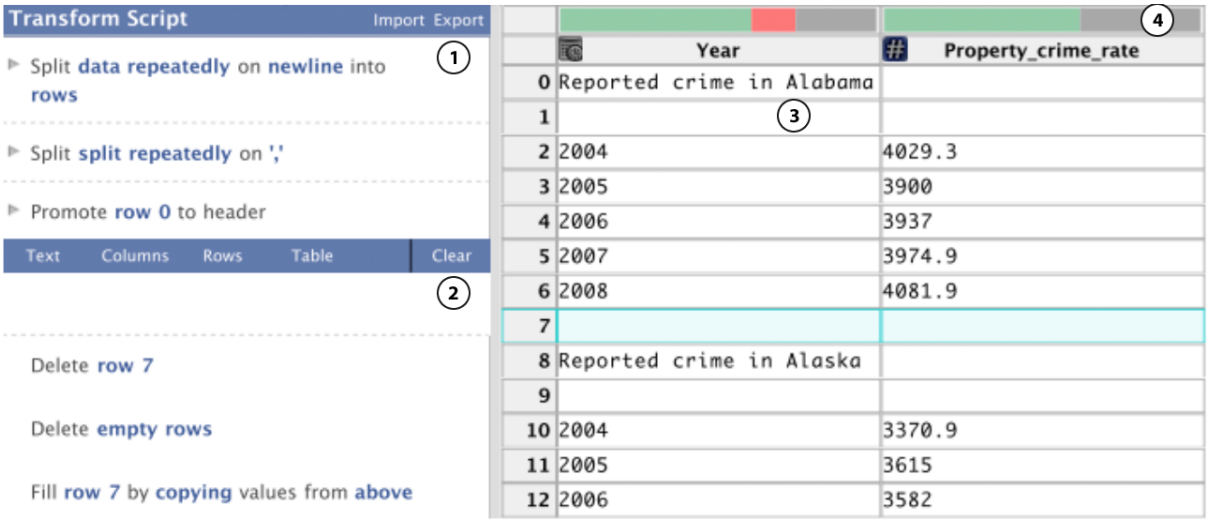



**Abbildung 3.1:** Benutzeroberfläche von Wrangler [KPHH11]

Die Benutzeroberfläche von Wrangler ist in Abbildung 3.1 abgebildet und teilt sich in zwei funktionale Bereiche: eine Liste bereits ausgeführter *(1)* sowie vorgeschlagener *(2)* Umformungsregeln und die zu bearbeitenden Daten in tabellarischer Darstellung *(3)*. Des Weiteren befindet sich über jeder Spalte des Datensatzes eine Anzeige, welche die aktuelle Qualität der Daten visualisiert *(4)*. Diese wird anhand der Vollständigkeit, der Konsistenz des Datentyps und der Validität der Einträge in der jeweiligen Spalte bestimmt. Die Oberfläche bietet, neben einer direkten Manipulation, auch automatisch generierte, sowie aus einem Menü auswählbare Transformationsoperationen, um die vorliegenden Daten in die gewünschte Form zu bringen. Außerdem ist es dem Anwender möglich die Parameter einer genutzten Operation eigenständig an seine Bedürfnisse anzupassen. Um dem Anwender eine schnelle und intuitive Spezifikation zu ermöglichen, werden die Prozeduren in natürlicher Sprache dargestellt. Wrangler bietet dem Analysten eine direkte Vorschau auf die Datenstruktur nach der Ausführung einer spezifischen

Transformationsregel. Dies ermöglicht, die Auswirkung einer Operation schnell zu evaluieren und gegebenenfalls anzupassen oder zu verwerfen.

## 3.3 Mashroom+

Ein Ansatz zur interaktiven Datenintegration aus unterschiedlichen Quellen stammt von Liu, Wang und Han [LWH14]. Mashroom+ zielt darauf ab, mögliche Ungewissheiten, die im Integrationsprozess anfallen, durch Unterstützung eines Domänenexperten, zu behandeln. Dabei werden als Eingabe relationale Datenstrukturen, sowie die Formate HTML, XML und JSON unterstützt und ein eingebettetes relationales Schema als Ausgabe produziert. Dazu kombiniert die Anwendung namensbasierte und strukturbasierte Matchingalgorithmen, die bei der Unterschreitung einer definierten confidence bound ein Eingreifen des Anwenders erfordern. Dieser Schwellenwert kann vom Anwender jederzeit verändert werden und bestimmt den Grad der Interaktion während einer Matchingprozedur. Anhand der entdeckten Zusammenhänge empfiehlt die Anwendung Operationen zur Vereinigung. Der Anwender kann die vorgeschlagenen Operationen anwenden oder selbstständig andere Prozeduren auswählen. Mashroom+ protokolliert die Interaktionen und analysiert sie zur Verbesserung subsequenter Aggregationsprozesse.

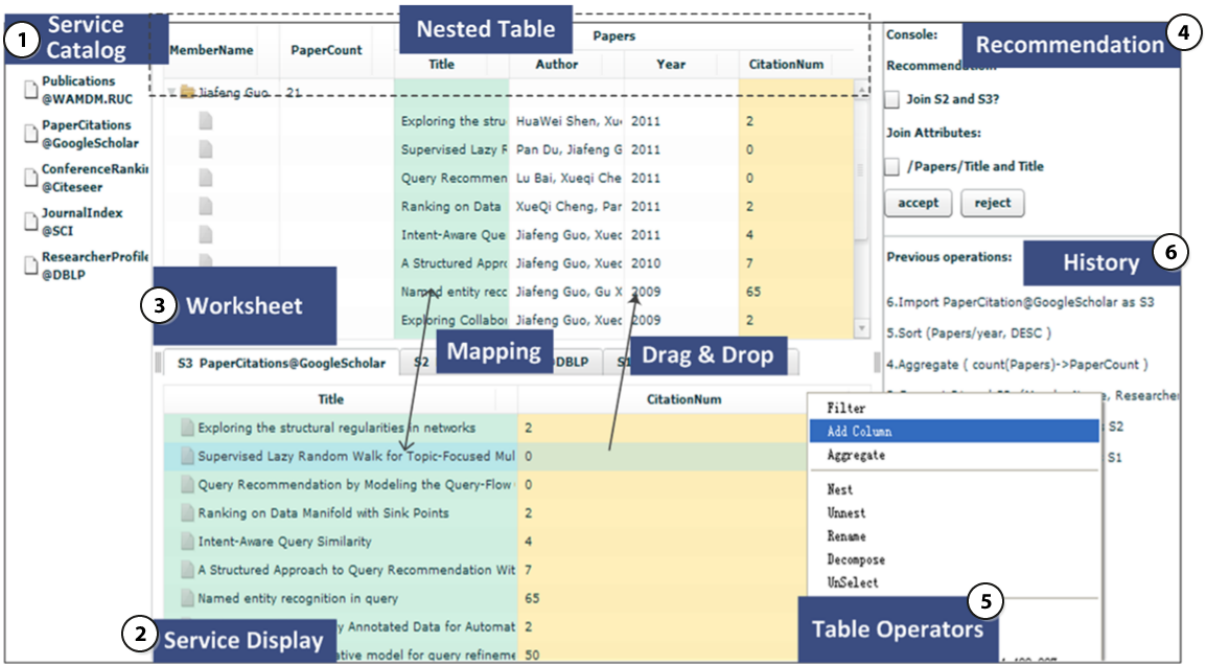



**Abbildung 3.2:** Benutzeroberfläche von Mashroom+ [LWH14]

Die Benutzeroberfläche von Mashroom+ wird in Abbildung 3.2 dargestellt und unterteilt sich in: Service Katalog *(1)*, Service Display *(2)*, Worksheet *(3)*, Matching Empfehlungen *(4)*, Tabellenoperationen *(5)* und Historie *(6)*. Wählt man einen Datenservice aus dem Service Katalog,

wird die Datenstruktur als Nested-Table innerhalb des Service Displays dargestellt. Durch Drag & Drop Operationen kann der Anwender Datensätze aus dem Service Display innerhalb des Worksheets zusammenfügen. Anschließend werden im Bereich der Matching-Empfehlungen gefundene Ungewissheiten, mit vorgeschlagener Integrationsanweisung angezeigt. Zur Modifikation eines zu integrierenden Datensatzes ist es dem Anwender möglich, adäquate Maßnahmen aus den angezeigten Tabellenoperationen auszuwählen. Innerhalb der Historie werden alle, vom Anwender ausgeführten Operationen, aufgelistet, die zur Dokumentation des Integrationsprozesses genutzt werden können.

# 4 Konzept zur semiautomatischen Aufbereitung und Integration

Dieses Kapitel stellt das, im Zuge dieser Arbeit entwickelte Konzept, dar. Es beschreibt die Anforderungen sowie die Methode. Die Spezifikation schildert detailliert, welche Anforderungen an die Applikation gestellt werden und inwiefern sie sich von den bereits existierenden Anwendungen aus Kapitel 3 unterscheidet. Der Methodenablauf(vgl. Abschnitt 4.2) schildert die einzelnen Operationen die vom System in Symbiose mit dem Anwender ausgeführt werden können, um ein für den Anwendungskontext adäquates Integrationsergebnis zu erzielen.

## 4.1 Anforderungen

Beim Umgang mit heterogenen Datenquellen kann es für Personen, die zwar ein umfangreiches Domänenwissen besitzen, jedoch nicht über die erforderlichen Programmierfähigkeiten verfügen, sehr herausfordernd sein, Datensätze aufzubereiten und zu integrieren. Um das vorhandene Domänenwissen des Anwenders bestmöglich zu nutzen, soll das Konzept eine Interaktion über Tabellen erlauben.

Damit eine hohe Datenqualität für das Ergebnis gewährleistet werden kann, soll es dem Anwender ermöglicht werden, die Datensätze vor der Vereinigung computergestützt aufzubereiten. Dazu soll eine Vielzahl von Operationen bereitgestellt werden, die der Domänenexperte, je nach Anwendungskontext, auf Teile oder den gesamten Datensatz anwenden kann. Anschließend soll die Applikation, basierend auf den Schemainformationen und Inhalten der beiden Datenquellen, automatisiert semantische Äquivalenzen ermitteln. Die Klassifikation gefundener Kongruenzen beruht auf einer definierten Korrektheits-Metrik (vgl. Abschnitt 4.2.2).

Die unterschiedlichen Klassen der gefundenen Übereinstimmungen sollen durch die Anwendung visuell oder symbolisch hervorgehoben werden. Damit soll es dem Anwender möglich sein, durch sein Domänenwissen automatisch gefundene semantische Äquivalenzen zu bestätigen oder bei Bedarf zu korrigieren. Anschließend soll die Möglichkeit gegeben werden, die integrierten Datenquellen nachzubearbeiten, um Unzulänglichkeiten der Datenqualität zu beheben.

Zu Dokumentationszwecken und einer leichteren Nachvollziehbarkeit soll die Applikation alle vom Anwender vorgenommenen Transformationsschritte in einer Transformationshistorie abspeichern, die für zukünftige Verwendungen exportiert werden kann.

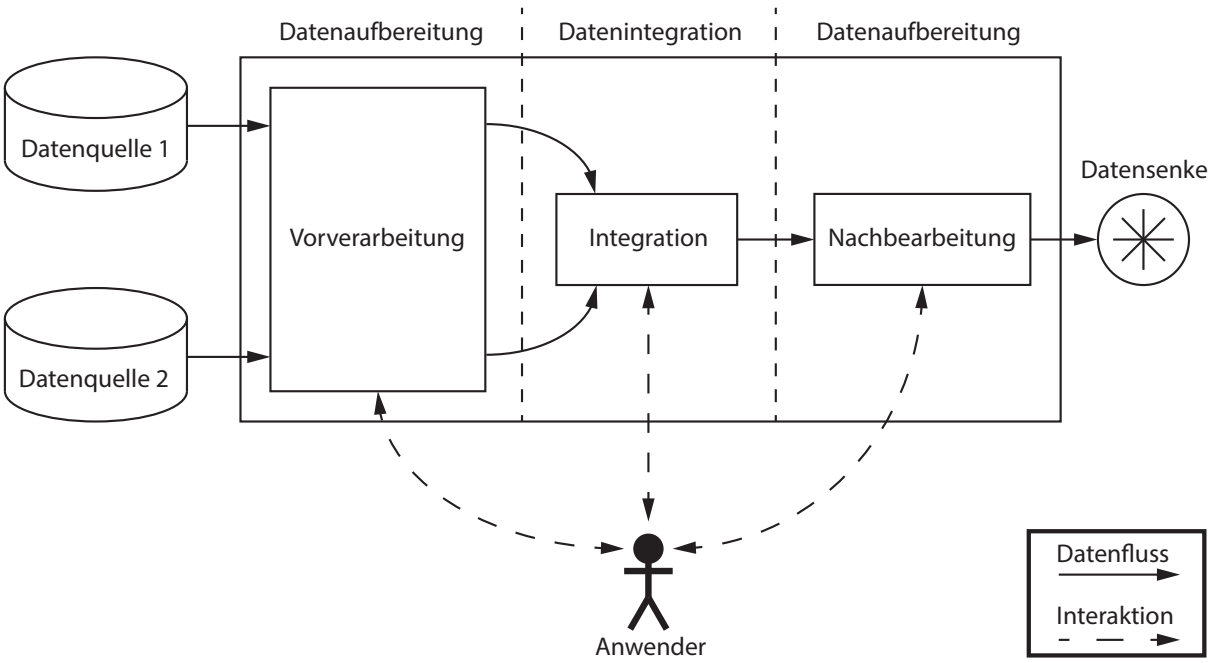



**Abbildung 4.1:** Methode zur semiautomatischen Datenaufbereitung und Schemaintegration

Das in dieser Arbeit entwickelte Konzept unterscheidet sich von den im vorhergehenden Kapitel evaluierten Applikationen dahingehend, dass sie sowohl Methoden zur Datenaufbereitung, wie sie bei *Wrangler* (vgl. Abschnitt 3.2) genutzt werden, als auch Techniken zur computergestützten Schemaintegration, ähnlich wie *Mashroom+* (vgl. Abschnitt 3.3), in einer einzigen Anwendung vereint.

## 4.2 Methode zur semiautomatischen Datenaufbereitung und Schemaintegration

Der Prozessablauf des entwickelten Systems wird in Abbildung 4.1 dargestellt und besteht aus zwei funktionalen Komponenten, welche in drei Verarbeitungsstufen unterteilt werden. Die Datenaufbereitung umfasst die Stufen Vorverarbeitung und Nachbereitung der Datensätze während die Schemaintegration automatisiert die gegebenen Schemata vereinigt. In der Vorverarbeitung wird der Anwender dabei unterstützt, die einzelnen Datensätze für die automatisierte Integration vorzubereiten. Die nächste Stufe dient dazu, semantische Zusammenhänge zwischen den Datensätzen automatisiert zu finden und sie auf dieser Grundlage für den Anwender zusammenzufassen. Dabei kann der Anwender den Selektionsprozess aktiv beeinflussen. Im Zuge der letzten Verarbeitungsstufe werden die letzten qualitativen Mängel des integrierten Resultats durch den Anwender bereinigt. Alle vorgenommenen Operationen werden in einer globalen Historie festgehalten. Diese kann seperat exportiert werden.

### 4.2.1 Datenaufbereitung

Um die möglichen Komplikationen im Zuge einer Datenintegration zu minimieren ist es entscheidend, die Datensätze vorzubereiten. Weiterhin ist es für nachfolgende Operationen wichtig, ein möglichst einheitliches Integrationsergebnis zu erzeugen. Nachfolgend werden Konzepte zur Bereinigung unterschiedlicher Mängel der Datensätze beschrieben.

#### Normierung der Datensätze

Damit semantische Zusammenhänge leichter automatisiert ermittelt werden können, sollten die Datenwerte in einer konsistenten Form vorliegen (vgl. Abschnitt 2.3.1). Die nachfolgenden Operationen können je nach Bedarf des Anwenders auf die gesamte Datenstruktur oder auf ausgewählte Teile angewandt werden.

**Definition unerlaubter Zeichenfolgen**
Dem Anwender muss es ermöglicht werden, eine Definition nicht zugelassener Zeichen oder zusammenhängender Zeichenketten, vorzunehmen. Durch das Löschen kann die syntaktische Ähnlichkeit der Datenstrukturen an die Anforderungen des Domänenexperten angepasst werden.

**Restrukturierung durch reguläre Ausdrücke**
Bei der Bearbeitung umfangreicher Datensätze kann es für den Anwender mühsam sein, unerlaubte Zeichenfolgen schnell und zuverlässig zu erkennen. Daher muss es dem Anwender ermöglicht werden, zulässige Zeichen, in Form regulärer Ausdrücke, zu definieren.

**Neuberechnung numerischer Werte**
Damit numerische Werte in einer identischen semantischen Form vorliegen, müssen sie möglicherweise durch arithmetische Operationen umgeformt werden. Der Anwender kann dabei eigene Transformationsregeln definieren oder aus einer Liste prominenter Umrechnungen, wie z.B. Meilen in Kilometer, Grad Fahrenheit in Kelvin, auswählen. Um Werte nach einer Umwandlung in eine verwertbare Form zu bringen, kann die Anzahl der Nachkommastellen vom Anwender bei der Auswahl der Transformationsvorschrift festgelegt werden.

#### Bereinigung von Duplikaten

Doppelte Einträge innerhalb einer Datenquelle sollten entfernt werden, da sie sich sonst negativ auf nachfolgende Analysen auswirken können (vgl. Abschnitt 2.3.5). Damit alle Dubletten zuverlässig erkannt werden, müssen im Zuge der Vorverarbeitung die beiden zu integrierenden Datenquellen auf Intra-Quellen-Duplikate untersucht werden. Außerdem muss nach einer abgeschlossenen Schemaintegration das Resultat nach Inter-Quellen-Duplikaten durchsucht werden. Gefundene Dubletten sollten für den Anwender gut erkennbar dargestellt werden,

damit dieser sie, durch Auswahl von Operationen, bearbeiten kann. Dazu sollten sie gruppiert werden und sich sichtbar von den übrigen Daten abheben. Dabei können sie entweder durch eine farbliche oder symbolische Markierung abgesetzt werden. Außerdem sollte der Anwender bestimmen können, welche Schemaelemente zur Duplikaterkennung betrachtet werden sollten. Dadurch können Elemente, die einige syntaktisch äquivalente Attribute enthalten, aber unterschiedliche Entitäten beschreiben, aus der Menge gefundener Duplikate ausgeschlossen werden. Zur Bereinigung der gefundenen Dubletten kann der Anwender aus den nachfolgenden Operationen auswählen:

**Entfernen ermittelter Dubletten**
Es ist möglich, gefundene Duplikate vollständig aus dem Datensatz zu entfernen oder eines von ihnen, welches weiterhin im Schema verbleiben sollte, zu spezifizieren.

**Vereinigung gefundener Duplikate**
Sollten lückenhafte Einträge mit unterschiedlichen Informationen als Dubletten erkannt worden sein, können sie miteinander zu einem einzigen Element zusammengefasst werden. Dabei können die, für die Vereinigung relevanten Attribute, manuell festgelegt werden.

### Einfügen fehlender Werte

Das Aufspüren und Einsetzen fehlender Werte (vgl. Abschnitt 2.3.2) kann sich bei umfangreichen Datensätzen als kompliziert und zeitaufwendig erweisen. Diese Methode muss mit der größtmöglichen Sorgfalt erfolgen, um die gegebenen Daten nicht zu verfälschen und bestehende Muster nicht zu zerstören. Unbesetzte Attribute sollten vom System deutlich hervorgehoben werden, damit sie der Anwender manuell oder durch computergestützte Techniken bearbeiten kann.

**Numerische Werte**
Können entweder durch den globalen Median, das globale arithmetische Mittel, einen lokalen Median oder durch eine anwenderspezifische Konstante ersetzt werden. Die globalen Werte werden jeweils aus allen Einträgen der jeweiligen Spalte berechnet. Die lokalen Werte errechnen sich aus den $n$ oberen sowie unteren Nachbarn der leeren Zelle. Die Größe von $n$ kann durch den Anwender frei gewählt werden. Dabei wird die gesamte Liste der Einträge wie ein Ring-Buffer betrachtet: hat eine Zelle weniger als $n$ obere bzw. untere Nachbarn, werden die fehlenden Nachbarn jeweils vom unteren bzw. oberen Ende der Liste betrachtet.

**Korrelierende Attribute**
Innerhalb einer Datenstruktur können unterschiedliche Attribute miteinander korrelieren. Dieser Umstand kann ausgenutzt werden, um fehlende Werte innerhalb dieser

Schemaelemente automatisiert durch korrekte Einträge zu ersetzen. Diese Einsetzungsstrategie erlaubt es, bei nicht korrelierenden Attributen Umsetzungstabellen zu spezifizieren. Nach vollzogener automatisierter Einsetzung, kann die gefundene Auswahl bestätigt oder in Teilen verändert werden.

**Ersetzen durch globale Konstante**
Werte die nicht mit anderen Attributen korrelieren und sich nicht approximieren lassen, sollten durch eine, vom Anwender definierte, Konstante ausgetauscht werden. Dazu können beispielsweise alphanumerische Werte durch *undefined* und numerische Werte durch $-\infty$ ersetzt werden.

### Entfernung orthographischer Fehler

Ein automatisiertes Integrationsverfahren kann beeinträchtigt werden, indem eigentlich semantisch äquivalente Objekte durch einen orthographischen Fehler (vgl. Abschnitt 2.3.2) nicht als solche erkannt werden. Dies kann eine umfassende Bearbeitung durch den Anwender erfordern, die für große Datensätze nicht praktikabel ist. Damit dies verhindert wird, müssen zunächst die Kopfzeilen und anschließend die Instanzen des Schemas auf Mängel untersucht werden.

**Fehlerkorrektur durch allgemeines Wörterbuch**
Diese Methode bedient sich eines allgemeingültigen Wörterbuchs in der vom Anwender definierten Sprache.

**Fehlerkorrektur durch domänenspezifisches Wörterbuch**
In vielen kontextspezifischen Anwendungen existieren Ausdrücke, die nicht in einem allgemeinen Wörterbuch zu finden sind. Damit Fehler innerhalb dieser Begriffe automatisiert entdeckt werden können, ist es dem Anwender möglich, ein individuelles Wörterverzeichnis festzulegen, welches das allgemeine Wörterbuch ergänzt.

### Beseitigung fehlerhafter Werte

Das Beseitigen unterschiedlicher fehlerhafter Werte kann für den Anwender bei großen Datenmengen sehr aufwendig sein. Deshalb ist es notwendig, zunächst vom System eine automatisierte Fehlererkennung auszuführen, die auf mögliche Makel innerhalb der Daten hinweist. Dazu wird eine zufällig bestimmte Stichprobe von Instanzen eines Attributs zunächst hinsichtlich der enthaltenen Typen und anschließend ihrer semantischen Korrektheit untersucht.

**Fehlererkennung durch Typanalyse**
Innerhalb dieser Prozessstufe werden aus den Stichproben die Datentypen des Attributs bestimmt. Anschließend wird der Datentyp, welcher am häufigsten identifiziert werden konnte, als Klasse für das jeweilige Attribut gesetzt. Dies geschieht unter Verwendung

vordefinierter regulärer Ausdrücke prominenter Datentypen. Der Anwender kann diese Auswahl durch selbst definierte Ausdrücke ergänzen. Nachdem der Anwender die ermittelte Klasse bestätigt hat, werden alle Instanzen des Attributs auf Abweichungen überprüft. Gefundene Makel sollten unschwer erkennbar sein.

**Fehlererkennung durch Umsetzungstabellen**
Nach der Ausführung einer Typenanalyse sollten die Instanzen eines Attributs, hinsichtlich der semantischen Korrektheit, untersucht werden. Damit das System in der Lage ist, diesen Fehlertyp zuverlässig zu ermitteln, sollte der Anwender eine Umsetzungstabelle zugelassener Werte definieren.

Nach abgeschlossener Fehleranalyse kann der Anwender notwendige Schritte ergreifen, um die ermittelten Fehler zu korrigieren.

**Fehlerkorrektur durch korrelierende Attribute**
Um fehlerhafte Werte automatisiert zu korrigieren, kann der Anwender, kongruent zu Abschnitt 4.2.1, korrelierende Attribute spezifizieren.

### Eliminierung von Ausreißern

Objekte, die sich substanziell von allen anderen Objekten innerhalb des Datensatzes unterscheiden, können sich stark auf nachgelagerte Datenanalysen auswirken und daher potentiell unerwünscht sein (vgl. Abschnitt 2.3.4). Dazu kann der Anwender entweder einen zugelassenen Wertebereich definieren oder Ausreißer automatisiert vom System erkennen lassen.

**Angabe eines Wertebereichs**
Es wird ermöglicht, einen Definitionsbereich für nummerische Attribute zu spezifizieren. Alle Werte, die sich außerhalb dieses Bereichs befinden, werden vom System als Ausreißer klassifiziert. Dabei erhält der Anwender die Möglichkeit, gefundene Anomalien aus dem Datensatz zu entfernen oder sie visuell, von den übrigen Instanzen für eine manuelle Nachbearbeitung, abzugrenzen.

**Distanzberechnung zwischen den Elementen**
Diese Operation wird automatisch zu Beginn der Datenaufbereitung vom System ausgeführt. Dabei werden die semantischen Distanzen der Instanzen eines Attributes berechnet und die Elemente, die am weitesten von ihren Nachbarn entfernt sind, als Ausreißer gekennzeichnet. Der Anwender kann die vom System markierten Ausreißer entfernen, bearbeiten oder eine inkorrekte Klassifikation aufheben.

### Entfernung mangelhafter Einträge

Unvollständige Elemente innerhalb einer Datenstruktur sollten aus dem integrierten Datensatz entfernt werden, da sie keinen oder nicht ausreichend Informationsgehalt für nachfolgende Analysen bergen. Die gefundenen Einträge können aus dem Datensatz gelöscht werden oder für eine einfachere manuelle Nachbearbeitung visuell hervorgehoben werden.

**Standardisierter Mindestfüllgrad**
Das System soll automatisch bei der Initialisierung Einträge aufspüren, die einen Füllgrad von $50\%$ unterschreiten und diese als lückenhaft markieren.

**Individueller Mindestfüllgrad**
Der Anwender kann individuell für jeden der zu integrierenden Datensätze einen prozentualen Mindestfüllgrad festlegen. Sobald dieser Wert unterschritten ist, wird der jeweilige Eintrag als mangelhaft klassifiziert.

**Für den Anwender kritische Attribute**
Unter Umständen kann es für den Anwendungskontext des Anwenders entscheidend sein, dass eine Selektion besonderer Attribute innerhalb eines Datensatzes vorhanden ist. Dazu wird es ermöglicht, eine Auswahl essentieller Attribute festzulegen.

### Behebung struktureller Unzulänglichkeiten

Damit keine Mängel, aufgrund fehlender struktureller Übereinstimmungen der Datensätze im Zuge der Schemaintegration, auftreten, muss es dem Anwender ermöglicht werden, die Struktur der Daten manuell seinen Bedürfnissen anzupassen.

**Hinzufügen und Entfernen von Spalten**
Der Anwender kann neue Spalten erstellen oder bestehende aus dem Datensatz entfernen. Diese Operation ist vor allem dann notwendig, wenn ein Attribut lediglich in einem Datensatz vorhanden ist.

**Verschieben von Spalten**
Der Anwender kann die Spalten der Schemata nach Belieben verschieben, um die Übersichtlichkeit zu verbessern oder eine, für den Anwendungskontext sinnvolle Reihenfolge, zu erzeugen. Dabei soll es möglich sein, die Spalten um eine definierte Anzahl von Stellen nach rechts oder links zu verschieben oder durch Angabe eines Index die Position explizit zu bestimmen.

**Teilung und Vereinigung von Spalten**
Es ist möglich, dass unterschiedliche Domänen eine unterschiedliche Granularität von Attributen enthalten. Damit ein zufriedenstellendes Integrationsergebnis erzielt werden kann, muss der Anwender in der Lage sein, die Granularität einzelner Attribute nach Belieben zu variieren. Dies kann durch die Vereinigung oder Trennung, unter Angabe eines Trennzeichens und eines neuen Namen der bearbeiteten Attribute, realisiert werden.

**Transpositionieren von Attributen und Instanzen**
Um mögliche Redundanzen, die durch die Struktur der Daten verursacht werden, zu entfernen, kann der Anwender zwei Spalten zur Transpositionierung auswählen. Die Operation erstellt aus der ersten Auswahl neue, einzigartige Attribute und populiert sie mit den Elementen aus der zweiten Auswahl. Die übrigen Attribute und ihre Instanzen werden der Transformation entsprechend angepasst [KPHH11].

#### Berechnung der Datenqualität

Die Datenqualität unterstützt den Anwender dabei festzustellen, ob die vorliegenden Daten seinen Ansprüchen genügen (vgl. Abschnitt 2.2). Dabei soll innerhalb der Applikation die Datenqualität individuell für jedes Schemaelement ermittelt werden, die anschließend als arithmetisches Mittel die globale Qualität für den jeweiligen Datensatz bildet. Dies bietet eine informative Rückmeldung darüber, welche Elemente noch aufbereitet werden müssen, um ein, für den Anwendungskontext akzeptables Qualitätsniveau, zu erreichen.

**Standardfaktoren**
Die Applikation berechnet die Datenqualität eines Attributs unter Berücksichtigung der Vollständigkeit, Typeinheit und Korrektheit seiner Instanzen. Aus den ermittelten prozentualen Werten der Faktoren wird das arithmetische Mittel gebildet, welches die Qualität repräsentiert.

**Anwenderspezifische Faktoren**
Dem Anwender soll es ermöglicht werden, die Priorität einer Auswahl von Standardfaktoren, durch Angabe individueller Gewichtungen, zu erhöhen oder herabzustufen, um so das, für den Anwendungskontext relevante Ergebnis, zu erzielen.

### 4.2.2 Schemaintegration

Damit syntaktische Äquivalenzen zuverlässig und schnell vom Anwender erkannt werden können ist es notwendig, ihn bei umfassenden Datensätzen dabei zu unterstützen. Im nachfolgenden Abschnitt wird ein Konzept entwickelt welches diese Anforderung erfüllen soll.

#### Ermittlung geeigneter Matchkandidaten

Das im Rahmen dieser Arbeit entwickelte Verfahren für das automatisierten Schemamatching besteht aus drei verschachtelten Verarbeitungsstufen und wird in Abbildung 4.2 dargestellt.

Der Ansatz beschreibt einen hybriden Matchingalgorithmus (vgl. Abschnitt 2.4.3), der Schema-Level-Matching (vgl. Abschnitt 2.4.1) und Instance-Level-Matching (vgl. Abschnitt 2.4.2) miteinander vereint. Dabei werden anhand der Spaltennamen passende Matchkandidaten ermittelt,

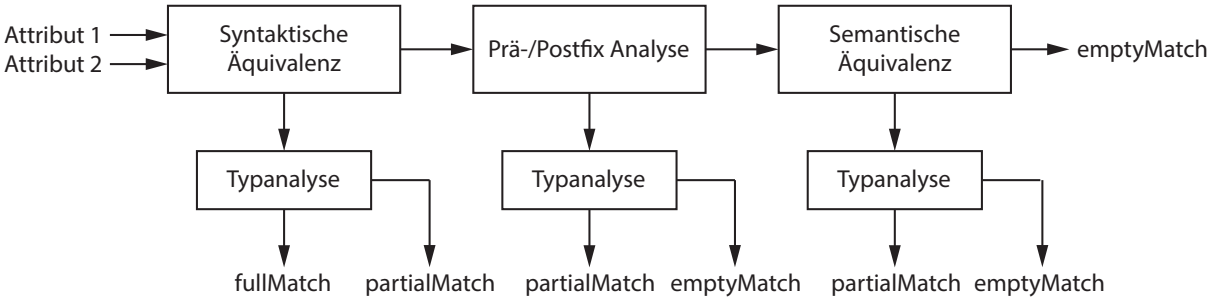



**Abbildung 4.2:** Methode zur Klassifikation von Schemaattributen

die durch eine Analyse ihrer Instanzen validiert werden. Es wird zwischen *fullMatch*, *partialMatch* und *emptyMatch* unterschieden.

Ein Match ist ein *fullMatch* genau dann, wenn die Namen der Kandidaten syntaktisch übereinstimmen und ihre Instanzen vom selben Datentyp sind. Ein Match ist ein *partialMatch* genau dann, wenn die Namen der Kandidaten entweder semantisch äquivalent oder sie Präfix oder Postfix des anderen Kandidaten sind und ihre Instanzen vom selben Datentyp sind. Ein Match ist ein *emptyMatch* genau dann, wenn weder ein syntaktisch, noch ein semantisch äquivalenter Matchkandidat gefunden wurde oder die Datentypen der gefundenen Matchkandidaten nicht übereinstimmen.

Da in der FlexMash-Umgebung nur JSON[1]-Objekte existieren, kann dieses Verfahren weder beschreibungsbasierte noch bedingungsbasierte Methoden zur Ermittlung eines geeigneten Matchkadidaten nutzen. Aus diesem Grund verwendet die beschriebene Prozedur Namensbasiertes-Matching mit nachgelagertem Instance-Level-Matching.

**Namensbasiertes Schema-Level-Matching**
Damit eine Vorauswahl über geeignete Matchkandidaten durchgeführt werden kann, müssen die Namen der Spalten auf syntaktische und semantische Äquivalenzen überprüft werden. Zunächst werden alle Spalten auf syntaktische Äquivalenz untersucht. Sobald eine Übereinstimmung gefunden wurde, werden ihre Instanzen miteinander verglichen. Wenn auch die Datentypen übereinstimmen, werden die Attribute als Schlüssel-Wert-Paar in das *fullMatch*-Array geschrieben und aus den nachfolgenden Untersuchungen ausgeschlossen.

Sollte keine Typäquivalenz festgestellt worden sein, werden die Attribute in das *partialMatch*-Array eingetragen und bleiben für nachfolgende Analysen erhalten. Sobald alle Attribute, die einem *fullMatch* entsprechen, gefunden wurden, werden die übrigen Attribute einer Präfix-/Postfix-Analyse nach Liu et al. [LWH14] unterzogen. Dabei wird untersucht, ob ein Attribut Präfix oder Postfix eines anderen Attributs ist. Ist dies der Fall, werden ihre Instanzen verglichen und alle übereinstimmenden Attribute als

[1]http://json.org

Schlüssel-Wert-Paar in das *partialMatch*-Array geschrieben und aus den nachfolgenden Untersuchungen ausgeschlossen. Sollte keine Typäquivalenz ermittelt werden können, werden die Attribute als *emptyMatch* klassifiziert.

Falls Attribute weder Präfix noch Postfix eines anderen Attributs sind, werden sie im nächsten Schritt mit Hilfe eines Thesaurus (z. B. WordNet[2]) auf semantische Äquivalenzen untersucht. Nach einer Überprüfung der Instanzen der gefundenen Attribute, werden sie bei Übereinstimmung in das *partialMatch*-Array geschrieben. Alle, nach den obigen Verarbeitungsschritten übrig bleibenden Attribute, werden als *emptyMatch* klassifiziert.

**Validierung gefundener Matches durch Instance-Level-Matching**
Um die Wahrscheinlichkeit eines inkorrekten Matches zu minimieren, werden in dieser Stufe die Instanzen der zuvor bestimmten Matchkandidaten auf Äquivalenz ihrer Datentypen hin untersucht. Dabei werden zufällig ausgewählte Stichproben von Instanzen beider Matchkandidaten, unter Verwendung regulärer Ausdrücke, miteinander verglichen.

Die Applikation besitzt bereits eine Auswahl vordefinierter regulärer Ausdrücke, die durch anwenderspezifische Ausdrücke ergänzt werden kann. Das Erweitern der Auswahl kann die Flexibilität und Präzision des Match-Vorgangs erhöhen.

### Validierung gefundener Matches durch den Anwender

Nach der Initialklassifikation ist es dem Anwender möglich, die vom System gefundenen Matches zu bestätigen oder in Teilen durch eigene Relationen zu verändern. Die vom Anwender kontrollierten oder manuell definierten Matches werden vom System als *finalMatch* klassifiziert und visuell von den übrigen Äquivalenzen abgegrenzt.

Der Anwender sollte im Zuge des Matchingprozesses aus den Vorschlägen im *partialMatch*-Array den entsprechenden Kandidaten bestimmen. Attribute, für die durch das automatisierte Verfahren kein Matchkandidat ermittelt werden konnte, können vom Anwender bestätigt werden, durch eine definierte Relation ergänzt werden oder aus dem Integrationsresultat entfernt werden. Eine Bestätigung als *emptyMatch* wirkt sich negativ auf die Datenqualität des integrierten Resultats aus und sollte möglichst vermieden werden.

Basierend auf den bestätigten bzw. ausgewählten *finalMatches* werden die Inhalte der Datenquelle vereinigt und zur Nachbereitung weitergegeben.

[2] http://wordnet.princeton.edu

### 4.2.3 Aufzeichnung vorgenommener Transformationsoperationen

Es kann zu unterschiedlichen Zwecken hilfreich sein, vorgenommene Operationen festzuhalten. Dazu gehören vor allem Dokumentation und nachgelagerte Fehleranalysen. Außerdem ermöglicht es, eine gespeicherte Chronik durchgeführter Transformationen andere Datensätze aus identischen Domänen vollautomatisch aufzubereiten, zu integrieren und nachzubearbeiten. Damit die unterschiedlichsten Operationen der Applikation so präzise wie möglich dargestellt werden können, sollten Operationen zur Datenaufbereitung und zur Datenintegration jeweils über eine eigene Syntax verfügen.

**Syntax der Datenaufbereitung**
Damit der Anwender die ausgeführten Transformationsoperationen problemlos nachvollziehen kann, muss jeder Eintrag in der Historie die vorgenommene Operation, die verwendeten Parameter und den Anwendungsbereich im Datensatz enthalten.

$$Entry = operation\ [-\{parameter\}] - tableName\ [tableColumn]\ [tableCell]$$

Ein Beispiel für einen korrekten Eintrag einer Operation, die das Zeichen @ aus der Spalte $email$ eines Datensatzes entfernt, wäre:

$$Entry = defineIllegalCharacter - @ - dataSource1, email$$

Dadurch, dass sich Datensätze gleicher Domänen stark unterscheiden können, ist es bei einer historienbasierten Datenaufbereitung wichtig, ausschließlich Operationen zu betrachten, die entweder auf den gesamten Datensatz oder auf Attribute angewandt wurden.

**Syntax der Datenintegration**
Damit die definierten Matches eindeutig zu unterscheiden sind, muss die Historie die Namen des Datensatzes sowie die semantisch äquivalenten Attribute umfassen.

$$Entry = tableName1, attribute1 \Leftrightarrow tableName2, attribute2$$

Sollte der Anwender einen *emptyMatch* bestätigen und in das Integrationsergebnis aufnehmen, enthält der Eintrag den Namen des Datensatzes, das Attribut und das Schlüsselwort *empty*.

$$Entry = tableName1, attribute1 \Leftrightarrow empty$$

# 5 Prototypische Implementierung

Dieses Kapitel beschreibt die prototypische Implementierung einer Auswahl einiger im vorigen Kapitel erarbeiteten Konzepte. Die Selektion umfasst die Interaktion mit dem Anwender sowie das automatisierte Matchingverfahren.

## 5.1 Interaktion

In diesem Abschnitt wird die Art und Weise beschrieben, mit der ein Anwender mit der Benutzeroberfläche der Applikation interagiert, um zwei Datensätze aufzubereiten, zu integrieren und anschließend das Resultat nachzubearbeiten.

### 5.1.1 Datenaufbereitung

Im Zuge der Datenaufbereitung sollen zunächst qualitative Mängel und strukturelle Ungleichheiten der Datenquellen behoben werden. Dieser Schritt dient einerseits als Vorbereitung für die nachfolgende computergestützte Schemaintegration, andererseits als Nachbearbeitung des Integrationsresultats.

#### Benutzeroberfläche Vorverarbeitung

Die in Abbildung 5.1 dargestellte Benutzeroberfläche enthält drei Tabs *(1)*, die zur Navigation zwischen den einzelnen Verarbeitungsstufen verwendet werden. Unmittelbar unter den Navigationselementen befindet sich die visuelle Darstellung der zu integrierenden Datensätze in tabellarischer Form. Jeder Datensatz verfügt unterhalb seines Namens *(2)* über einen individuellen Indikator *(3)*, der die jeweilige Datenqualität anzeigt. Außerdem enthält jedes Attribut*(4)* eine eigene Qualitätsanzeige, die dem Anwender als Anhaltspunkt dienen kann, welcher Teil des Datensatzes noch bearbeitet werden sollte. Im darunterliegenden Bereich befindet sich der gegenwärtige Zustand der zu integrierenden Daten *(5)*. Auf der rechten Seite befindet sich die Transformationshistorie *(6)* mit Einträgen zu den durchgeführten Operationen.

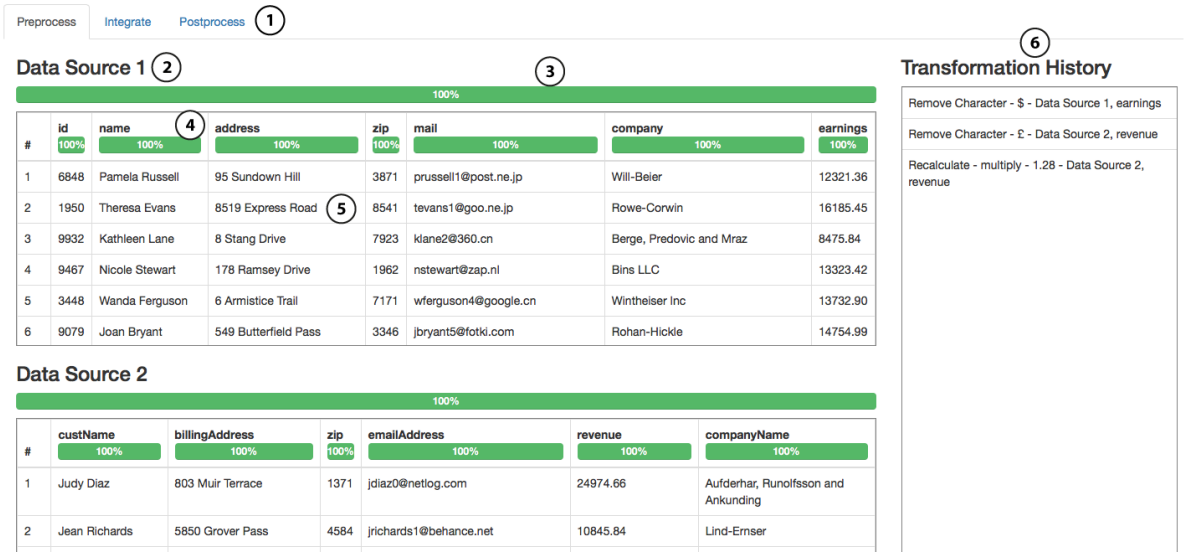



**Abbildung 5.1:** Benutzeroberfläche Vorverarbeitung

## Benutzeroberfläche Nachbereitung

Die in Abbildung 5.2 dargestellte Bedienoberfläche enthält Teile der in Abbildung 5.1 beschriebenen Elemente (*(1)*, *(2)*, *(3)*, *(4)*, *(6)*). Im Gegensatz zur Oberfläche der Datenaufbereitung besitzt diese Benutzeroberfläche nur den gegenwärtigen Zustand der integrierten Daten *(7)*.

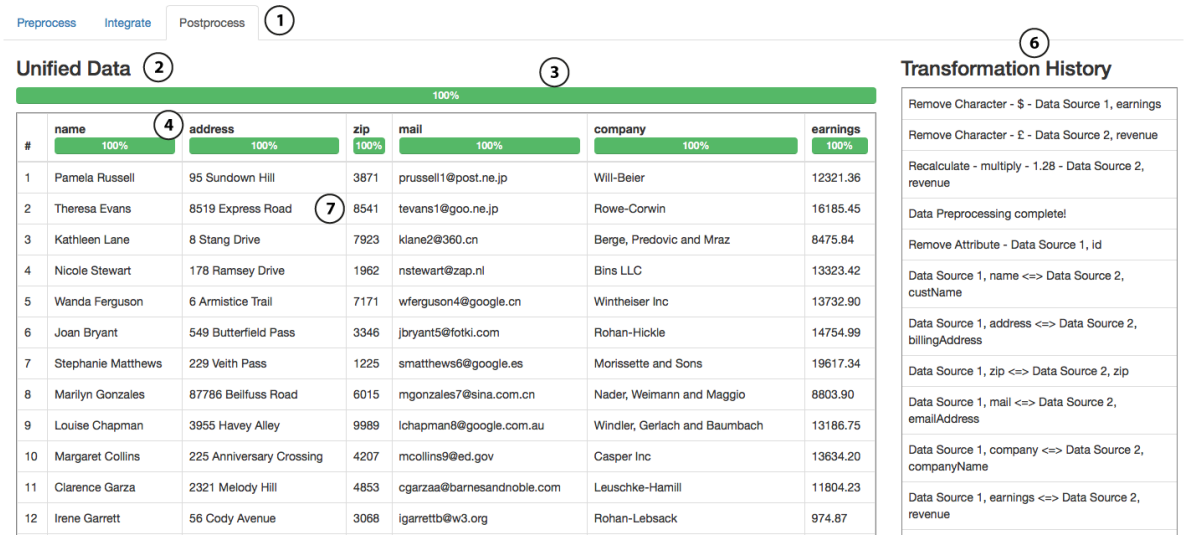
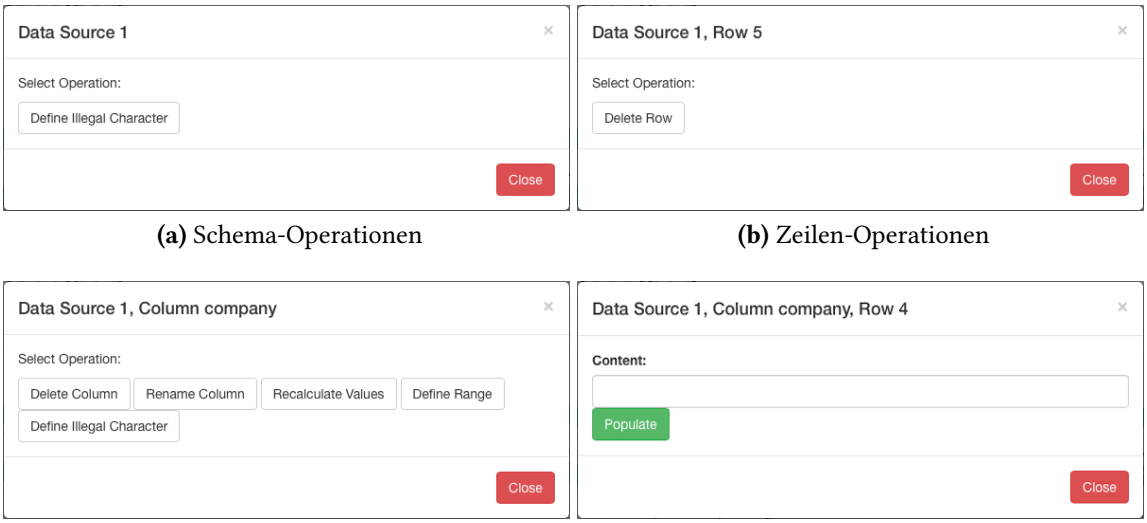



**Abbildung 5.2:** Benutzeroberfläche Nachbearbeitung




**(a)** Schema-Operationen **(b)** Zeilen-Operationen

**(c)** Spalten-Operationen **(d)** Zellen-Operationen

**Abbildung 5.3:** Menüs der Operationen zur Datenaufbereitung

### Operationen zur Datenaufbereitung

Für die computergestützte Datenaufbereitung kann unter den nachfolgenden Operation ausgewählt werden:

**Schema-Operationen**
Durch Auswahl des Namens der Datenquelle öffnet sich ein Menü, aus dem die folgende Operation ausgewählt werden kann (vgl. Abbildung 5.3a):

**Entfernen unerlaubter Zeichen**
Diese Prozedur ermöglicht es dem Anwender eine Folge nicht zulässiger Zeichen in einem Eingabefeld zu definieren, die nach Bestätigung aus allen Einträgen des Schemas entfernt werden. Des Weiteren können erlaubte Zeichen durch einen regulären Ausdruck definiert werden, die nach der Operation im Schema verbleiben.

**Spalten-Operationen**
Durch Selektion des Namens des Attributs öffnet sich ein Menü, aus dem eine der folgenden Operationen ausgewählt werden kann (vgl. Abbildung 5.3c):

**Löschen von Spalten**
Durch die Auswahl dieser Operation wird das Attribut vollständig aus dem Datensatz entfernt.

**Umbenennen von Spalten**
Durch Anwendung dieser Prozedur wird der Name des gewählten Attributs in einem Eingabefeld angezeigt und kann beliebig verändert werden.

**Neuberechnen enthaltener Werte**
Hierbei kann derzeit eine der vier Grundrechenarten aus einem Dropdown-Menü ausgewählt werden. Anschließend muss ein Wert innerhalb eines Eingabefelds spezifiziert werden, mit dem die gewählte Rechenoperation auf jeden Eintrag der Spalte angewandt wird.

**Markieren von Ausreißern**
Die Operation markiert alle numerischen Werte, die sich außerhalb des spezifizierten Definitionsbereichs befinden, mit roter Farbe. Außerdem können alle Einträge, deren Attribut außerhalb des Wertebereichs liegt, automatisch vom System entfernt werden.

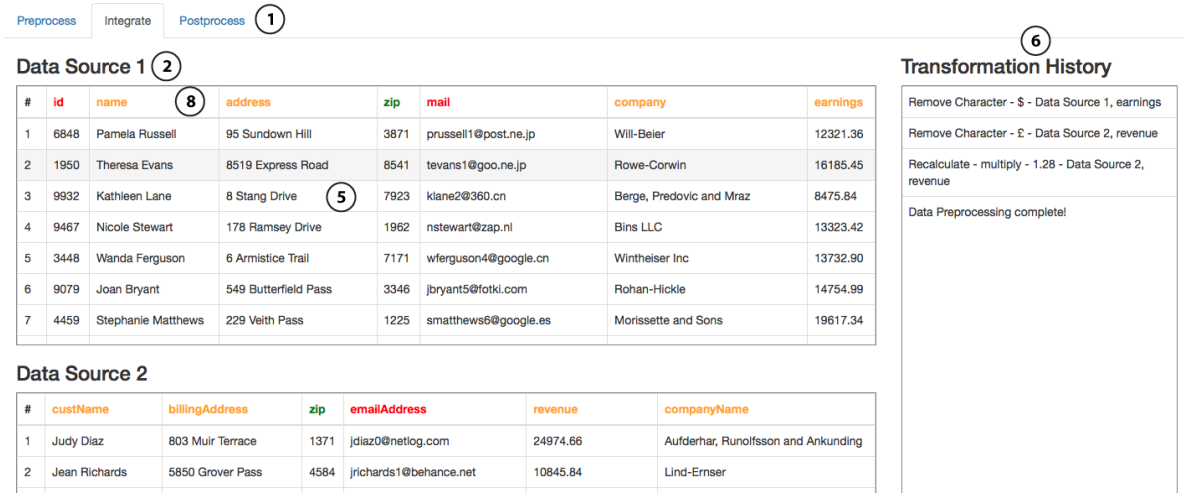
**Entfernen unerlaubter Zeichen**
Diese Operation funktioniert kongruent zu der Prozedur auf Schemaebene.

**Zeilen-Operationen**
Durch Auswahl der Zeilennummer eines Schemas öffnet sich ein Menü aus dem die folgende Operation ausgewählt werden kann (vgl. Abbildung 5.3b):

**Zeile löschen**
Die spezifizierte Zeile kann vollständig aus dem Schema entfernt werden.

**Zellen-Operationen**
Durch Selektion einer beliebigen Zelle innerhalb der beiden Schemata öffnet sich ein Menü aus dem die folgende Operation ausgewählt werden kann (vgl. Abbildung 5.3d):

**Manuelles Eintragen eines Werts**
Wenn die Zelle leer ist kann diese über ein Eingabefeld mit dem geeigneten Wert befüllt werden. Falls jedoch bereits ein Wert enthalten ist kann dieser manuell über ein Eingabefeld verändert werden.




| # | id | name | address | zip | mail | company | earnings |
|---|---|---|---|---|---|---|---|
| 1 | 6848 | Pamela Russell | 95 Sundown Hill | 3871 | prussell1@post.ne.jp | Will-Beier | 12321.36 |
| 2 | 1950 | Theresa Evans | 8519 Express Road | 8541 | tevans1@goo.ne.jp | Rowe-Corwin | 16185.45 |
| 3 | 9932 | Kathleen Lane | 8 Stang Drive | 7923 | klane2@360.cn | Berge, Predovic and Mraz | 8475.84 |
| 4 | 9467 | Nicole Stewart | 178 Ramsey Drive | 1962 | nstewart@zap.nl | Bins LLC | 13323.42 |
| 5 | 3448 | Wanda Ferguson | 6 Armistice Trail | 7171 | wferguson4@google.cn | Wintheiser Inc | 13732.90 |
| 6 | 9079 | Joan Bryant | 549 Butterfield Pass | 3346 | jbryant5@fotki.com | Rohan-Hickle | 14754.99 |
| 7 | 4459 | Stephanie Matthews | 229 Veith Pass | 1225 | smatthews6@google.es | Morissette and Sons | 19617.34 |

| # | custName | billingAddress | zip | emailAddress | revenue | companyName |
|---|---|---|---|---|---|---|
| 1 | Judy Diaz | 803 Muir Terrace | 1371 | jdiaz0@netlog.com | 24974.66 | Aufderhar, Runolfsson and Ankunding |
| 2 | Jean Richards | 5850 Grover Pass | 4584 | jrichards1@behance.net | 10845.84 | Lind-Ernser |

**Abbildung 5.4:** Benutzeroberfläche Datenintegration

## 5.1.2 Datenintegration

Im Zuge der Datenintegration werden zunächst automatisch semantische Äquivalenzen zwischen den Attributen vom System ermittelt und für den Anwender durch unterschiedliche Farbwerte kenntlich gemacht. Anschließend ist der Anwender dazu angehalten, durch Nutzung seines Domänenwissens, die vom System ermittelten Kongruenzen manuell zu überarbeiten, um ein korrektes Integrationsergebnis zu erzielen.

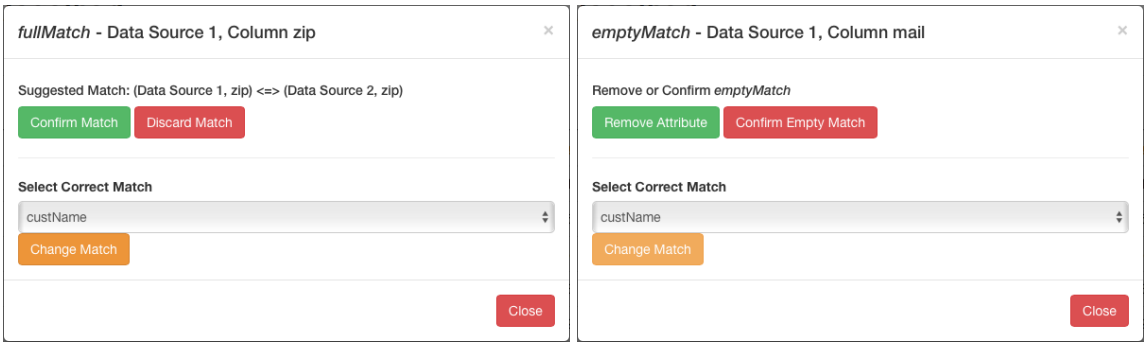
### Benutzeroberfläche

Die in Abbildung 5.4 dargestellte Benutzeroberfläche enthält drei Navigations-Tabs *(1)*. Darunter befinden sich die tabellarischen Darstellungen der Schemata *(5)*, mit den farblich hervorgehobenen semantischen Äquivalenzen ihrer Attribute *(8)*. Auf der rechten Seite befindet sich die Transformationshistorie *(6)* mit den bisher durchgeführten Operationen.

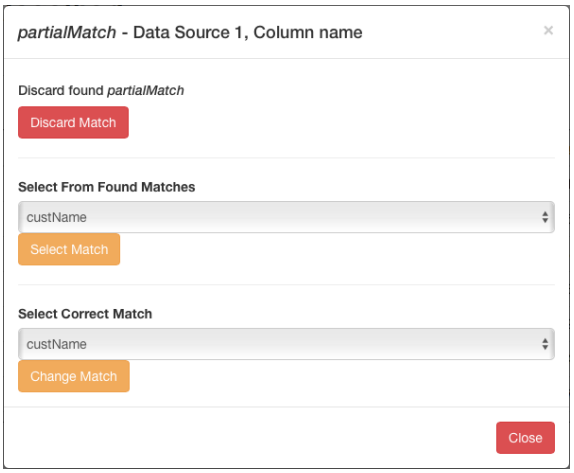
### Operationen zur Datenintegration

Damit der Anwender das System beim Ermitteln semantischer Äquivalenzen unterstützen kann, werden eine Reihe von Operationen, basierend auf der Klassifikation der Attribute, bereitgestellt.

***fullMatch*-Operationen**
Durch das Auswählen eines Attributs in grüner Schrift öffnet sich ein Menü mit folgenden Operationen (vgl. Abbildung 5.5a):




**(a)** fullMatch-Operationen **(b)** emptyMatch-Operationen




**(c)** partialMatch-Operationen

**Abbildung 5.5:** Menüs der Operationen zur Datenintegration

**Bestätigung der Relation**
Durch Bestätigen wird die vom System gefundene semantische Äquivalenz als korrekt markiert.

**Definition neuer Relation**
Durch die manuelle Auswahl über ein Dropdown-Menü können gänzlich neue Relationen definiert werden, falls die vom System gefundenen Übereinstimmungen inkorrekt sind.

**Verwerfen ermittelter Relation**
Die gefundene Übereinstimmung kann explizit verworfen werden. Dadurch wird das betreffende Attribut als *emptyMatch* klassifiziert und rot gefärbt.

***partialMatch*-Operationen**
Durch das Selektieren eines Attributs in orangenefarbener Schrift öffnet sich ein Menü mit folgenden Operationen (vgl. Abbildung 5.5c):

**Auswahl korrekter Relation**
Aus dem Dropdown-Menü können, die vom System vorgeschlagenen Übereinstimmungen des jeweiligen Attributs, ausgewählt und bestätigt werden.

**Definition neuer Relation**
Diese Operation funktioniert kongruent zu der bei den *fullMatch*-Operationen erwähnten Prozedur.

**Verwerfen ermittelter Relationen**
Durch Betätigen des *Discard Match*-Knopfes werden die gefundenen Relationen gelöscht und das ausgewählte Attribut, sowie alle ermittelten Übereinstimmungen, als *emptyMatch* klassifiziert und rot gefärbt.

***emptyMatch*-Operationen**
Durch das Auswählen eines Attributs in roter Schrift öffnet sich ein Menü mit folgenden Operationen (vgl. Abbildung 5.5b):

**Bestätigung der Relation**
Durch diese Operation wird bestätigt, dass das jeweilige Attribut keine semantische Äquivalenz innerhalb des anderen Schemas besitzt. Dabei wird eine geminderte Vollständigkeit der Daten in Kauf genommen.

**Definition neuer Relation**
Diese Funktion bedient sich dem selben Funktionsprinzip der Prozedur aus den *fullMatch*-Operationen.

**Entfernung des Attributs**
Falls im Zuge der Datenaufbereitung vor der Integration versäumt wurde Attribute zu entfernen die kein semantisches Äquivalent innerhalb des anderen Schemas besitzen, existiert die Möglichkeit Spalten aus dem jeweiligen Schema, durch Betätigen des *Delete Attribute*-Knopfes zu entfernen.

Nach der Kontrolle aller vom System ermittelten Relationen durch den Anwender, werden die Attribute, welche als *finalMatch* klassifiziert wurden, dazu genutzt, die Attribute der Schemata in ein neues vereinigtes Schema zu überführen. Dabei werden semantisch äquivalente Attribute des zweiten Schemas an die Attribute des ersten Schemas angehängt.

## 5.2 Automatisiertes Matchingverfahren

Dieser Abschnitt beschreibt eine beispielhafte Anwendung des entwickelten Konzepts zur automatisierten Erkennung und Zusammenführung semantischer Äquivalenzen von Attributen aus unterschiedlichen Datensätzen. Dazu werden im Folgenden zwei bereits aufbereitete Datensätze Tabelle 5.1 und Tabelle 5.2 verwendet, um die Interaktion mit dem Domänenexperten und dem System darzustellen. Weiterhin wird auf das in Abschnitt 1.1 beschriebene Motivationsszenario Bezug genommen.

Sobald der Anwender die Aufbereitung der Datensätze abgeschlossen hat und zur Ansicht der Datenintegration wechselt, führt das System eine Initialklassifizierung durch. Dabei betrachtet der Algorithmus in der ersten Verarbeitungsstufe die syntaktische Äquivalenz der Attribute aus Tabelle 5.1 und Tabelle 5.2.

| id | name | address | zip | mail | company | earnings |
|---|---|---|---|---|---|---|
| 5947 | Pamela Clark | 61 South Alley | 3421 | pclark@jimdo.it | Skinix | 17349.11 |
| 2687 | Lyn Flores | 3 Farwell Ave | 8367 | cflores@disqus.to | Jayo | 19749.89 |
| 5988 | John Carroll | 40 Fall Pass | 4269 | jcarr@post.jp | CloudRed | 6037.71 |

**Tabelle 5.1:** Kundendatenbank 1 nach der Initialklassifikation

| custName | billingAddress | zip | emailAddress | revenue | companyName |
|---|---|---|---|---|---|
| Gloria Fuller | 66 Knutson Ave | 7489 | gfuller@scrib.de | 8270.06 | Skynode |
| John Henry | 54 Dixon Lane | 5238 | jhenry@moza.pl | 1924.86 | Babble |
| Lisa James | 9 Mill Park | 9378 | ejames@talk.fr | 6357.92 | Jaxnat |

**Tabelle 5.2:** Kundendatenbank 2 nach der Initialklassifikation

Die Attribute *zip* gelten als *fullMatch*, da sie in beiden Datenstrukturen einzigartig und weder Präfix noch Postfix eines anderen Attributs sind, sowie die Instanzen beider Attribute vom Typ *number* sind. Das Attribut *name* aus Tabelle 5.1 wird in der *Prä-/Postfix-Analyse* und der nachfolgenden Typüberprüfung als *partialMatch* von *custName* und *companyName* aus Tabelle 5.2 definiert. In Tabelle 5.2 werden zunächst *billingAddress* und *emailAddress* als *partialMatch* von *address* aus Tabelle 5.1 klassifiziert. Erst die nachfolgende Analyse der Instanzen identifiziert *billingAddress* als Attribut vom Typ Adresse und somit wahrscheinlicheren Matchkandidaten für *billingAddress*. Dabei wird *emailAddress* als *emptyMatch* klassifiziert und rot gefärbt. Das Attribut *company* aus Tabelle 5.1 ist ein Präfix von *companyName* aus Tabelle 5.2 und somit werden die Attribute durch die anschließende Prüfung ihrer Instanzen als *partialMatch* klassifiziert. Die Attribute *earnings* aus Tabelle 5.1 und *revenue* aus Tabelle 5.2 werden durch einen Thesaurus als semantisch äquivalent klassifiziert. Die nachfolgende Instanzprüfung stellt eine

Typäquivalenz fest und somit sind sie ein *partialMatch*. Die Attribute *id* und *mail* in Tabelle 5.1 erfüllen keinerlei spezifizierte Matchkriterien und werden als *emptyMatch* klassifiziert.

Nachdem die Klassifikation durch das System abgeschlossen ist, wird der Anwender dazu angehalten, die gefundenen Äquivalenzen zu bestätigen oder aus einer Liste die korrekte Übereinstimmung auszuwählen. Attribute, die keine Korrespondenzen besitzen, können hierbei aus den Schemata entfernt werden.

Das Attribut *id* aus Tabelle 5.1 besitzt kein Äquivalent in Tabelle 5.2 und wird vom Anwender gelöscht. Für das Attribut *name* in Tabelle 5.1 wählt der Anwender das nach seinem Domänenwissen passende Attribut *custName*. Anschließend werden *address*, *zip*, *company* sowie *earnings* aus Tabelle 5.1 und *billingAddress*, *zip*, *companyName* sowie *revenue* aus Tabelle 5.2 als gültige Äquivalenzen vom Anwender bestätigt. Anschließend wählt der Anwender für das, als *emptyMatch* klassifizierte Attribut *mail* aus Tabelle 5.1 manuell, unter Verwendung seines Domänenwissens, das ebenfalls als *emptyMatch* klassifizierte Attribut *emailAddress* als Übereinstimmung aus.

Somit ist der Integrationsprozess abgeschlossen und das System überführt die Datenquellen Tabelle 5.1 und Tabelle 5.2, basierend auf den vom Anwender definierten semantischen Äquivalenzen, in ein integriertes Schema Tabelle 5.3.

| **name** | **address** | **zip** | **mail** | **company** | **earnings** |
|---|---|---|---|---|---|
| Pamela Clark | 61 South Alley | 3421 | pclark@jimdo.it | Skinix | 17349.11 |
| Lyn Flores | 3 Farwell Ave | 8367 | cflores@disqus.to | Jayo | 19749.89 |
| John Carroll | 40 Fall Pass | 4269 | jcarr@post.jp | CloudRed | 6037.71 |
| Gloria Fuller | 66 Knutson Ave | 7489 | gfuller@scrib.de | Skynode | 8270.06 |
| John Henry | 54 Dixon Lane | 5238 | jhenry@moza.pl | Babble | 1924.86 |
| Lisa James | 9 Mill Park | 9378 | ejames@talk.fr | Jaxnat | 6357.92 |

**Tabelle 5.3:** Integrierte Kundendatenbank

# 6 Zusammenfassung und Ausblick

In dieser Arbeit wird ein geeignetes Konzept entwickelt, um es Anwendern, ohne umfassendes technisches Hintergrundwissen, ermöglicht, heterogene Datenquellen computergestützt aufzubereiten und miteinander zu vereinen und dabei stets die Kontrolle über den Prozess zu haben.

Im Zuge einer umfassenden Literaturrecherche werden bekannte Konzepte zur Datenaufbereitung und zur Schemaintegration untersucht, die eine Unterstützung des Anwenders ermöglichen. Das vorgestellte Konzept wird in drei grundlegende Verarbeitungsebenen unterteilt, welche durch unterschiedliche Funktionen den Anwender bei der Ausführung seiner Aufgaben helfen. Dabei ist es stets möglich, diese Funktionen durch Variation der Parameter zu verändern oder durch zusätzliche, domänenspezifische Definitionen zu erweitern. Besonderes Augenmerk liegt auf dem semiautomatischen Verfahren zur Unterstützung des Anwenders beim Ermitteln semantischer Äquivalenzen zwischen zwei Datensätzen.

Die Applikation unterteilt die gefundenen Übereinstimmungen in vier unterschiedliche Klassen, die in dieser Arbeit entwickelt werden. Anschließend wird, unter Verwendung des Data Mashup-Ansatzes, eine prototypische Implementierung vorgestellt, die einige der zuvor erarbeiteten Funktionalitäten in einer graphischen Oberfläche realisiert.

Um dem Anwender eine einfache und intuitive Interaktion zu ermöglichen, werden die Daten innerhalb der Benutzeroberfläche in tabellarischer Form dargestellt. Bei der Implementierung werden, neben dem Klassifikationsalgorithmus, hauptsächlich Operationen zur Datenaufbereitung umgesetzt, die eine Interaktion mit dem Anwender voraussetzen.

Durch das vorgestellte Konzept ist es nicht mehr notwendig die Aufbereitung und Integration von Daten an IT-Experten, ohne Kenntnis über die Daten, zu delegieren, sondern kann unmittelbar von einem Domänenexperten durchgeführt werden. Dies reduziert die Anzahl der involvierten Personen und kann zu einer Senkung der Fehlerzahl, Arbeitszeit sowie der Personalkosten führen.

## Ausblick

Bei der prototypischen Implementierung des entwickelten Konzepts wurden nur ausgewählte Funktionen umgesetzt, weshalb die Funktionalität ausgebaut werden sollte, um mehr Anwendungsfälle abdecken zu können.

Momentan dient die Transformationshistorie als informative Rückmeldung für den Anwender bei der Durchführung und Nachvollziehbarkeit einer Datenintegration. Um Anpassungen während des Integrationsprozesses zu ermöglichen, sollten die Parameter eines Eintrags der Historie dynamisch verändert werden können und alle nachfolgenden Schritte erneut ausgeführt werden, bis ein Konflikt auftritt.

Die Validierung der gefundenen Matchkadidaten wird im dargestellten Konzept durch eine Klassifikation der Datentypen, mit Hilfe regulärer Ausdrücke, bewerkstelligt. Dieser Ansatz könnte durch maschinelles Lernen präzisiert werden. Hierbei könnten domänenspezifische Datentypen automatisiert erkannt und dadurch die definierte Auswahl regulärer Ausdrücke ergänzt werden.

Da die Operationen zur Datenaufbereitung bei umfassenden Datensätzen sehr ressourcenintensiv sind, wäre es sinnvoll, sie weitestgehend zu parallelisieren, um die Wartezeiten für den Anwender zu minimieren. An dieser Stelle ist es notwendig festzustellen, welche Operationen für eine nebenläufige Ausführung geeignet sind.

Um die praktische Anwendbarkeit des vorgestellten Konzepts zu untersuchen wäre es sinnvoll, eine Nutzerstudie durchzuführen. Hierbei gilt zu prüfen, ob das entwickelte Bedienkonzept auch auf sehr große Datensätze anwendbar ist und die definierten Operationen für eine erfolgreiche Datenintegration ausreichend sind.

# Literaturverzeichnis


[APTD11] M. I. Ali, R. Pichler, H.-L. Truong, S. Dustdar. „On Integrating Data Services Using Data Mashups“. In: *British National Conference on Databases*. Springer. 2011, S. 132–135 (zitiert auf S. 24).

[BH08] P. A. Bernstein, L. M. Haas. „Information Integration in the Enterprise“. In: *Communications of the ACM* 51.9 (2008), S. 72–79 (zitiert auf S. 13).

[BR99] C. Becchetti, L. P. Ricotti. *Speech Recognition: Theory and C++ Implementation*. Wiley, 1999 (zitiert auf S. 17).

[BS16] C. Batini, M. Scannapieco. „Data Quality Dimensions“. In: *Data and Information Quality*. Springer, 2016, S. 21–51 (zitiert auf S. 15).

[Chr12] P. Christen. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer Science & Business Media, 2012 (zitiert auf S. 15, 16, 18).

[CTT09] Y. Chan, J. Talburt, T. M. Talley. *Data Engineering: Mining, Information and Intelligence*. Bd. 132. Springer Science & Business Media, 2009 (zitiert auf S. 13, 17).

[Dam64] F. J. Damerau. „A Technique for Computer Detection and Correction of Spelling Errors“. In: *Communications of the ACM* 7.3 (1964), S. 171–176 (zitiert auf S. 17).

[DM14] F. Daniel, M. Matera. *Mashups: Concepts, Models and Architectures*. Springer, 2014 (zitiert auf S. 11, 24).

[HM16] P. Hirmer, B. Mitschang. „FlexMash–Flexible Data Mashups Based on Pattern-Based Model Transformation“. In: *Rapid Mashup Development Tools*. Springer, 2016, S. 12–30 (zitiert auf S. 25).

[HPK11] J. Han, J. Pei, M. Kamber. *Data Mining: Concepts and Techniques*. Elsevier, 2011 (zitiert auf S. 15, 18–20).

[HRWM15] P. Hirmer, P. Reimann, M. Wieland, B. Mitschang. „Extended Techniques for Flexible Modeling and Execution of Data Mashups“. In: *Proceedings of 4th International Conference on Data Management Technologies and Applications*. SCITEPRESS - Science, 2015, S. 111–122 (zitiert auf S. 25).

[HS98] M. A. Hernández, S. J. Stolfo. „Real-World Data is Dirty: Data Cleansing and the Merge/Purge Problem“. In: *Data Mining and Knowledge Discovery* 2.1 (1998), S. 9–37 (zitiert auf S. 21).

[Jar89] M. A. Jaro. „Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida“. In: *Journal of the American Statistical Association* 84.406 (1989), S. 414–420 (zitiert auf S. 21).

[KHP+11] S. Kandel, J. Heer, C. Plaisant, J. Kennedy, F. van Ham, N. H. Riche, C. Weaver, B. Lee, D. Brodbeck, P. Buono. „Research Directions in Data Wrangling: Visualizations and Transformations for usable and credible data“. In: *Information Visualization* 10.4 (2011), S. 271–288 (zitiert auf S. 15).

[Knu98] D. E. Knuth. *The Art of Computer Programming: Volume 3: Sorting and Searching*. Pearson Education, 1998 (zitiert auf S. 16).

[KPHH11] S. Kandel, A. Paepcke, J. Hellerstein, J. Heer. „Wrangler: Interactive Visual Specification of Data Transformation Scripts“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2011, S. 3363–3372 (zitiert auf S. 11, 26, 36).

[Kuk92] K. Kukich. „Techniques for Automatically Correcting Words in Text“. In: *ACM Computing Surveys (CSUR)* 24.4 (1992), S. 377–439 (zitiert auf S. 16, 17).

[KZS+10] H. Köhler, X. Zhou, S. Sadiq, Y. Shu, K. Taylor. „Sampling Dirty Data for Matching Attributes“. In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. ACM. 2010, S. 63–74 (zitiert auf S. 22).

[Lev66] V. I. Levenshtein. „Binary Codes Capable of Correcting Deletions, Insertions, and Reversals“. In: *Soviet physics doklady*. Bd. 10. 8. 1966, S. 707–710 (zitiert auf S. 17).

[LNE89] J. A. Larson, S. B. Navathe, R. Elmasri. „A theory of Attributed Equivalence in Databases with Application to Schema Integration“. In: *IEEE Transactions on software engineering* 15.4 (1989), S. 449–463 (zitiert auf S. 22).

[LWH14] C. Liu, J. Wang, Y. Han. „Mashroom+: An Interactive Data Mashup Approach with Uncertainty Handling“. In: *Journal of Grid Computing* 12.2 (2014), S. 221–244 (zitiert auf S. 11, 27, 37).

[MC86] R. Morris, L. L. Cherry. „Computer Detection of Typographical Errors“. In: *IEEE Transactions on Professional Communication* PC-18 (1986), S. 54–56 (zitiert auf S. 16).

[MCB+11] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, A. H. Byers. *Big data: The Next Frontier for Innovation, Competition, and Productivity*. Hrsg. von M. G. Institute. 2011. URL: http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation (zitiert auf S. 11).

[NH10] F. Naumann, M. Herschel. „An Introduction to Duplicate Detection“. In: *Synthesis Lectures on Data Management* 2.1 (2010), S. 1–87 (zitiert auf S. 20, 21).

[Phi90] L. Philips. „Hanging on the Metaphone“. In: *Computer Language* 7.12 (1990), S. 39–43 (zitiert auf S. 17).

[PLW02] L. L. Pipino, Y. W. Lee, R. Y. Wang. „Data quality assessment“. In: *Communications of the ACM* 45.4 (2002), S. 211–218 (zitiert auf S. 15).

[Pyl99] D. Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann, 1999 (zitiert auf S. 18).

[RB01] E. Rahm, P. A. Bernstein. „A Survey of Approaches to Automatic Schema Matching“. In: *The VLDB Journal* 10.4 (2001), S. 334–350 (zitiert auf S. 11, 13, 14, 21–23).

[RD00] E. Rahm, H. H. Do. „Data Cleaning: Problems and Current Approaches“. In: *IEEE Data Engineering Bulletin* 23.4 (2000), S. 3–13 (zitiert auf S. 15).

[Shv04] P. Shvaiko. „A Classification of Schema-Based Matching Approaches“. In: (2004) (zitiert auf S. 21, 23).

[WS96] R. Y. Wang, D. M. Strong. „Beyond Accuracy: What Data Quality Means to Data Consumers“. In: *Journal of management information systems* 12.4 (1996), S. 5–33 (zitiert auf S. 14).

[YF83] E. J. Yannakoudakis, D. Fawthrop. „The Rules of Spelling Errors“. In: *Information Processing & Management* 19.2 (1983), S. 87–99 (zitiert auf S. 17).

Alle URLs wurden zuletzt am 24. 04. 2017 geprüft.

**Erklärung**

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

---

Ort, Datum, Unterschrift